MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

L1

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :I-Introduction

Date of Lecture:

Topic of Lecture:Notion of an Algorithm

Introduction :

An algorithm is a sequence of unambiguous instruction for solving a problem, for obtaining a required output for any legitimate input in a finite amount of time.

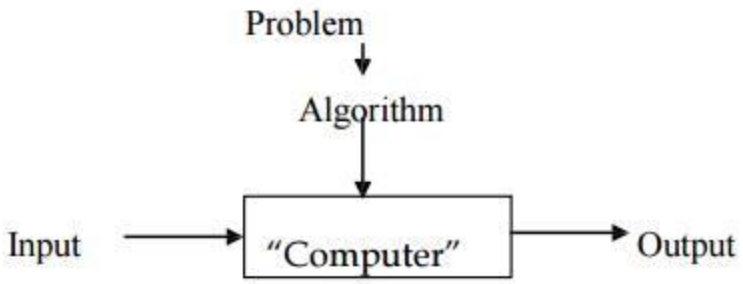
Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Understanding of Algorithm

An algorithm is a sequence of unambiguous instruction for solving a problem, for obtaining a required output for any legitimate input in a finite amount of time.



Algorithm design and analysis process

Video Content / Details of website for further learning (if any):
https://www.brainkart.com/article/Introduction-and-Definition-of-Algorithm_7964/
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

L2

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :I-Introduction

Date of Lecture:

Topic of Lecture: Fundamentals of Algorithmic Problem Solving

Introduction :

- First thing you need to do before designing an algorithm is to understand completely the problem given.
- Read the problem's description carefully and ask questions if you have any doubts about the problem, do a few small examples by hand, think about special cases, and ask questions again if needed.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

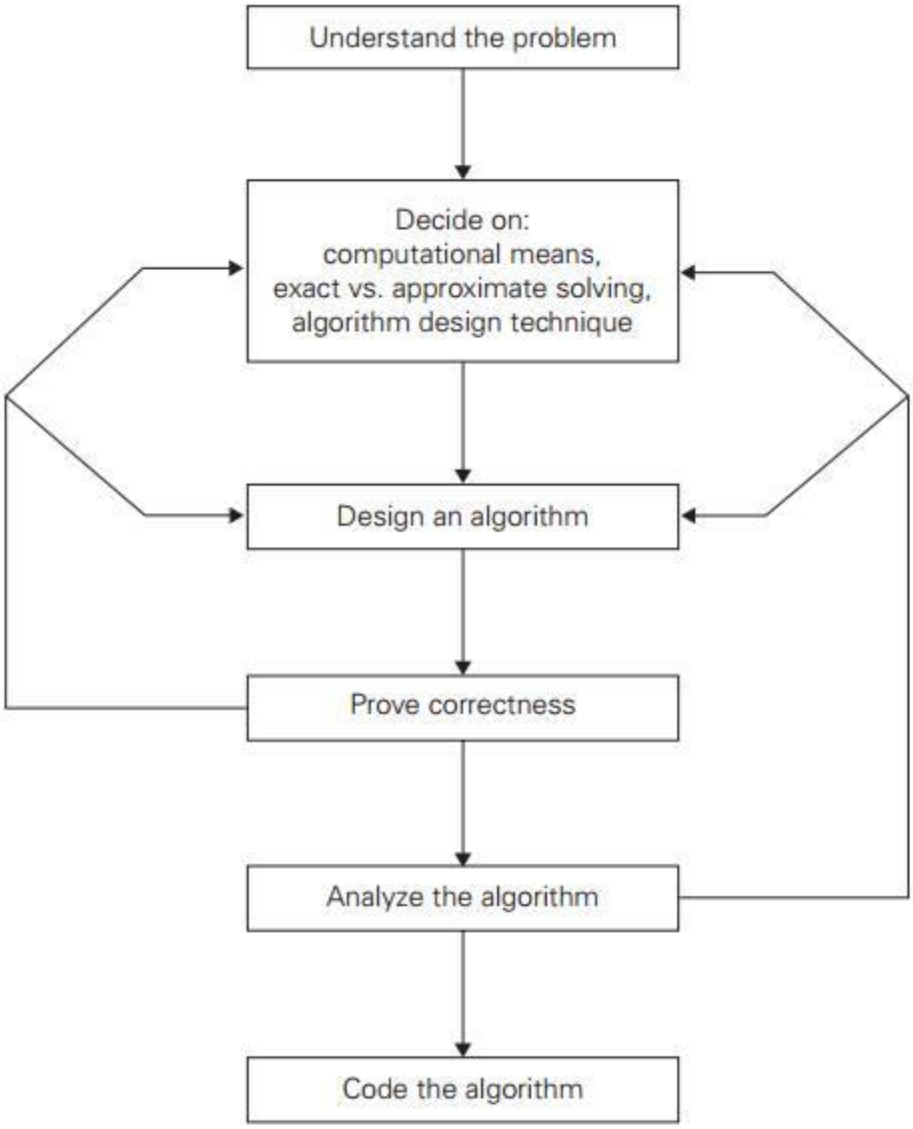


FIGURE 1.2 Algorithm design and analysis process.

)

Video Content / Details of website for further learning (if any):

https://www.brainkart.com/article/Fundamentals of problem solving-8013/www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L3

IT

IV/II

Course Name with Code **:19ITC11/Design and Analysis of Algorithm**

Course Teacher **:Mr.T.Manivel**

Unit **:I-Introduction** **Date of Lecture:**

Topic of Lecture: Important Problem Types

Introduction :

- Sorting.
- Searching.
- String processing (e.g. string matching)
- Graph problems (e.g. graph coloring problem)
- Combinatorial problems (e.g. maximizes a cost)
- Geometric problems (e.g. convex hull problem)
- Numerical problems (e.g. solving equations)

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Sorting

The sorting problem is to rearrange the items of a given list in nondecreasing order.

Searching

The searching problem deals with finding a given value, called a search key, in a given set

String Processing

Dealing with nonnumer-ical data has intensified the interest of researchers and computing practitioners in string-handling algorithms

Graph Problems

Informally, a graph can be thought of as a collection of points called vertices, some of which are connected by line segments called edges.

Combinatorial Problems

The traveling salesman problem and the graph-coloring problem are examples of combinatorial problems.

Geometric Problems

Geometric algorithms deal with geometric objects such as points, lines, and poly-gons.

Numerical Problems

Numerical problems, another large special area of applications, are problems that involve mathematical objects of continuous nature

Video Content / Details of website for further learning (if any):
https://www.brainkart.com/article/Important-Problem-Types-in-Algorithms-Analysis_8000/www.khanacademy.org/computing/computer-science/algorithm
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L4

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :I-Introduction **Date of Lecture:**

Topic of Lecture: Fundamentals of the Analysis of Algorithm Efficiency

Introduction :

- Time efficiency - indicates how fast an algorithm in question runs.
- Space efficiency - deals with the extra space the algorithm requires.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Fundamentals of Analysis of Algorithm:

1 Analysis of Framework

2 Measuring an input size

3 Units for measuring runtime

4 Worst case, Best case and Average case

5 Asymptotic Notations

Analysis Frame Work

- Time efficiency - indicates how fast an algorithm in question runs.
- Space efficiency - deals with the extra space the algorithm requires.

Measuring an Input Size

An algorithm's efficiency as a function of some parameter n indicating the algorithm's input size.

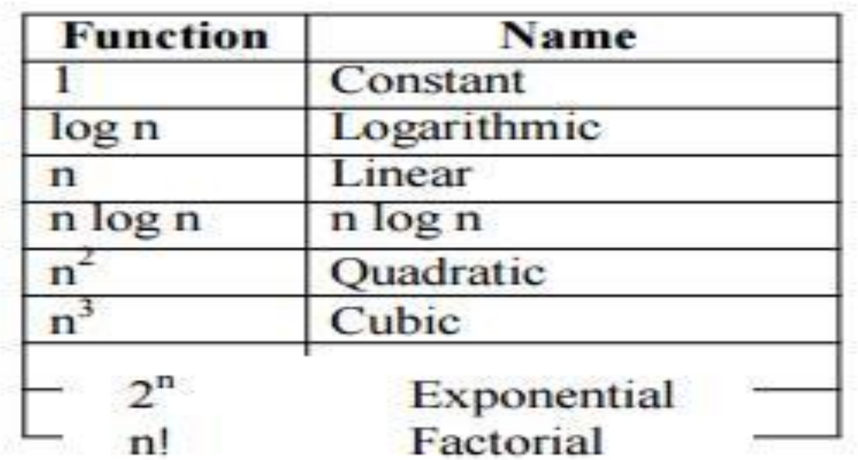
Units for measuring run time:

We can simply use some standard unit of time measurement-a second, a millisecond, and so on-to measure the running time of a program implementing the algorithm.

Worst case, Best case and Average case Efficiences

It is reasonable to measure an algorithm's efficiency as a function of a parameter indicating the size of the algorithm's input.

Asymptotic Notations



Function	Name
1	Constant
log n	Logarithmic
n	Linear
n log n	n log n
n^2	Quadratic
n^3	Cubic
2^n	Exponential
n!	Factorial

Video Content / Details of website for further learning (if any):

https://www.brainkart.com/article/Fundamentals-of-the-Analysis-of-Algorithm-Efficiency_7965/www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

L5

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :I-Introduction

Date of Lecture:

Topic of Lecture: Analysis Framework

Introduction :

- Time efficiency, also called time complexity, indicates how fast an algorithm in question runs.
- Space efficiency, also called space complexity, refers to the amount of memory units required by the algorithm in ad-dition to the space needed for its input and output.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

The Analysis Framework

1. Measuring an Input's Size
2. Units for Measuring Running Time
3. Orders of Growth
4. Worst-Case, Best-Case, and Average-Case Efficiencies
5. Recapitulation of the Analysis Framework

Time efficiency, also called time complexity, indicates how fast an algorithm in question runs.

Space efficiency, also called space complexity, refers to the amount of memory units required by the algorithm in ad-dition to the space needed for its input and output.

Measuring an Input's Size

All algorithms run longer on larger inputs. For example, it takes longer to sort larger arrays, multiply larger matrices, and so on

Units for Measuring Running Time

Established framework for the analysis of an algorithm's time ef-ficiency suggests measuring it by counting the number of times the algorithm's basic operation is executed on inputs of size ***n***

Orders of Growth

- Omagha –notation
- 0-notation
- –notation

Recapitulation of the Analysis Framework

- Both time and space efficiencies are measured as functions of the algorithm's input size.
- Time efficiency is measured by counting the number of times the algorithm's basic operation is executed. Space efficiency is measured by counting the number of extra memory units consumed by the algorithm.

Video Content / Details of website for further learning (if any):

http://www.brainkart.com/article/The-Analysis-Framework_8002/www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L6

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :I-Introduction **Date of Lecture:**

Topic of Lecture: Asymptotic Notations

Introduction :

- Time efficiency, also called time complexity, indicates how fast an algorithm in question runs. Space efficiency, also called space complexity, refers to the amount of memory units required by the algorithm in ad-dition to the space needed for its input and output.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure4.
Mathematical knowledge

Detailed content of the Lecture:

***1.O*–notation**

Definition : A function $t\ (n)$ is said to be in $O(g(n))$, denoted $t\ (n) \in O(g(n))$, if $t\ (n)$ is bounded above by some constant multiple of $g(n)$ for all large n, i.e., if there exist some positive constant c and some nonnegative integer n_0 such that

$t\ (n) \leq cg(n)$ for all $n \geq n_0$.

2. Omagha –notation (Ω-notation)

Definition: A function t (n) is said to be in (g(n)), denoted t (n) ∈ (g(n)), if t (n) is bounded below by some positive constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some nonnegative integer n_0 such that

$\boldsymbol{t\ (n) \geq cg(n)}$ for all $\boldsymbol{n \geq n_0}$.

3. 0-notation

Definition: A function t (n) is said to be in (g(n)), denoted t (n) ∈ (g(n)), if t (n) is bounded both above and below by some positive constant multiples of g(n) for all large n, i.e., if there exist some positive constants c_1 and c_2 and some nonnegative integer n_0 such that

$c_2 g(n) \leq t(n) \leq c_1 g(n)$ for all $n \geq n_0$.

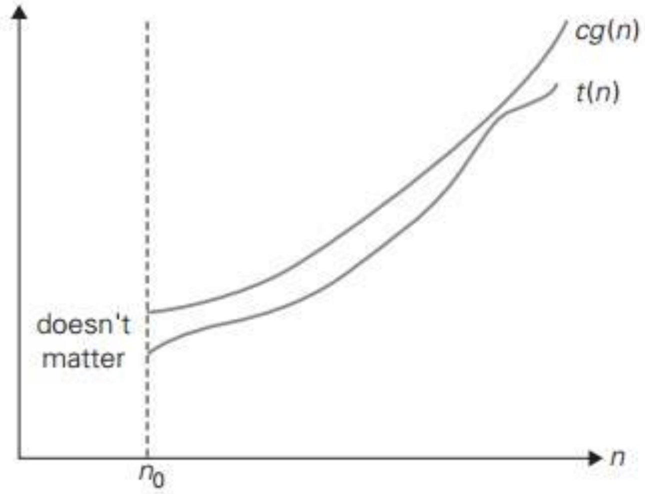


FIGURE 2.1 Big-oh notation: $t(n) \in O(g(n))$.

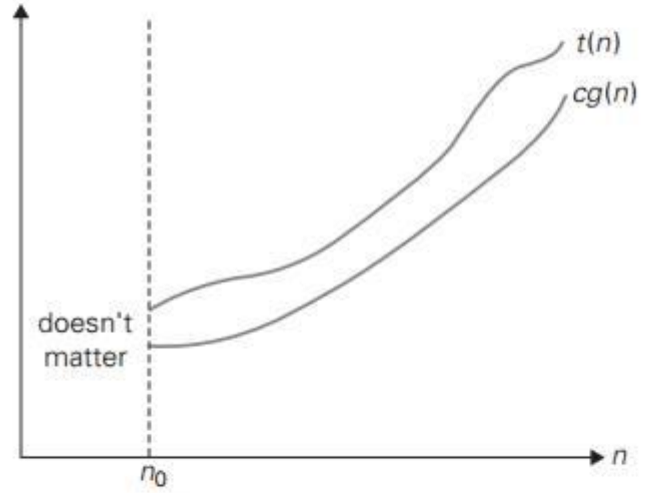


FIGURE 2.2 Big-omega notation: $t(n) \in \Omega(g(n))$.

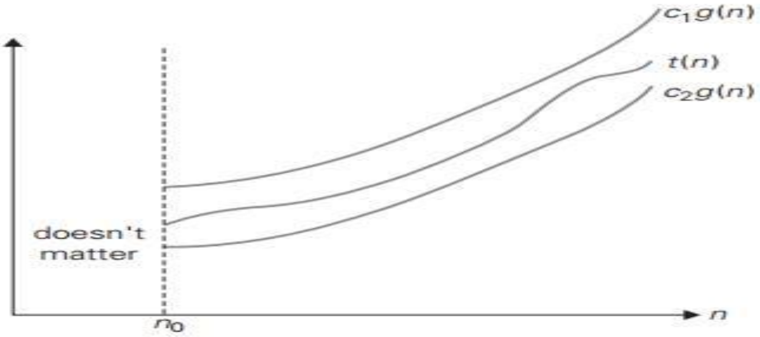


FIGURE 2.3 Big-theta notation: $t(n) \in \Theta(g(n))$.

Θ-notation

Video Content / Details of website for further learning (if any):

http://www.brainkart.com/article/Asymptotic-Notations-and-Basic-Efficiency-Classes_8003/

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L7

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :I-Introduction **Date of Lecture:**

Topic of Lecture: Asymptotic Properties

Introduction :

- Time efficiency, also called time complexity, indicates how fast an algorithm in question runs. Space efficiency, also called space complexity, refers to the amount of memory units required by the algorithm in ad-dition to the space needed for its input and output.

Prerequisite knowledge for Complete understanding and learning of Topic:

Basics of Algorithm
Programming Knowledge
Data Structure
Mathematical knowledge

Detailed content of the Lecture:

1.*O* –notation

Definition : A function *t (n)* is said to be in *O(g(n))*, denoted *t (n)* ∈ *O(g(n)),* if *t (n)* is bounded above by some constant multiple of *g(n)* for all large *n*, i.e., if there exist some positive constant *c* and some nonnegative integer n_0 such that

$t(n) \leq cg(n)$ for all $n \geq n_0$.

2. Omagha –notation (Ω-notation)

Definition: A function t (n) is said to be in (g(n)), denoted t (n) ∈ (g(n)), if t (n) is bounded below by some positive constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some nonnegative integer n_0 such that

$\boldsymbol{t(n) \geq cg(n)}$ for all $\boldsymbol{n \geq n_0}$.

3. 0-notation

Definition: A function t (n) is said to be in (g(n)), denoted t (n) ∈ (g(n)), if t (n) is bounded both above and below by some positive constant multiples of g(n) for all large n, i.e., if there exist some positive constants c_1 and c_2 and some nonnegative integer n_0 such that

$c_2 g(n) \leq t(n) \leq c_1 g(n)$ for all $n \geq n_0$.

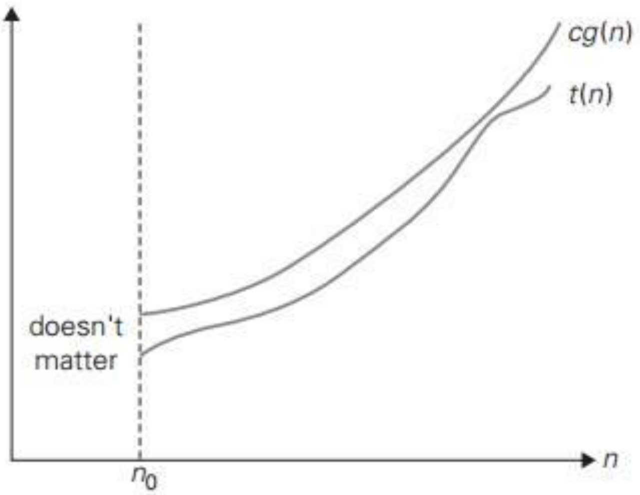


FIGURE 2.1 Big-oh notation: $t(n) \in O(g(n))$.

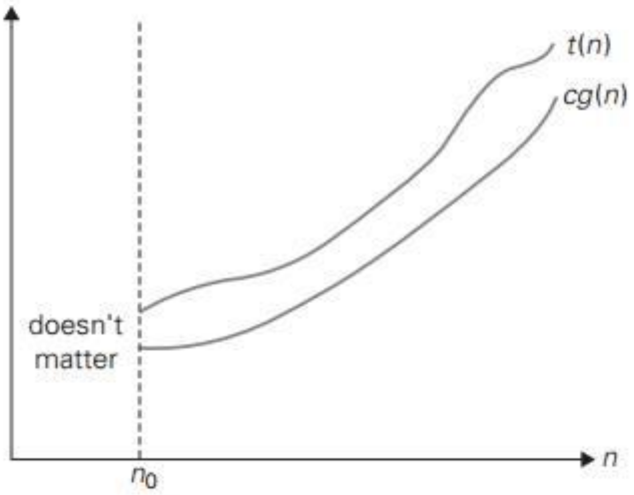


FIGURE 2.2 Big-omega notation: $t(n) \in \Omega(g(n))$.

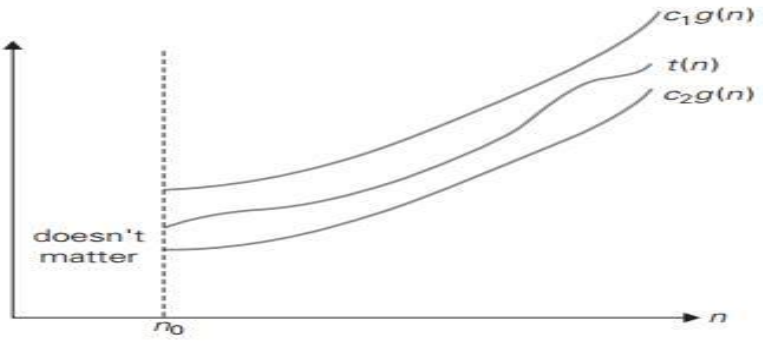


FIGURE 2.3 Big-theta notation: $t(n) \in \Theta(g(n))$.

Θ-notation

Video Content / Details of website for further learning (if any):

http://www.brainkart.com/article/Asymptotic-Notations-and-Basic-Efficiency-Classes_8003/

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L8

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :I-Introduction

Date of Lecture:

Topic of Lecture: Mathematical analysis for Recursive algorithms

Introduction :

- A recursive algorithm is an algorithm which calls itself

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure4.
Mathematical knowledge

Detailed content of the Lecture:

Decide on a parameter (or parameters) indicating an input’s size.

Identify the algorithm’s basic operation.

- Check whether the number of times the basic operation is executed can vary on different inputs of the same size;
- if it can, the worst-case, average-case, and best-case efficiencies must be investigated separately.
- Set up a recurrence relation, with an appropriate initial condition, for the number of times the basic operation is executed.
- Solve the recurrence or, at least, ascertain the order of growth of its solution.

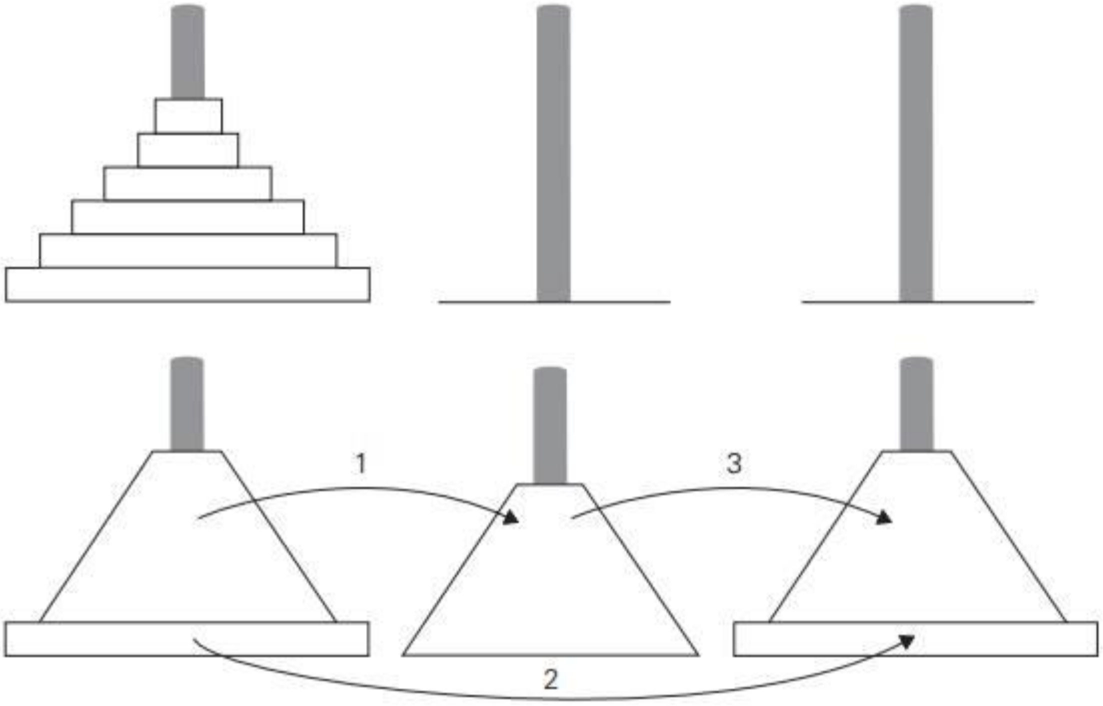


FIGURE 2.4 Recursive solution to the Tower of Hanoi puzzle.

Video Content / Details of website for further learning (if any):

http://www.brainkart.com/article/Mathematical-Analysis-of-Recursive-Algorithms_8005/www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L9

IT

IV/II

Course Name with Code :**19ITC11/Design and Analysis of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**I-Introduction** **Date of Lecture:**

Topic of Lecture: Mathematical analysis for Recursive algorithms

Introduction :

- A recursive algorithm is an algorithm which does not calls itself

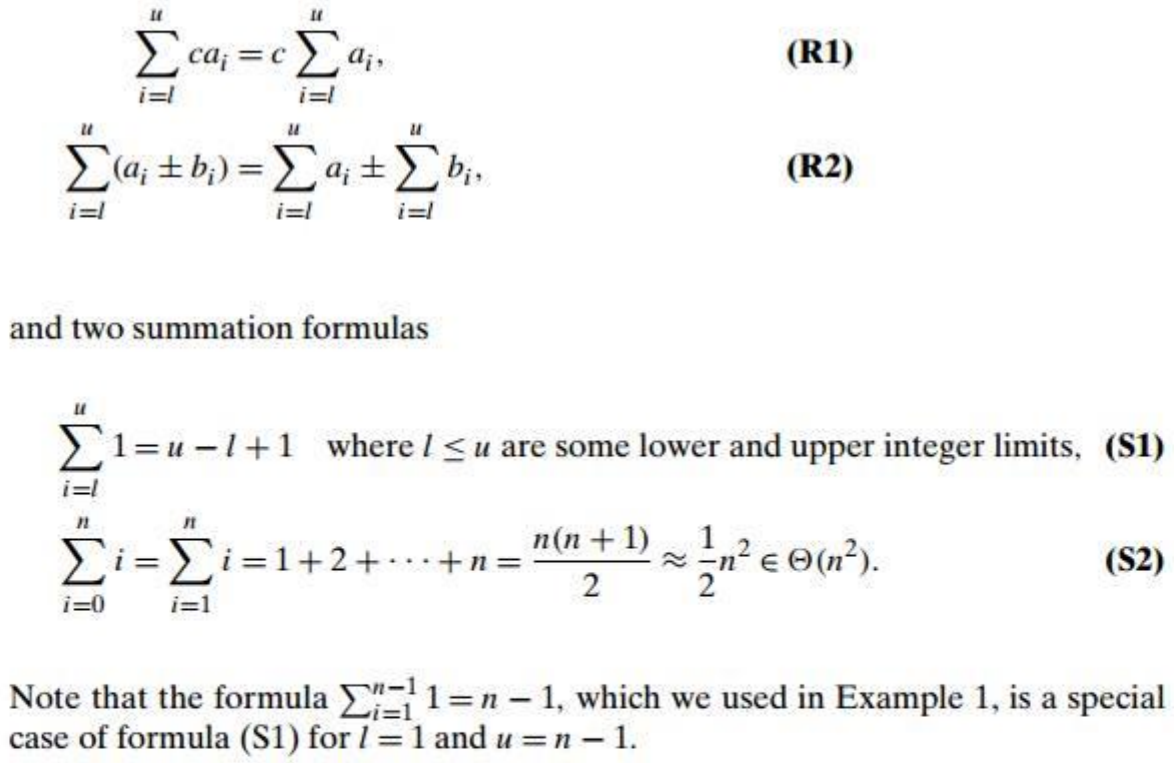
Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Decide on a parameter (or parameters) indicating an input's size.

- Identify the algorithm's basic operation. (As a rule, it is located in the inner-most loop.)
- Check whether the number of times the basic operation is executed depends only on the size of an input.
- If it also depends on some additional property, the worst-case, average-case, and, if necessary, best-case efficiencies have to be investigated separately.
- Set up a sum expressing the number of times the algorithm's basic operation is executed.
- Using standard formulas and rules of sum manipulation, either find a closed-form formula for the count or, at the very least, establish its order of growth.
- Before proceeding with further examples, you may want to review Appen-dix A, which contains a list of summation formulas and rules that are often useful in analysis of algorithms. In particular, we use especially frequently two basic rules of sum manipulation



$$\sum_{i=l}^{u} ca_i = c\sum_{i=l}^{u} a_i, \qquad \textbf{(R1)}$$

$$\sum_{i=l}^{u}(a_i \pm b_i) = \sum_{i=l}^{u} a_i \pm \sum_{i=l}^{u} b_i, \qquad \textbf{(R2)}$$

and two summation formulas

$$\sum_{i=l}^{u} 1 = u - l + 1 \quad \text{where } l \le u \text{ are some lower and upper integer limits,} \quad \textbf{(S1)}$$

$$\sum_{i=0}^{n} i = \sum_{i=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} \approx \frac{1}{2}n^2 \in \Theta(n^2). \qquad \textbf{(S2)}$$

Note that the formula $\sum_{i=1}^{n-1} 1 = n - 1$, which we used in Example 1, is a special case of formula (S1) for $l = 1$ and $u = n - 1$.

Video Content / Details of website for further learning (if any):

https://www.brainkart.com/article/Strassen---s-Matrix-Multiplication_8026/www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

1.

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

L10

LECTURE HANDOUTS

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit : II-Brute Force and Divide-and-Conquer **Date of Lecture:**

Topic of Lecture:Brute Force- Closest Pair and Convex Hull problem

Introduction :

- The closest-pair problem calls for finding the two closest points in a set of ***n*** points.
- It is the simplest of a variety of problems in computational geometry that deals with proximity of points in the plane or higher-dimensional spaces.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

```
Algorithm    BruteForceClosestPair(P )

//Finds distance between two closest points in the plane by brute force
//Input: A list P of n (n ≥ 2) points p1(x1, y1), . . . , pn(xn, yn) //Output: The distance between the closest pair of points

d ← ∞

for i ← 1 to n − 1 do

for j ← i + 1 to n do
d ← min(d, sqrt((xi − xj )^2 + (yi − yj )^2)) //sqrt is square root
return d
```

Convex-Hull Problem

A set of points (finite or infinite) in the plane is called ***convex*** if for any two points ***p*** and ***q*** in the set, the entire line segment with the endpoints at ***p*** and ***q*** belongs to the set.

Convex hull of a set ***S*** of points is the smallest convex set containing ***S***. (The "smallest" requirement means that the convex hull of ***S*** must be a subset of any convex set containing ***S***.)

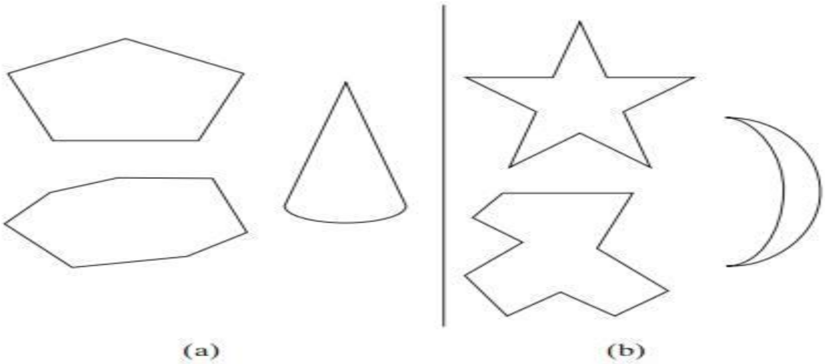


FIGURE 3.4 (a) Convex sets. (b) Sets that are not convex.

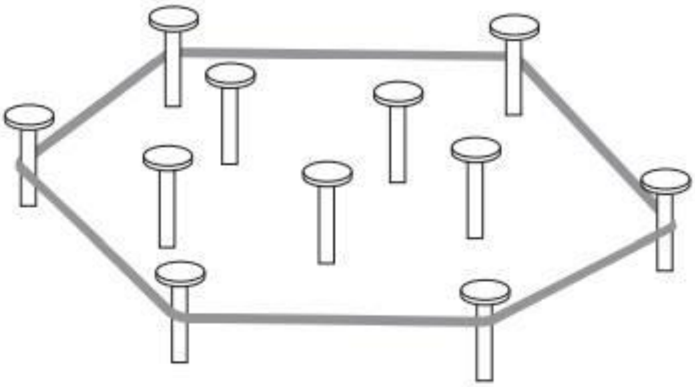


FIGURE 3.5 Rubber-band interpretation of the convex hull.

Video Content / Details of website for further learning (if any):
https://www.brainkart.com/article/Closest-Pair-and-Convex-Hull-Problems-by-Brute-Force_8012/
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L11

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit : II-Brute Force and Divide-and-Conquer **Date of Lecture:**

Topic of Lecture: Exhaustive Search- Traveling Salesman Problem

Introduction :

- Exhaustive search is simply a brute-force approach to combinatorial prob-lems.
- It suggests generating each and every element of the problem domain, se-lecting those of them that satisfy all the constraints, and then finding a desired element (e.g., the one that optimizes some objective function

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Exhaustive Search:

- Exhaustive search is simply a brute-force approach to combinatorial prob-lems.
- It suggests generating each and every element of the problem domain, se-lecting those of them that satisfy all the constraints, and then finding a desired element (e.g., the one that optimizes some objective function).

1. Traveling Salesman Problem

2. Knapsack Problem

3. Assignment Problem

Traveling Salesman Problem

A Hamiltonian circuit is defined as a cycle that passes through all the vertices of the graph exactly once.

Knapsack Problem

Given n items of known weights $w_1, w_2, \ldots, w_n$ and values $v_1, v_2, \ldots, v_n$ and a knapsack of capacity W, find the most valuable subset of the items that fit into the knapsack.

Assignment Problem

there are *n* people who need to be assigned to execute *n* jobs, one person per job. (That is, each person is assigned to exactly one job and each job is assigned to exactly one person.

Video Content / Details of website for further learning (if any):

https://www.brainkart.com/article/Exhaustive-Search_8013/www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS | **L12**

IT | **IV/II**

Course Name with Code :**19ITC11/Design and Analysis of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**II-Brute Force and Divide-and-Conquer** **Date of Lecture:**

Topic of Lecture: Kanpsack Problem, Assignment Problem

Introduction :

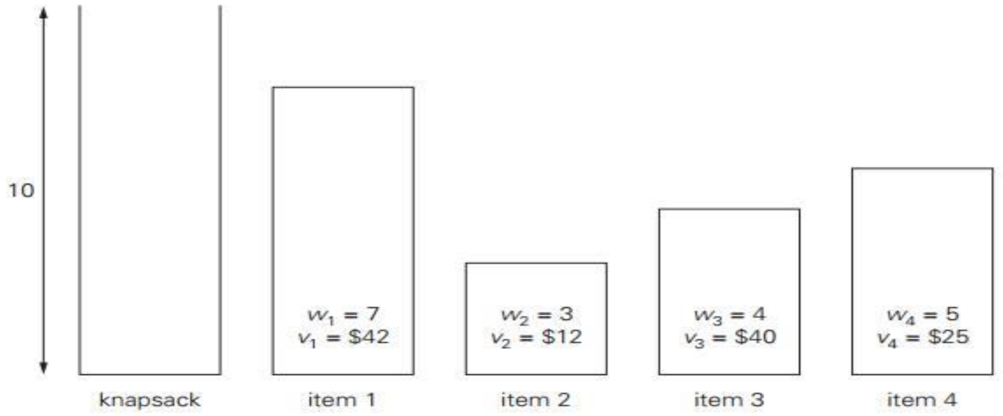
- Exhaustive search is simply a brute-force approach to combinatorial prob-lems

Prerequisite knowledge for Complete understanding and learning of Topic:

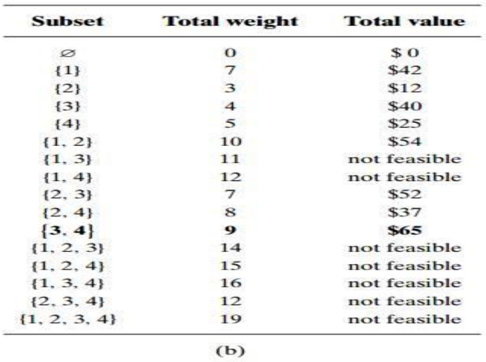
1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Kanpsack Problem



(a)



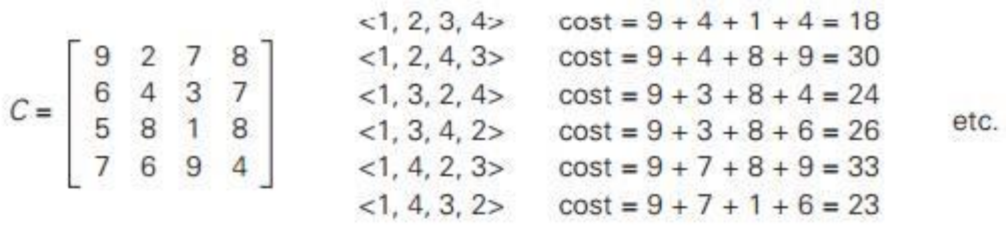
Subset	Total weight	Total value
∅	0	$ 0
{1}	7	$42
{2}	3	$12
{3}	4	$40
{4}	5	$25
{1, 2}	10	$54
{1, 3}	11	not feasible
{1, 4}	12	not feasible
{2, 3}	7	$52
{2, 4}	8	$37
{3, 4}	**9**	**$65**
{1, 2, 3}	14	not feasible
{1, 2, 4}	15	not feasible
{1, 3, 4}	16	not feasible
{2, 3, 4}	12	not feasible
{1, 2, 3, 4}	19	not feasible

(b)

FIGURE 3.8 (a) Instance of the knapsack problem. (b) Its solution by exhaustive search. The information about the optimal selection is in bold.

Assignment Problem

there are *n* people who need to be assigned to execute *n* jobs, one person per job. (That is, each person is assigned to exactly one job and each job is assigned to exactly one person.)



$$C = \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix}$$

<1, 2, 3, 4> cost = 9 + 4 + 1 + 4 = 18

<1, 2, 4, 3> cost = 9 + 4 + 8 + 9 = 30

<1, 3, 2, 4> cost = 9 + 3 + 8 + 4 = 24

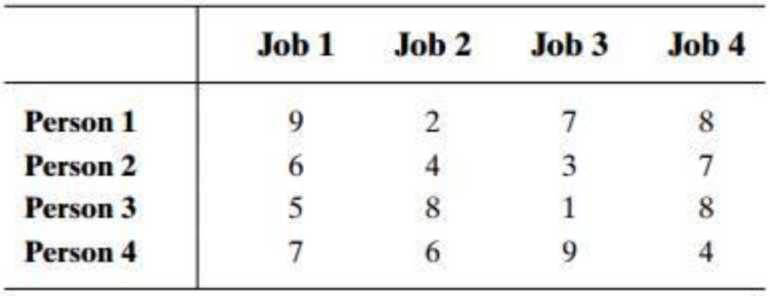
<1, 3, 4, 2> cost = 9 + 3 + 8 + 6 = 26

<1, 4, 2, 3> cost = 9 + 7 + 8 + 9 = 33

<1, 4, 3, 2> cost = 9 + 7 + 1 + 6 = 23

etc.

FIGURE 3.9 First few iterations of solving a small instance of the assignment problem by exhaustive search.



	Job 1	Job 2	Job 3	Job 4
Person 1	9	2	7	8
Person 2	6	4	3	7
Person 3	5	8	1	8
Person 4	7	6	9	4

Video Content / Details of website for further learning (if any):

https://www.brainkart.com/article/Exhaustive-Search_8013/www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L13

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :II-Brute Force and Divide-and-Conquer

Date of Lecture:

Topic of Lecture: Divide and conquer methodology

Introduction :

- Divide-and-conquer, breaks a problem into subproblems that are similar to the original problem, recursively solves the subproblems,

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

- The merging of two sorted arrays can be done as follows.
- Two pointers (array indices) are initialized to point to the first elements of the arrays being merged.
- Then the elements pointed to are compared and the smaller of them is added to a new array or list being constructed.
- Then the index of the smaller element is incremented to point to its immediate successor in the array.
- This operation is continued until one of the two given arrays is exhausted
- Then the remaining elements of the other array are copied to the end of the newarray.

Algorithm : Merge Sort B(0....p-1), C(0.... q-1), A(0....p + q -1)
// Merges two sorted arrays into one sorted array .

// Input: Array B (0....p-1) and C (0....q-1) both sorted

//Output: Sorted array A (0....p + q -1) of the elements of B and C

```
i $^{f}$ 0; j $^{f}$0 ;k$^{f}$ 0 while i<p and j<q do if B[i] ≤ C[j]
A[k] $^{f}$B[i];i $^{f}$i+1
else A[k] $^{f}$ C[j] j$^{f}$ j+1 k $^{f}$k+1
if i=p

copy C[j....q-1] to A[k....p+q-1] else copy B[i....p-1] to A[k....p+q-1]
```

Video Content / Details of website for further learning (if any):

https://www.brainkart.com/article/Divide-and-Conquer-Method_7968/www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L14

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :II-Brute Force and Divide-and-Conquer Date of Lecture:

Topic of Lecture: Merge sort

Introduction :

- Divide-and-conquer, breaks a problem into subproblems that are similar to the original problem, recursively solves the subproblems,

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

- The merging of two sorted arrays can be done as follows.
- Two pointers (array indices) are initialized to point to the first elements of the arrays being merged.
- Then the elements pointed to are compared and the smaller of them is added to a new array or list being constructed.
- Then the index of the smaller element is incremented to point to its immediate successor in the array.
- This operation is continued until one of the two given arrays is exhausted
- Then the remaining elements of the other array are copied to the end of the newarray.

Algorithm : Merge Sort B(0....p-1), C(0.... q-1), A(0....p + q -1)
// Merges two sorted arrays into one sorted array .

// Input: Array B (0....p-1) and C (0....q-1) both sorted

//Output: Sorted array A (0....p + q -1) of the elements of B and C

i ^{f} 0; j ^{f}0 ;k^{f} 0 while i<p and j<q do if B[i] ≤ C[j]
A[k] ^{f}B[i];i ^{f}i+1

```
else A[k] $^{f}$ C[j] j$^{f}$ j+1 k $^{f}$k+1
if i=p

copy C[j....q-1] to A[k....p+q-1] else copy B[i....p-1] to A[k....p+q-1]
```

Video Content / Details of website for further learning (if any):

https://www.brainkart.com/article/Divide-and-Conquer-Method_7968/www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

L15

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :MsMr.T.Manivel

Unit :II-Brute Force and Divide-and-Conquer

Date of Lecture:

Topic of Lecture: Quick sort

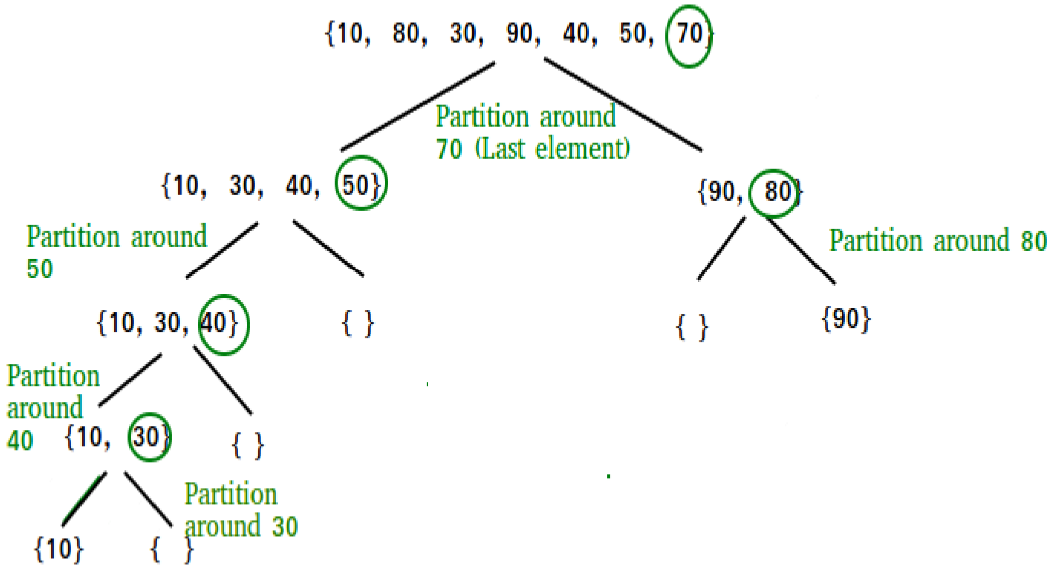
Introduction :

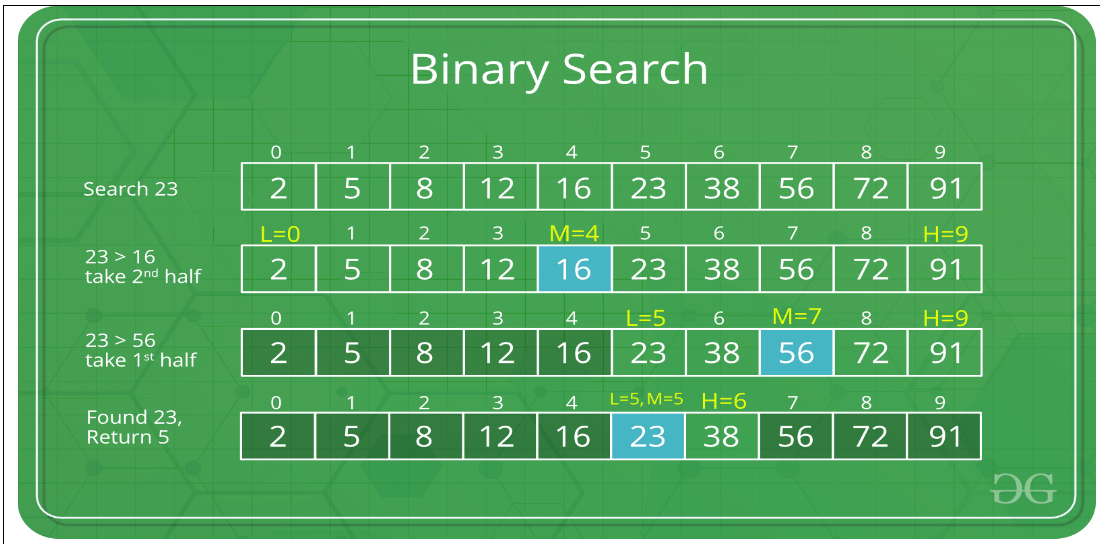
Divide-and-conquer, breaks a problem into subproblems that are similar to the original problem, recursively solves the subproblems,

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:





Video Content / Details of website for further learning (if any):

https://www.geeksforgeeks.org/quick-sort/

https://www.geeksforgeeks.org/binary-search/

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L16

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :MsMr.T.Manivel

Unit :II-Brute Force and Divide-and-Conquer

Date of Lecture:

Topic of Lecture: Binary search

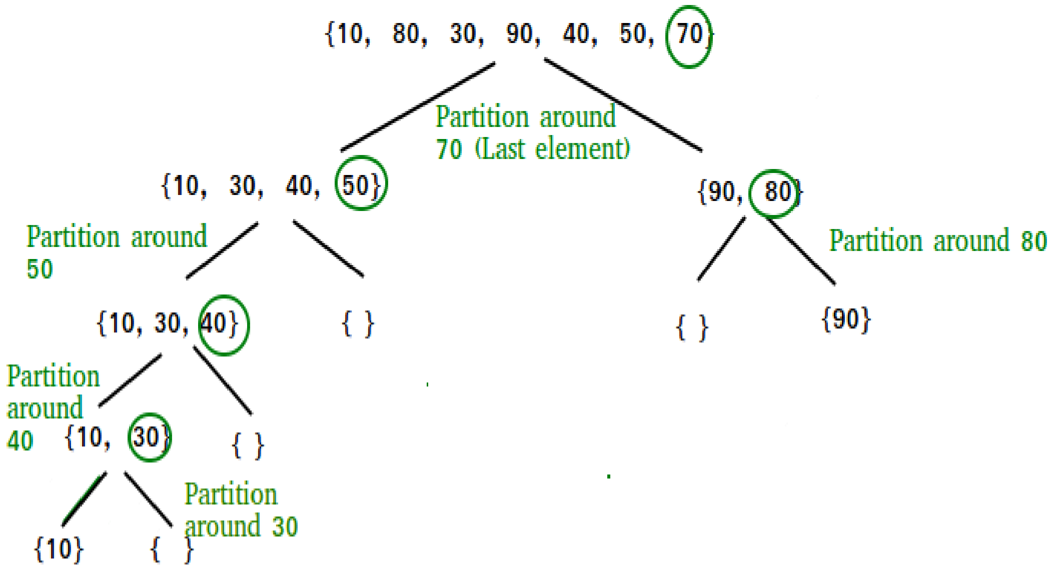
Introduction :

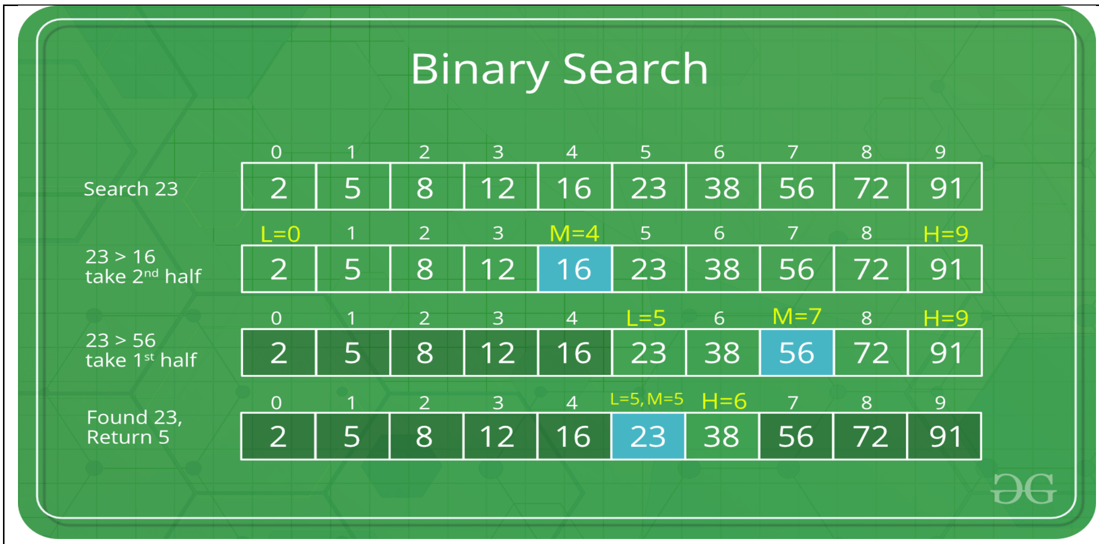
Divide-and-conquer, breaks a problem into subproblems that are similar to the original problem, recursively solves the subproblems,

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:





Binary Search

	0	1	2	3	4	5	6	7	8	9
Search 23	2	5	8	12	16	23	38	56	72	91
	L=0	1	2	3	M=4	5	6	7	8	H=9
23 > 16 take 2nd half	2	5	8	12	16	23	38	56	72	91
	0	1	2	3	4	L=5	6	M=7	8	H=9
23 > 56 take 1st half	2	5	8	12	16	23	38	56	72	91
	0	1	2	3	4	L=5, M=5	H=6	7	8	9
Found 23, Return 5	2	5	8	12	16	23	38	56	72	91

Video Content / Details of website for further learning (if any):

https://www.geeksforgeeks.org/quick-sort/

https://www.geeksforgeeks.org/binary-search/

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L17

IT

IV/II

Course Name with Code :19ITC11/Design and Analysis of Algorithm

Course Teacher :Mr.T.Manivel

Unit :II-Brute Force and Divide-and-Conquer **Date of Lecture:**

Topic of Lecture: Multiplication of Largest Integer, Strassens Matrix Multiplication

Introduction :

- Some applications, notably modern cryptography, require manipulation of inte-gers that are over 100 decimal digits long.
- Since such integers are too long to fit in a single word of a modern computer, they require special treatment.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

$= 2 \cdot 10^1 + 3 \cdot 10^0$ and $14 = 1 \cdot 10^1 + 4 \cdot 10^0$.

Now let us multiply them:

$23 * 14 = (2 \cdot 10^1 + 3 \cdot 10^0) * (1 \cdot 10^1 + 4 \cdot 10^0)$

$= (2 * 1)10^2 + (2 * 4 + 3 * 1)10^1 + (3 * 4)10^0$.

Of course, there is nothing special about the numbers we just multiplied. For any pair of two-digit numbers $a = a_1a_0$ and $b = b_1b_0$, their product c can be computed by the formula

$c = a * b = c_2 10^2 + c_1 10^1 + c_0,$

where

$c_2 = a_1 * b_1$ is the product of their first digits,

$c_0 = a_0 * b_0$ is the product of their second digits,

$c_1 = (a_1 + a_0) * (b_1 + b_0) - (c_2 + c_0)$ is the product of the sum of the a's digits and the sum of the b's digits minus the sum of c_2 and c_0.
$c_2 10^n + c_1 10^{n/2} + c_0,$

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/quick-sort/
https://www.geeksforgeeks.org/binary-search/
www.khanacademy.org/computing/computer-science/algorithm
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS **L18**

IT **IV/II**

Course Name with Code **:19ITC11/Design and Analysis of Algorithm**

Course Teacher **:Mr.T.Manivel**

Unit **:II-Brute Force and Divide-and-Conquer** **Date of Lecture:**

Topic of Lecture: Closest Pair and Convex Hull problem

Introduction :

- The closest-pair problem calls for finding the two closest points in a set of ***n*** points
- Finding the convex hull for a given set of points in the plane or a higher dimensional space is one of the most important
- most important—problems in com-putational geometry.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Convex

- The closest-pair problem calls for finding the two closest points in a set of ***n*** points.
- It is the simplest of a variety of problems in computational geometry that deals with proximity of points in the plane or higher-dimensional spaces.

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

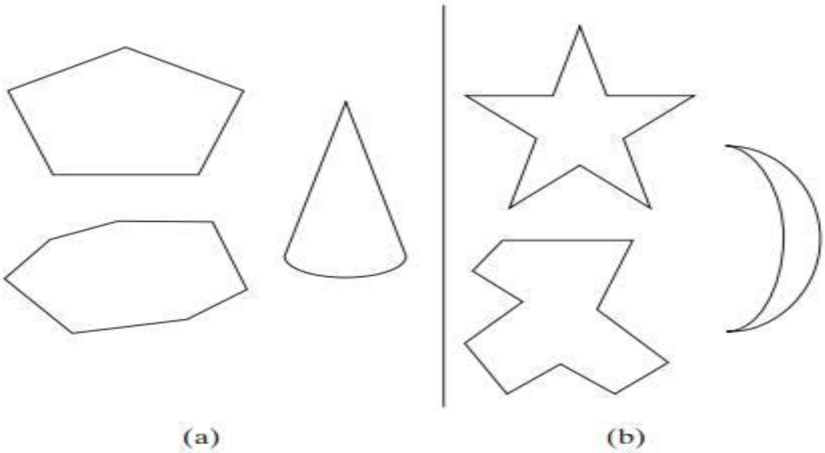


FIGURE 3.4 (a) Convex sets. (b) Sets that are not convex.

Convex hull

- Finding the convex hull for a given set of points in the plane or a higher dimensional space is one of the most important problems in com-putational geometry

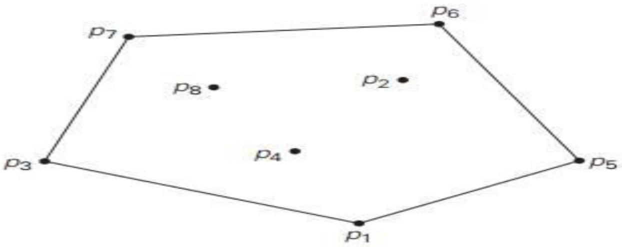


FIGURE 3.6 The convex hull for this set of eight points is the convex polygon with vertices at p_1, p_5, p_6, p_7, and p_3.

Video Content / Details of website for further learning (if any):
https://www.brainkart.com/article/Strassen---s-Matrix-Multiplication_8026/www.khanacademy.org/computing/computer-science/algorithm
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

L19

LECTURE HANDOUTS

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :III- Dynamic Programming And Greedy Technique

Date of Lecture:

Topic of Lecture:Computing a Binomial Coefficient

Introduction :

- Dynamic Programming was invented by Richard Bellman, 1950. It is a very general technique for solving optimization problems. Using Dynamic Programming requires that the problem can be divided into overlapping similar sub-problems. A recursive relation between the larger and smaller sub problems is used to fill out a table.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Computing binomial coefficients is non optimization problem but can be solved using dynamic programming.

Binomial coefficients are represented by $C(n, k)$ or $({}^{n}{}_{k})$ and can be used to represent the coefficients of a binomail:

$$(a + b)^n = C(n, 0)a^n + ... + C(n, k)a^{n-k}b^k + ... + C(n, n)b^n$$

The recursive relation is defined by the prior power

$$C(n, k) = C(n-1, k-1) + C(n-1, k) \text{ for } n > k > 0$$

$$\text{IC } C(n, 0) = C(n, n) = 1$$

Dynamic algorithm constructs a nxk table, with the first column and diagonal filled out using the IC.

Construct the table:

				k			
		0	1	2	...	k-1	k
	0	1					
	1	1	1				
	2	1	2	1			
n	.						
	.						
	.						
	k	1					1
	.						
	.						
	.						
	n-1	1				$C(n\text{-}1, k\text{-}1)$	
	n	1					$C(n, k)$

The table is then filled out iteratively, row by row using the recursive relation.

```
Algorithm Binomial(n, k)
    for i ← 0 to n do  // fill out the table row wise
        for i = 0 to min(i, k) do
            if j==0 or j==i then C[i, j] ← 1  // IC
            else C[i, j] ← C[i-1, j-1] + C[i-1, j]  // recursive relation
    return C[n, k]
```

The cost of the algorithm is filing out the table. Addition is the basic operation. Because $k \leq n$, the sum needs to be split into two parts because only the half the table needs to be filled out for $i < k$ and remaining part of the table is filled out across the entire row.

$A(n, k)$ = sum for upper triangle + sum for the lower rectangle

$= \sum_{i=1}^{k} \sum_{j=1}^{i-1} 1 + \sum_{i=1}^{n} \sum_{j=1}^{k} 1$

$= \sum_{i=1}^{k} (i-1) + \sum_{i=1}^{n} k$

$= (k-1)k/2 + k(n-k) \; \varepsilon \; \Theta(nk)$

Video Content / Details of website for further learning (if any):

http://www.csl.mtu.edu/cs4321/www/Lectures/Lecture%2015%20-%20Dynamic%20Programming%20Binomial%20Coefficients.htm

https://www.geeksforgeeks.org/binomial-coefficient-dp-9/

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L20

IT

II/IV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**III- Dynamic Programming And Greedy Technique**

Date of Lecture:

Topic of Lecture: Warshall's algorithm

Introduction :

- Warshall's algorithm for computing the transitive closure of a directed graph and Floyd's algorithm for the all-pairs shortest-paths problem. These algorithms are based on essentially the same idea: exploit a relationship between a problem and its simpler rather than smaller version. Warshall and Floyd published their algorithms without mentioning dynamic programming.

Prerequisite knowledge for Complete understanding and learning of Topic:

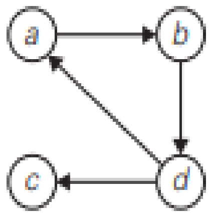
1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Warshall'salgorithm

The transitive closure of a directed graph with n vertices can be defined as the n × n boolean matrix T = {tij }, in which the element in the ith row and the j th column is 1 if there exists a nontrivial path (i.e., directed path of a positive length) from the ith vertex to the j th vertex; otherwise, tij is 0.

Example:



$$R^{(0)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 0 & 1 & 0 \end{array}$$

1's reflect the existence of paths with no intermediate vertices ($R^{(0)}$ is just the adjacency matrix); boxed row and column are used for getting $R^{(1)}$.

$$R^{(1)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & \mathbf{1} & 1 & 0 \end{array}$$

1's reflect the existence of paths with intermediate vertices numbered not higher than 1, i.e., just vertex *a* (note a new path from *d* to *b*); boxed row and column are used for getting $R^{(2)}$.

$$R^{(2)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & \mathbf{1} \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & \mathbf{1} \end{array}$$

1's reflect the existence of paths with intermediate vertices numbered not higher than 2, i.e., *a* and *b* (note two new paths); boxed row and column are used for getting $R^{(3)}$.

$$R^{(3)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array}$$

1's reflect the existence of paths with intermediate vertices numbered not higher than 3, i.e., *a*, *b*, and *c* (no new paths); boxed row and column are used for getting $R^{(4)}$.

$$R^{(4)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & \mathbf{1} & 1 & \mathbf{1} & 1 \\ b & \mathbf{1} & \mathbf{1} & \mathbf{1} & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array}$$

1's reflect the existence of paths with intermediate vertices numbered not higher than 4, i.e., *a*, *b*, *c*, and *d* (note five new paths).

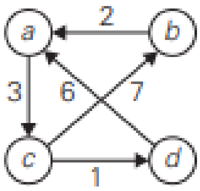
Floydy's algorithm:

We can generate the distance matrix with an algorithm that is very similar to Warshall's algorithm. It is called Floyd's algorithm after its co-inventor Robert W. Floyd. 1 It is applicable to both undirected and directed weighted graphs provided.

ALGORITHM Floyd(W[1..n, 1..n])

```
//Implements Floyd's algorithm for the all-pairs shortest-paths problem
//Input: The weight matrix W of a graph with no negative-length cycle
//Output: The distance matrix of the shortest paths' lengths
D ← W //is not necessary if W can be overwritten
for k ← 1 to n do
for i ← 1 to n do
for j ← 1 to n do
D[i, j ] ← min{D[i, j ], D[i, k] + D[k, j ]}
return D
```

EXAMPLE:



$$D^{(0)} = \begin{array}{c|cccc|} & a & b & c & d \\ a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \infty & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \infty & 0 \end{array}$$

Lengths of the shortest paths with no intermediate vertices ($D^{(0)}$ is simply the weight matrix).

$$D^{(1)} = \begin{array}{c|cccc|} & a & b & c & d \\ a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \mathbf{5} & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \mathbf{9} & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 1, i.e., just *a* (note two new shortest paths from *b* to *c* and from *d* to *c*).

$$D^{(2)} = \begin{array}{c|cccc|} & a & b & c & d \\ a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & 5 & \infty \\ c & \mathbf{9} & 7 & 0 & 1 \\ d & 6 & \infty & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 2, i.e., a and *b* (note a new shortest path from *c* to *a*).

$$D^{(3)} = \begin{array}{c|cccc|} & a & b & c & d \\ a & 0 & \mathbf{10} & 3 & \mathbf{4} \\ b & 2 & 0 & 5 & \mathbf{6} \\ c & 9 & 7 & 0 & 1 \\ d & 6 & \mathbf{16} & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 3, i.e., a, *b*, and *c* (note four new shortest paths from *a* to *b*, from a to *d*, from *b* to *d*, and from *d* to *b*).

$$D^{(4)} = \begin{array}{c|cccc|} & a & b & c & d \\ a & 0 & 10 & 3 & 4 \\ b & 2 & 0 & 5 & 6 \\ c & \mathbf{7} & 7 & 0 & 1 \\ d & 6 & 16 & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 4, i.e., a, *b*, *c*, and *d* (note a new shortest path from *c* to *a*).

Video Content / Details of website for further learning (if any):

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

https://www.gatevidyalay.com/floyd-warshall-algorithm-shortest-path-algorithm/

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS **L21**

IT **II/IIV**

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :III- Dynamic Programming And Greedy Technique

Date of Lecture:

Topic of Lecture: Floyd' algorithm

Introduction :

- Warshall's algorithm for computing the transitive closure of a directed graph and Floyd's algorithm for the all-pairs shortest-paths problem. These algorithms are based on essentially the same idea: exploit a relationship between a problem and its simpler rather than smaller version. Warshall and Floyd published their algorithms without mentioning dynamic programming.

Prerequisite knowledge for Complete understanding and learning of Topic:

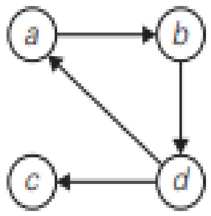
1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Warshall'salgorithm

The transitive closure of a directed graph with n vertices can be defined as the n × n boolean matrix T = {tij }, in which the element in the ith row and the j th column is 1 if there exists a nontrivial path (i.e., directed path of a positive length) from the ith vertex to the j th vertex; otherwise, tij is 0.

Example:



$$R^{(0)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 0 & 1 & 0 \end{array}$$

1's reflect the existence of paths with no intermediate vertices ($R^{(0)}$ is just the adjacency matrix); boxed row and column are used for getting $R^{(1)}$.

$$R^{(1)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 0 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & \mathbf{1} & 1 & 0 \end{array}$$

1's reflect the existence of paths with intermediate vertices numbered not higher than 1, i.e., just vertex *a* (note a new path from *d* to *b*); boxed row and column are used for getting $R^{(2)}$.

$$R^{(2)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & \mathbf{1} \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & \mathbf{1} \end{array}$$

1's reflect the existence of paths with intermediate vertices numbered not higher than 2, i.e., *a* and *b* (note two new paths); boxed row and column are used for getting $R^{(3)}$.

$$R^{(3)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 1 & 0 & 1 \\ b & 0 & 0 & 0 & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array}$$

1's reflect the existence of paths with intermediate vertices numbered not higher than 3, i.e., *a*, *b*, and *c* (no new paths); boxed row and column are used for getting $R^{(4)}$.

$$R^{(4)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & \mathbf{1} & 1 & \mathbf{1} & 1 \\ b & \mathbf{1} & \mathbf{1} & \mathbf{1} & 1 \\ c & 0 & 0 & 0 & 0 \\ d & 1 & 1 & 1 & 1 \end{array}$$

1's reflect the existence of paths with intermediate vertices numbered not higher than 4, i.e., *a*, *b*, *c*, and *d* (note five new paths).

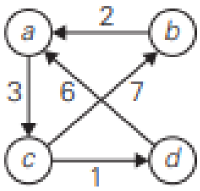
Floydy's algorithm:

We can generate the distance matrix with an algorithm that is very similar to Warshall's algorithm. It is called Floyd's algorithm after its co-inventor Robert W. Floyd. 1 It is applicable to both undirected and directed weighted graphs provided.

ALGORITHM Floyd(W[1..n, 1..n])

```
//Implements Floyd's algorithm for the all-pairs shortest-paths problem
//Input: The weight matrix W of a graph with no negative-length cycle
//Output: The distance matrix of the shortest paths' lengths
D ← W //is not necessary if W can be overwritten
for k ← 1 to n do
for i ← 1 to n do
for j ← 1 to n do
D[i, j ] ← min{D[i, j ], D[i, k] + D[k, j ]}
return D
```

EXAMPLE:



$$D^{(0)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \infty & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \infty & 0 \end{array}$$

Lengths of the shortest paths with no intermediate vertices ($D^{(0)}$ is simply the weight matrix).

$$D^{(1)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & \mathbf{5} & \infty \\ c & \infty & 7 & 0 & 1 \\ d & 6 & \infty & \mathbf{9} & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 1, i.e., just *a* (note two new shortest paths from *b* to *c* and from *d* to *c*).

$$D^{(2)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \infty & 3 & \infty \\ b & 2 & 0 & 5 & \infty \\ c & \mathbf{9} & 7 & 0 & 1 \\ d & 6 & \infty & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 2, i.e., a and *b* (note a new shortest path from *c* to *a*).

$$D^{(3)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & \mathbf{10} & 3 & \mathbf{4} \\ b & 2 & 0 & 5 & \mathbf{6} \\ c & 9 & 7 & 0 & 1 \\ d & 6 & \mathbf{16} & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 3, i.e., a, *b*, and *c* (note four new shortest paths from *a* to *b*, from a to *d*, from *b* to *d*, and from *d* to *b*).

$$D^{(4)} = \begin{array}{c|cccc} & a & b & c & d \\ \hline a & 0 & 10 & 3 & 4 \\ b & 2 & 0 & 5 & 6 \\ c & \mathbf{7} & 7 & 0 & 1 \\ d & 6 & 16 & 9 & 0 \end{array}$$

Lengths of the shortest paths with intermediate vertices numbered not higher than 4, i.e., a, *b*, *c*, and *d* (note a new shortest path from *c* to *a*).

Video Content / Details of website for further learning (if any):

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

https://www.gatevidyalay.com/floyd-warshall-algorithm-shortest-path-algorithm/

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

L22

IT

II/IIV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :III- Dynamic Programming And Greedy Technique

Date of Lecture:

Topic of Lecture: Optimal Binary Search Trees

Introduction :

- A binary search tree is one of the most important data structures in computer science. One of its principal applications is to implement a dictionary, a set of elements with the operations of searching, insertion, and deletion.

Prerequisite knowledge for Complete understanding and learning of Topic:

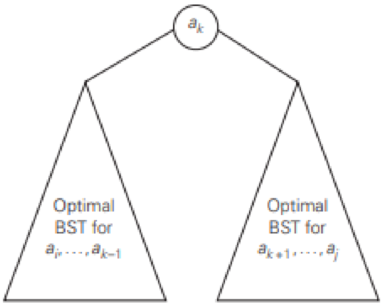
1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Algorithm:

```
//Finds an optimal binary search tree by dynamic programming
//Input: An array P[1..n] of search probabilities for a sorted list of n keys
//Output: Average number of comparisons in successful searches in the
// optimal BST and table R of subtrees’ roots in the optimal BST
    for i ← 1 to n do
        C[i, i − 1]← 0
        C[i, i]← P[i]
        R[i, i]← i
    C[n + 1, n]← 0
    for d ← 1 to n − 1 do //diagonal count
        for i ← 1 to n − d do
            j ← i + d
            minval ← ∞
            for k ← i to j do
                if C[i, k − 1] + C[k + 1, j ] < minval
```

```
minval ← C[i, k − 1] + C[k + 1, j ]; kmin ← k
R[i, j ] ← kmin
sum ← P[i]; for s ← i + 1 to j do sum ← sum + P[s]
C[i, j ]← minval + sum
return C[1, n], R
```


a_k
Optimal BST for $a_i, \dots, a_{k-1}$
Optimal BST for $a_{k+1}, \dots, a_j$

Video Content / Details of website for further learning (if any):

https://www.geeksforgeeks.org/optimal-binary-search-tree-dp-24/

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

L23

IT

II/IIV

Course Name with Code **:19ITC11/Design And Analysis Of Algorithm**

Course Teacher **:Mr.T.Manivel**

Unit **:III- Dynamic Programming And Greedy Technique**

Date of Lecture:

Topic of Lecture: Knapsack Problem and Memory functions

Introduction:

- To design a dynamic programming algorithm, we need to derive a recurrence relation that expresses a solution to an instance of the knapsack problem in terms of solutions to its smaller sub instances.
- It is natural to try to combine the strengths of the top-down and bottom-up approaches. The goal is to get a method that solves only sub problems that are necessary and does so only once. Such a method exists; it is based on using memory functions

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm

2.Programming Knowledge

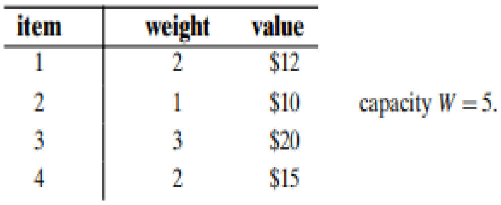
3.Data Structure

4.Mathematical knowledge

Detailed content of the Lecture:

Knapsack Problem:

Let us consider the instance given by the following data:



item	weight	value
1	2	\$12
2	1	\$10
3	3	\$20
4	2	\$15

capacity $W = 5$.

The dynamic programming table, filled by applying formulas (8.6) and (8.7), is shown in Figure 8.5. Thus, the maximal value is F (4, 5) = \$37. We can find the composition of an optimal subset by backtracing the computations of this entry in the table. Since F (4, 5)>F(3, 5), item 4 has to be included in an optimal solution along with an optimal subset for filling 5 − 2 = 3 remaining units of the knapsack capacity. The value of the latter is F (3, 3). Since F (3, 3) = F (2, 3), item 3 need not be in an optimal subset. Since F (2, 3)>F(1, 3), item 2 is a part of an optimal selection, which leaves element F (1, 3 − 1) to specify its remaining composition. Similarly, since F (1, 2)>F(0, 2), item 1 is the final part of the optimal solution {item 1, item 2, item 4}.

Memory functions

ALGORITHM MFKnapsack(i, j)

```
//Implements the memory function method for the knapsack problem
//Input: A nonnegative integer i indicating the number of the first
// items being considered and a nonnegative integer j indicating
// the knapsack capacity
//Output: The value of an optimal feasible subset of the first i items
//Note: Uses as global variables input arrays W eights[1..n], V alues[1..n],
//and table F[0..n, 0..W] whose entries are initialized with −1's except for
//row 0 and column 0 initialized with 0's
if F[i, j ] < 0
    if j < Weights[i]
        value ← MFKnapsack(i − 1, j)
    else
        value ← max(MFKnapsack(i − 1, j ),
            Values[i] + MFKnapsack(i − 1, j − Weights[i]))
    F[i, j ]← value
return F[i, j ]
```

Video Content / Details of website for further learning (if any):
http://www.csl.mtu.edu/cs4321/www/Lectures/Lecture%2017%20%20Knapsack%20Problem%20and%20Memory%20Function.htm
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L24

IT

II/IIV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**III- Dynamic Programming And Greedy Technique**

Date of Lecture:

Topic of Lecture: Greedy Technique-Prims algorithm

Introduction :

- The greedy approach suggests constructing a solution through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution to the problem is reached.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

For example, the widely used coin denominations in the United States are d1 = 25 (quarter), d2 = 10 (dime), d3 = 5 (nickel), and d4 = 1 (penny). How would you give change with coins of these denominations of, say, 48 cents?

If you came up with the answer 1 quarter, 2 dimes, and 3 pennies, you followed— consciously or not—a logical strategy of making a sequence of best choices among the currently available alternatives. Indeed, in the first step, you could have given one coin of any of the four denominations. "

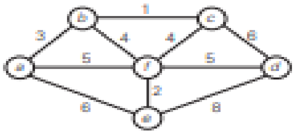
Greedy" thinking leads to giving one quarter because it reduces the remaining amount the most, namely, to 23 cents. In the second step, you had the same coins at your disposal, but you could not give a quarter, because it would have violated the problem's constraints.

So your best selection in this step was one dime, reducing the remaining amount to 13 cents. Giving one more dime left you with 3 cents to be given with three pennies. Is this solution to the instance of the change-making problem optimal? Yes, it is. In fact, one can prove that the greedy algorithm yields an optimal solution for every positive integer amount with these coin denominations. At the same time, it is easy to give an example of coin denominations that do not yield an optimal solution for some amounts—

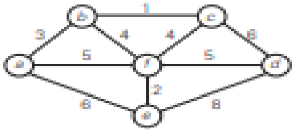
e.g., d1 = 25, d2 = 10, d3 = 1 and n = 30.

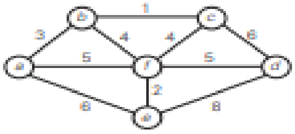
The approach applied in the opening paragraph to the change-making problem is called greedy. Computer scientists consider it a general design technique despite the fact that it is applicable to optimization problems only.

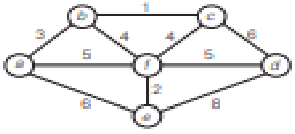
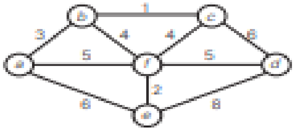
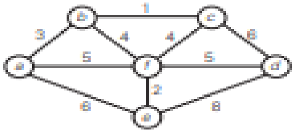
The greedy approach suggests constructing a solution through a sequence of steps, each expanding a partially constructed solution obtained so far, until a complete solution.



Tree vertices	Remaining vertices	Illustration

a(−, −)	**b(a, 3)** c(−, ∞) d(−, ∞)
	e(a, 6) f(a, 5)
		

b(a, 3)	**c(b, 1)** d(−, ∞) e(a, 6)
	f(b, 4)
		

c(b, 1)	d(c, 6) e(a, 6) **f(b, 4)**	
f(b, 4)	d(f, 5) **e(f, 2)**	
e(f, 2)	**d(f, 5)**	
d(f, 5)		

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_greedy_method.htm
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

IT

LECTURE HANDOUTS

L25

II/IIV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**III- Dynamic Programming And Greedy Technique**

Date of Lecture:

Topic of Lecture: Kruskal's Algorithm

Introduction :

Greedy algorithm that “grows” a minimum spanning tree through a greedy inclusion of the nearest vertex to the vertices already in the tree. Remarkably, there is another greedy algorithm for the minimum spanning tree problem that also always yields an optimal solution. It is named Kruskal’s algorithm.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

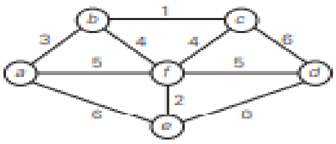
Detailed content of the Lecture:

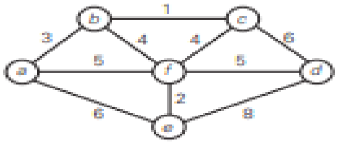
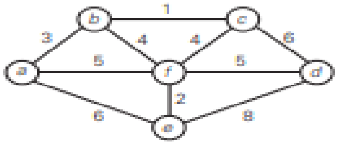
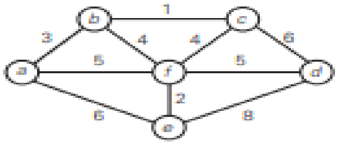
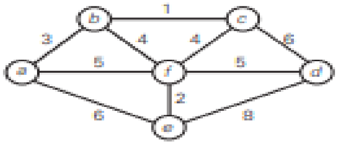
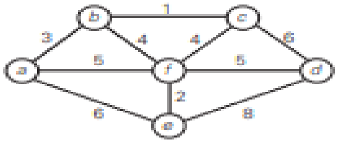
The algorithm begins by sorting the graph’s edges in non decreasing order of their weights. Then, starting with the empty subgraph, it scans this sorted list, adding the next edge on the list to the current subgraph if such an inclusion does not create a cycle and simply skipping the edge otherwise.

ALGORITHM Kruskal(G)

```
//Input: A weighted connected graph G = V,E
//Output: ET , the set of edges composing a minimum spanning tree of Gsort E in nondecreasing order of the edge weights w(ei1) ≤ ... ≤ w(ei|E|)
ET ← ∅; ecounter ← 0 //initialize the set of tree edges and its size
k ← 0 //initialize the number of processed edges
while ecounter < |V | − 1 do
k ← k + 1
if ET ∪ {eik} is acyclic
ET ← ET ∪ {eik}; ecounter ← ecounter + 1
return ET
```

EXAMPLE:



Tree edges	Sorted list of edges										Illustration
	bc **1**	ef 2	ab 3	bf 4	cf 4	af 5	df 5	ae 6	cd 6	de 8	
bc 1	bc 1	**ef** **2**	ab 3	bf 4	cf 4	af 5	df 5	ae 6	cd 6	de 8	
ef 2	bc 1	ef 2	**ab** **3**	bf 4	cf 4	af 5	df 5	ae 6	cd 6	de 8	
ab 3	bc 1	ef 2	ab 3	**bf** **4**	cf 4	af 5	df 5	ae 6	cd 6	de 8	
bf 4	bc 1	ef 2	ab 3	bf 4	cf 4	af 5	**df** **5**	ae 6	cd 6	de 8	
df 5											

Video Content / Details of website for further learning (if any):
https://en.wikipedia.org/wiki/Kruskal%27s_algorithm
https://www.geeksforgeeks.org/kruskals-minimum-spanning-tree-algorithm-greedy-algo-2/
www.khanacademy.org/computing/computer-science/algorithm
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L26

IT

II/IIV

Course Name with Code **:19ITC11/Design And Analysis Of Algorithm**

Course Teacher **:Mr.T.Manivel**

Unit **:III- Dynamic Programming And Greedy Technique**

Date of Lecture:

Topic of Lecture: Dijkstra's Algorithm

Introduction :

- It is applicable to undirected and directed graphs with nonnegative weights only. Since in most applications this condition is satisfied, the limitation has not impaired the popularity of Dijkstra's algorithm.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

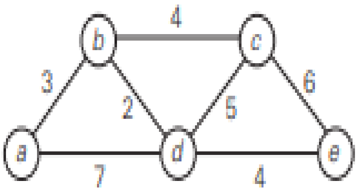
Detailed content of the Lecture:

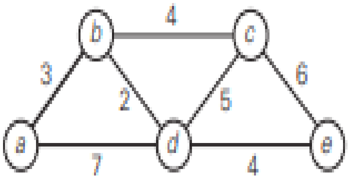
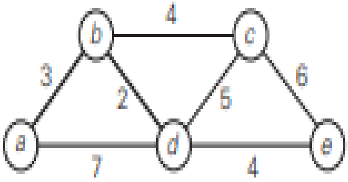
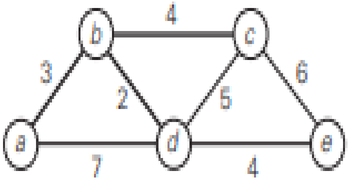
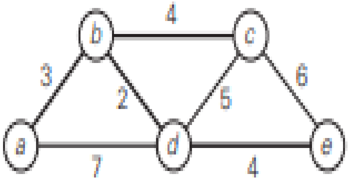
Dijkstra's algorithm finds the shortest paths to a graph's vertices in order of their distance from a given source. First, it finds the shortest path from the source to a vertex nearest to it, then to a second nearest, and so on.

ALGORITHM Dijkstra(G, s)

```
//Dijkstra's algorithm for single-source shortest paths
//Input: A weighted connected graph G = V,E with nonnegative weights
// and its vertex s
//Output: The length dv of a shortest path from s to v
// and its penultimate vertex pv for every vertex v in V
Initialize(Q) //initialize priority queue to empty
for every vertex v in V
    dv ← ∞; pv ← null
    Insert(Q, v, dv) //initialize vertex priority in the priority queue
ds ← 0; Decrease(Q, s, ds) //update priority of s with ds
VT ← ∅
for i ← 0 to |V | − 1 do
    u∗← DeleteMin(Q) //delete the minimum priority element
    VT ← VT ∪ {u∗}
    for every vertex u in V − VT that is adjacent to u∗ do
        if du∗ + w(u∗, u) < du
            du ← du∗ + w(u∗, u); pu ← u∗
            Decrease(Q, u, du)
```

Example:



Tree vertices	Remaining vertices	Illustration
a(−, 0)	**b(a, 3)** c(−, ∞) d(a, 7) e(−, ∞)	
b(a, 3)	c(b, 3 + 4) **d(b, 3 + 2)** e(−, ∞)	
d(b, 5)	**c(b, 7)** e(d, 5 + 4)	
c(b, 7)	**e(d, 9)**	
e(d, 9)		

The shortest paths (identified by following nonnumeric labels backward from a destination vertex in the left column to the source) and their lengths (given by numeric labels of the tree vertices) are as follows:

from a to b : a − b of length 3

from a to d : a − b − d of length 5

from a to c : a − b − c of length 7

from a to e : a − b − d − e of length 9

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/design_and_analysis_of_algorithms/design_and_analysis_of_algorithms_shortest_paths.htm

https://www.geeksforgeeks.org/dijkstras-shortest-path-algorithm-greedy-algo-7/

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L27

IT

II/IV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**III- Dynamic Programming And Greedy Technique**

Date of Lecture:

Topic of Lecture: Huffman Trees

Introduction :

- Variable-length encoding, which assigns codewords of different lengths to different symbols, introduces a problem that fixed-length encoding does not have. Namely, how can we tell how many bits of an encoded text represent the first (or, more generally, the ith) symbol? To avoid this complication, we can limit ourselves to the so-called prefix-free (or simply prefix) codes.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Huffman's algorithm

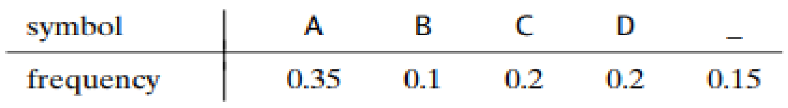
Step 1: Initialize n one-node trees and label them with the symbols of thealphabet given. Record the frequency of each symbol in its tree's rootto indicate the tree's weight. (More generally, the weight of a tree willbe equal to the sum of the frequencies in the tree's leaves.)

Step 2: Repeat the following operation until a single tree is obtained. Findtwo trees with the smallest weight (ties can be broken arbitrarily, butsee Problem 2 in this section's exercises). Make them the left and rightsubtree of a new tree and record the sum of their weights in the rootof the new tree as its weight.

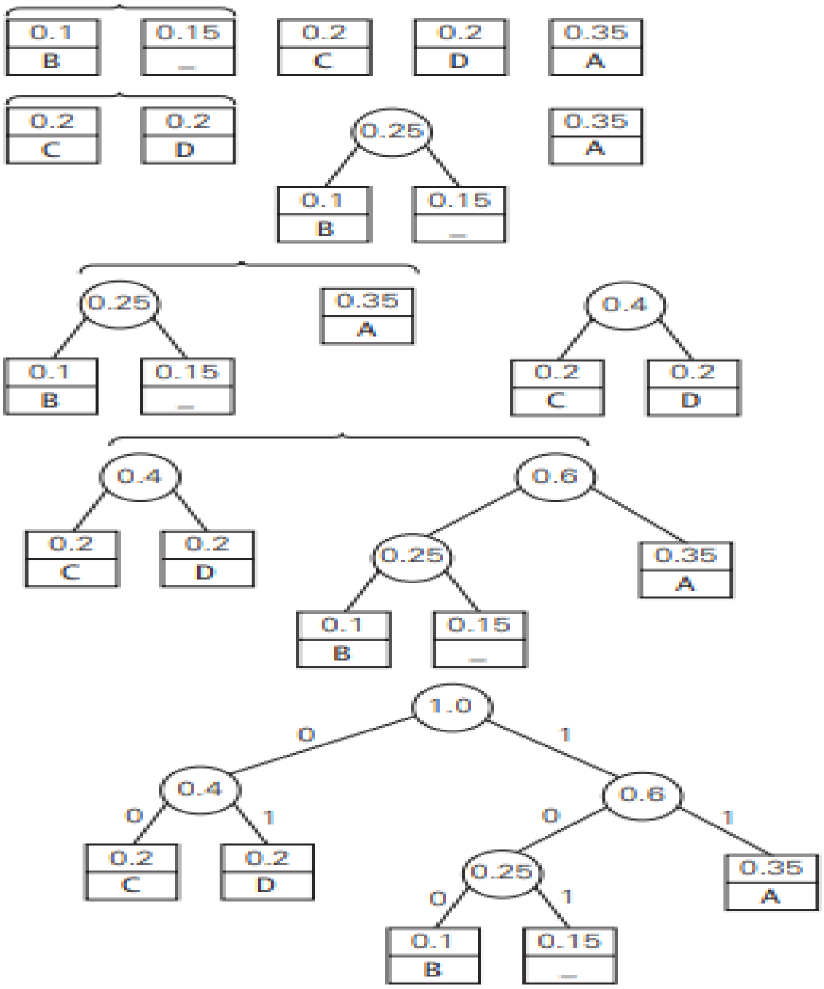
A tree constructed by the above algorithm is called a Huffman tree. It defines in the manner described above a Huffman code.

EXAMPLE

Consider the five-symbol alphabet {A, B, C, D, _} with the following occurrence frequencies in a text made up of these symbols:



symbol	A	B	C	D	_
frequency	0.35	0.1	0.2	0.2	0.15



Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/Huffman-Coding-Algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

https://www.geeksforgeeks.org/huffman-coding-greedy-algo-3/

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

L28

LECTURE HANDOUTS

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :IV-Iterative Improvement And Limitation Of Algorithm

Date of Lecture:

Topic of Lecture:The Simplex Method

Introduction :

- George B. Dantzig (1914–2005) has received many honors, including the National Medal of Sciencepresented by the president of the United States in 1976. The citation states that the National Medal wasawarded “for inventing linear programming and discovering methods that led to wide-scale scientificand technical applications to important problems in logistics, scheduling, and network optimization,and to the use of computers in making efficient use of the mathematical theory.”

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Step 0 : *Initialization* Present a given linear programming problem in standardform and set up an initial tableau with nonnegative entries in therightmost column and *m* other columns composing the *m* × *m* identitymatrix. (Entries in the objective row are to be disregarded in verifying these requirements.) These*m*columns define the basic variables of theinitial basic feasible solution, used as the labels of the tableau’s rows.

Step 1: *Optimality test* If all the entries in the objective row (except, possibly,the one in the rightmost column, which represents the value of theobjective function) are nonnegative—stop: the tableau represents anoptimal solution whose basic variables’ values are in the rightmost column and the remaining, nonbasic variables’ values are zeros.

Step 2 : *Finding the entering variable* Select a negative entry from among thefirst *n* elements of the objective row. (A commonly used rule is to selectthe negative entry with the largest absolute value, with ties brokenarbitrarily.) Mark its column to indicate the entering variable and thepivot column.

Step 3 : *Finding the departing variable* For each positive entry in the pivotcolumn, calculate the θ-ratio by dividing that row's entry in the rightmostcolumn by its entry in the pivot column. (If all the entries in thepivot column are negative or zero, the problem is unbounded—stop.)

Find the row with the smallest θ-ratio (ties may be broken arbitrarily),and mark this row to indicate the departing variable and the pivot row.

Step 4 : *Forming the next tableau* Divide all the entries in the pivot row byits entry in the pivot column. Subtract from each of the other rows,including the objective row, the new pivot row multiplied by the entryin the pivot column of the row in question. (This will make all the entries in the pivot column 0's except for 1 in the pivot row.) Replacethe label of the pivot row by the variable's name of the pivot columnand go back to Step 1.

Video Content / Details of website for further learning (if any):

https://mathworld.wolfram.com/SimplexMethod.html

https://www.hec.ca/en/cams/help/topics/The_steps_of_the_simplex_algorithm.pdf

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L29

IT

II/IV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**IV-Iterative Improvement And Limitation Of Algorithm**

Date of Lecture:

Topic of Lecture: The Maximum-Flow Problem

Introduction :

- The important problem of maximizing the flow of a material through a transportation network (pipeline system, communication system, electrical distribution system, and so on).

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

The transportationnetwork in question can be represented by a connected weighted digraph with *n* vertices numbered from 1 to *n* and a set of edges *E*, with the following properties:

- It contains exactly one vertex with no entering edges; this vertex is called the*source* and assumed to be numbered 1.
- It contains exactly one vertex with no leaving edges; this vertex is called the*sink* and assumed to be numbered *n*.
- The weight *uij* of each directed edge *(i, j)* is a positive integer, called the edge *capacity*. (This number represents the upper bound on the amount of the material that can be sent from *i* to *j* through a link represented by thisedge.)

Algorithm:

```
//Implements the shortest-augmenting-path algorithm
//Input: A network with single source 1, single sink n, and
// positive integer capacities uij on its edges (i, j )
//Output: A maximum flow x
assign xij = 0 to every edge (i, j ) in the network
label the source with ∞, − and add the source to the empty queue Q
while not Empty(Q) do
    i ← Front(Q); Dequeue(Q)
    for every edge from i to j do //forward edges
        if j is unlabeled
            rij ← uij − xij
            if rij > 0
```

```
lj ← min{li, rij }; label j with lj , i+
Enqueue(Q, j )
for every edge from j to i do //backward edges
if j is unlabeled
if xj i > 0
lj ← min{li, xj i}; label j with lj , i−
Enqueue(Q, j )
if the sink has been labeled
//augment along the augmenting path found
j ← n //start at the sink and move backwards using second labels
while j = 1 //the source hasn’t been reached
if the second label of vertex j is i+
xij ← xij + ln
else //the second label of vertex j is i−
xj i ← xj i − ln
j ← i; i ← the vertex indicated by i’s second label
erase all vertex labels except the ones of the source
reinitialize Q with the source
return x //the current flow is maximum
```

Video Content / Details of website for further learning (if any):

https://www.hackerearth.com/practice/algorithms/graphs/maximum-flow/tutorial/

https://www.geeksforgeeks.org/max-flow-problem-introduction/

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L30

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :IV-Iterative Improvement And Limitation Of Algorithm

Date of Lecture:

Topic of Lecture: Maximum Matching in Bipartite Graphs

Introduction :

- A matching in a graph is a subset of its edges with the property that no two edges share a vertex. A maximum matching—more precisely, a maximum cardinality matching—is a matching with the largest number of edges.

Prerequisite knowledge for Complete understanding and learning of Topic:

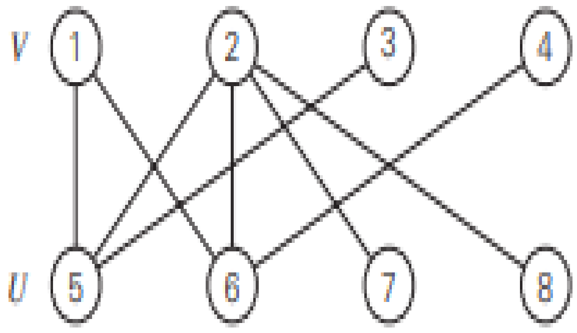
1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

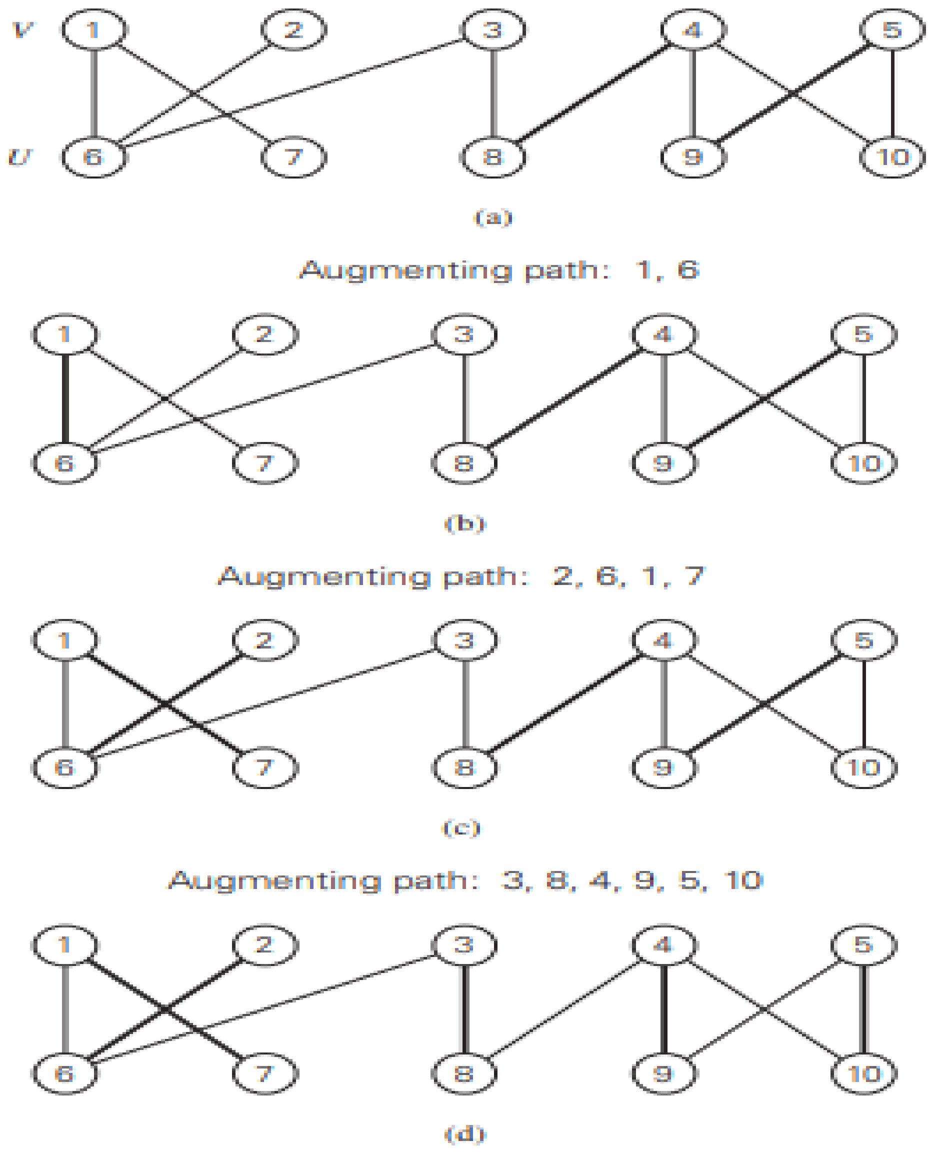
In a bipartite graph, all the vertices can be partitioned into two disjoint sets V and U, not necessarily of the same size, so that every edge connects a vertex in one of these sets to a vertex in the other set. In other words, a graph is bipartite if its vertices can be colored in two colors so that every edge has its vertices colored in different colors; such graphs are also said to be 2-colorable. The graph in Figure 10.8 is bipartite. It is not difficult to prove that a graph is bipartite if and only if it does not have a cycle of an odd length.

Example:

Consider the following



Solution:



(a)

Augmenting path: 1, 6

(b)

Augmenting path: 2, 6, 1, 7

(c)

Augmenting path: 3, 8, 4, 9, 5, 10

(d)

Maximum matching

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/Maximum-Bipartite-Matchingwww.tutorialspoint.com/design_and_analysis_of_algorithms/index.html
https://www.geeksforgeeks.org/maximum-bipartite-matching/

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

L31

IT

II/IV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**IV-Iterative Improvement And Limitation Of Algorithm**

Date of Lecture:

Topic of Lecture: the Stable marriage Problem

Introduction:

- An interesting version of bipartite matching called the stable marriage problem.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm

2.Programming Knowledge

3.Data Structure

4.Mathematical knowledge

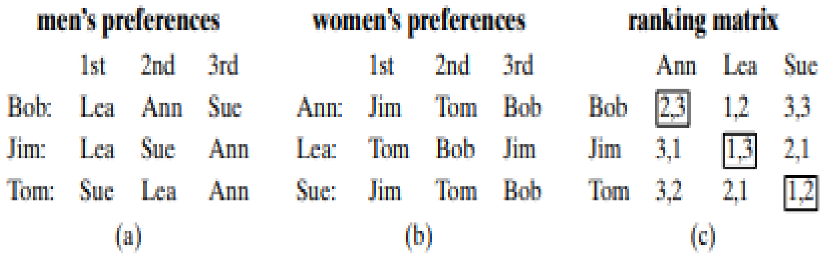
Detailed content of the Lecture:

Consider a set Y = {m1, m2,...,mn} of n men and a set X = {w1, w2,...,wn} of n women. Each man has a preference list ordering the women as potential marriage partners with no ties allowed. Similarly, each woman has a preference list of the men, also with no ties.

Examples of these two sets of lists are given. The same information can also be presented by an n × n ranking matrix .

The rows and columns of the matrix represent the men and women of the two sets, respectively. A cell in row m and column w contains two rankings: the first is the position (ranking) of w in the m's preference list; the second is the position (ranking) of m in the w's preference list. For example, the pair 3, 1 in Jim's row and Ann's column in the matrix indicates that Ann is Jim's third choice while Jim is Ann's first. Which of these two ways to represent such information is better depends on the task at hand.

For example, it is easier to specify a match of the sets' elements by using the ranking matrix, whereas the preference lists might be a more efficient data structure for implementing a matching algorithm. A marriage matching M is a set of n (m, w) pairs whose members are selected from disjoint n-element sets Y and X in a one-one fashion, i.e., each man m from Y is paired with exactly one woman w from X and vice versa. (If we represent Y and X as vertices of a complete bipartite graph with edges connecting possible marriage partners, then a marriage matching is a perfect matching in such a graph.)



men's preferences

	1st	2nd	3rd
Bob:	Lea	Ann	Sue
Jim:	Lea	Sue	Ann
Tom:	Sue	Lea	Ann

(a)

women's preferences

	1st	2nd	3rd
Ann:	Jim	Tom	Bob
Lea:	Tom	Bob	Jim
Sue:	Jim	Tom	Bob

(b)

ranking matrix

	Ann	Lea	Sue
Bob	$\boxed{2,3}$	1,2	3,3
Jim	3,1	$\boxed{1,3}$	2,1
Tom	3,2	2,1	$\boxed{1,2}$

(c)

A pair (m, w), where m ∈ Y, w ∈ X, is said to be a blocking pair for a marriage matching M if man m and woman w are not matched in M but they prefer each other to their mates in M. For example, (Bob, Lea) is a blocking pair for the marriage matching M = {(Bob, Ann), (Jim, Lea), (Tom, Sue)} because they are not matched in M while Bob prefers Lea to Ann and Lea prefers Bob to Jim. A marriage matching M is called stable if there is no blocking pair for it; otherwise, M is called unstable. According to this definition, the marriage matching in (c) is unstable because Bob and Lea can drop their designated mates to join in a union they both prefer. The stable marriage problem is to find a stable marriage matching for men's and women's given preferences

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/stable-marriage-problem/Memory%20Function.htm
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html
http://www.cs.cmu.edu/~arielpro/15896s16/slides/896s16-16.pdf

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L32

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :IV-Iterative Improvement And Limitation Of Algorithm

Date of Lecture:

Topic of Lecture: Limitations of Algorithm Power

Introduction :

- Tt is desirable to know the best possible efficiency any algorithm solving the problem may have. Knowing such a lower bound can tell us how much improvement we can hope to achieve in our quest for a better algorithm for the problem in question

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

The last section of this chapter deals with numerical analysis. This branch of computer science concerns algorithms for solving problems of "continuous" mathematics—solving equations and systems of equations, evaluating such functions as sin x and ln x, computing integrals, and so on.

The nature of such problems imposes two types of limitations. First, most cannot be solved exactly. Second, solving them even approximately requires dealing with numbers that can be represented in a digital computer with only a limited level of precision.

Manipulating approximate numbers without proper care can lead to very inaccurate results. We will see that even solving a basic quadratic equation on a computer poses significant difficulties that require a modification of the canonical formula for the equation's roots.

Video Content / Details of website for further learning (if any):

https://www.slideshare.net/muthukrishnavinayaga/coping-with-the-limitations-of-algorithm-power

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

L33

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :IV-Iterative Improvement And Limitation Of Algorithm

Date of Lecture:

Topic of Lecture: Lower-Bound Arguments

Introduction :

Knowing such a lower bound can tell us how much improvement we can hope to achieve in our quest for a better algorithm for the problem in question

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Trival lower bounds:

The simplest method of obtaining a lower-bound class is based on counting the number of items in the problem's input that must be processed and the number of output items that need to be produced. Since any algorithm must at least "read" all the items it needs to process and "write" all its outputs, such a count yields a trivial lower bound.

Information theoretic arguments:

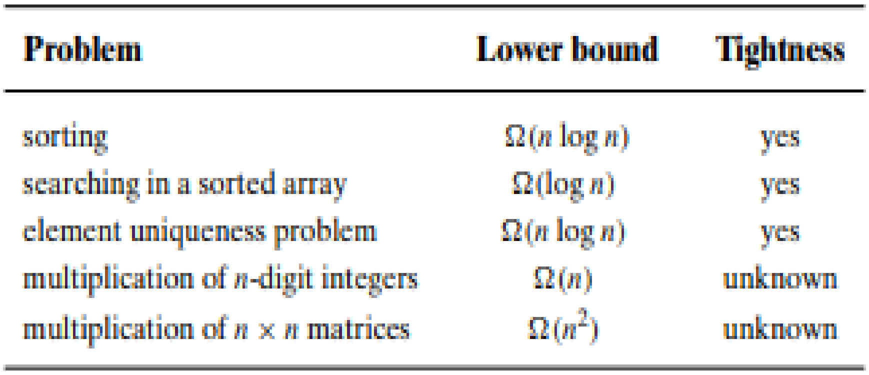
The approach we just exploited is called the information-theoretic argument because of its connection to information theory. It has proved to be quite useful for finding the so-called information-theoretic lower bounds for many problems involving comparisons, including sorting and searching. Its underlying idea can be realized much more precisely through the mechanism of decision trees.

Adversary arguments:

adversary method for establishing lower bounds. It is based on following the logic of a malevolent but honest adversary: the malevolence makes him push the algorithm down the most time-consuming path, and his honesty forces him to stay consistent with the choices already made. A lower bound is then obtained by measuring the amount of work needed to shrink a set of potential inputs to a single input along the most time-consuming path.

Problem reduction:

An algorithm for problem P by reducing it to another problem Q solvable with a known algorithm. A similar reduction idea can be used for finding a lower bound. To show that problem P is at least as hard as another problem Q with a known lower bound, we need to reduce Q to P (not P to Q!).



Problem	Lower bound	Tightness
sorting	$\Omega(n \log n)$	yes
searching in a sorted array	$\Omega(\log n)$	yes
element uniqueness problem	$\Omega(n \log n)$	yes
multiplication of n-digit integers	$\Omega(n)$	unknown
multiplication of $n \times n$ matrices	$\Omega(n^2)$	unknown

Video Content / Details of website for further learning (if any):

https://www.javatpoint.com/daa-lower-bound-theory

https://www.brainkart.com/article/Lower-Bound-Arguments_8057/

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

DESIGNING YOUR FUTURE
Estd. 2000

IQAC

LECTURE HANDOUTS

L34

IT

II/IV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**IV-Iterative Improvement And Limitation Of Algorithm**

Date of Lecture:

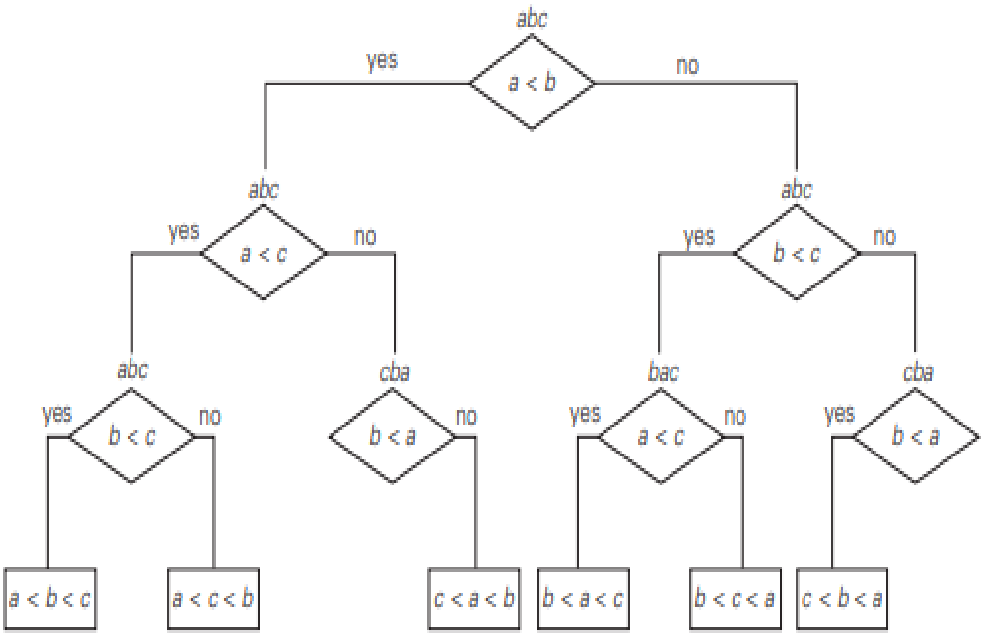
Topic of Lecture: Decision Trees

Introduction :
Most sorting algorithms are comparison based, i.e., they work by comparing elements in a list to be sorted. By studying properties of decision trees for such algorithms, we can derive important lower bounds on their time efficiencies.

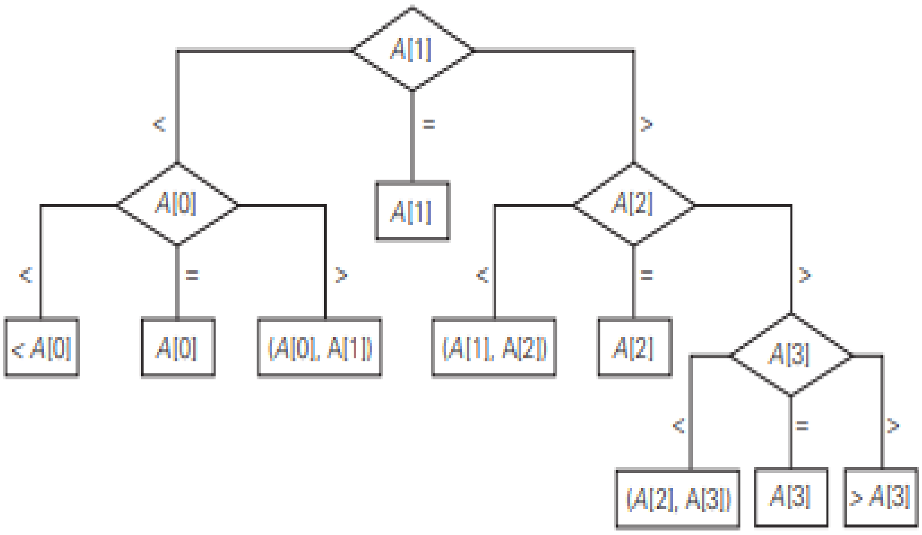
Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:
Algorithms, we can derive important lower bounds on their time efficiencies. We can interpret an outcome of a sorting algorithm as finding a permutation of the element indices of an input list that puts the list's elements in ascending order. Consider, as an example, a three-element list a, b, c of orderable items such as real numbers or strings. For the outcome a<c<b obtained by sorting this list.



Decision tree for searching a sorted array:
In this section, we shall see how decision trees can be used for establishing lower bounds on the number of key comparisons in searching a sorted array of n keys: A[0] < A[1] < ... < A[n − 1]. The principal algorithm for this problem is binary search. As we saw in Section 4.4, the number of comparisons made by binary search in the worst case, Cbs worst(n), is given by the formula
 Cbs worst(n) = [log2 n] + 1 = [Log2(n + 1)]



Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/data_mining/dm_dti.htm
www.khanacademy.org/computing/computer-science/algorithm
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L35

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :IV-Iterative Improvement And Limitation Of Algorithm

Date of Lecture:

Topic of Lecture: P, NP and NP Complete Problems

Introduction :

An algorithm solves a problem in polynomial time if its worst-case time efficiency belongs to O(p(n)) where p(n) is a polynomial of the problem's input size n. (Note that since we are using big-oh notation here, problems solvable in, say, logarithmic time are solvable in polynomial time as well.)

Problems that can be solved in polynomial time are called tractable, and problems that cannot be solved in polynomial time are called intractable.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

P, NP and NP complete problems:

In the study of the computational complexity of problems, the first concern of both computer scientists and computing professionals is whether a given problem can be solved in polynomial time by some algorithm.

Definition 1:

We say that an algorithm solves a problem in polynomial time if its worst-case time efficiency belongs to O(p(n)) where p(n) is a polynomial of the problem's input size n. (Note that since we are using big-oh notation here, problems solvable in, say, logarithmic time are solvable in polynomial time as well.) Problems that can be solved in polynomial time are called tractable, and problems that cannot be solved in polynomial time are called intractable.

Definition 2:

Class P is a class of decision problems that can be solved in polynomial time by (deterministic) algorithms. This class of problems is called polynomial.

Definition 3:

A nondeterministic algorithm is a two-stage procedure that takes as its input an instance I of a decision problem and does the following. Nondeterministic ("guessing") stage: An arbitrary string S is generated that can be thought of as a candidate solution to the given instance I (but may be complete gibberish as well).

Definition 4:

Class NP is the class of decision problems that can be solved by nondeterministic polynomial algorithms. This class of problems is called nondeterministic polynomial.

Definition 5:

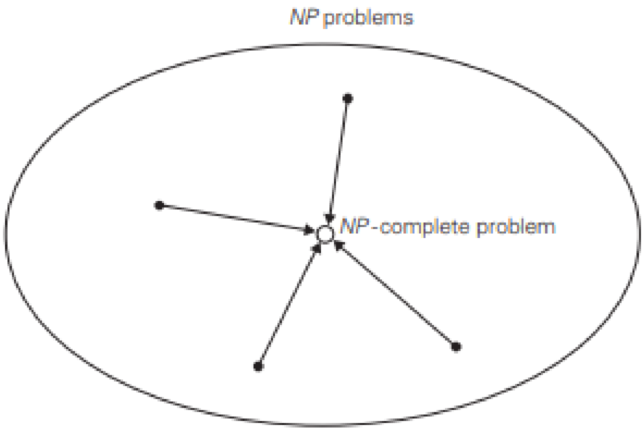
A decision problem D1 is said to be polynomially reducible to a decision problem D2, if there exists a function t that transforms instances of D1 to instances of D2 such that:

1. t maps all yes instances of D1 to yes instances of D2 and all no instances of D1 to no instances of D2
2. t is computable by a polynomial time algorithm

Definition 6:

A decision problem D is said to be NP-complete if:

1. it belongs to class NP
2. every problem in NP is polynomially reducible to D



Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/data_mining/dm_dti.htm

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L36

IT

II/IV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**

Course Teacher :**Mr.T.Manivel**

Unit :**IV-Iterative Improvement And Limitation Of Algorithm**

Date of Lecture:

Topic of Lecture: P, NP and NP Complete Problems

Introduction :

An algorithm solves a problem in polynomial time if its worst-case time efficiency belongs to O(p(n)) where p(n) is a polynomial of the problem’s input size n. (Note that since we are using big-oh notation here, problems solvable in, say, logarithmic time are solvable in polynomial time as well.)

Problems that can be solved in polynomial time are called tractable, and problems that cannot be solved in polynomial time are called intractable.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

P, NP and NP complete problems:

In the study of the computational complexity of problems, the first concern of both computer scientists and computing professionals is whether a given problem can be solved in polynomial time by some algorithm.

Definition 1:

We say that an algorithm solves a problem in polynomial time if its worst-case time efficiency belongs to O(p(n)) where p(n) is a polynomial of the problem’s input size n. (Note that since we are using big-oh notation here, problems solvable in, say, logarithmic time are solvable in polynomial time as well.) Problems that can be solved in polynomial time are called tractable, and problems that cannot be solved in polynomial time are called intractable.

Definition 2:

Class P is a class of decision problems that can be solved in polynomial time by (deterministic) algorithms. This class of problems is called polynomial.

Definition 3:

A nondeterministic algorithm is a two-stage procedure that takes as its input an instance I of a decision problem and does the following. Nondeterministic (“guessing”) stage: An arbitrary string S is generated that can be thought of as a candidate solution to the given instance I (but may be complete gibberish as well).

Definition 4:

Class NP is the class of decision problems that can be solved by nondeterministic polynomial algorithms. This class of problems is called nondeterministic polynomial.

Definition 5:

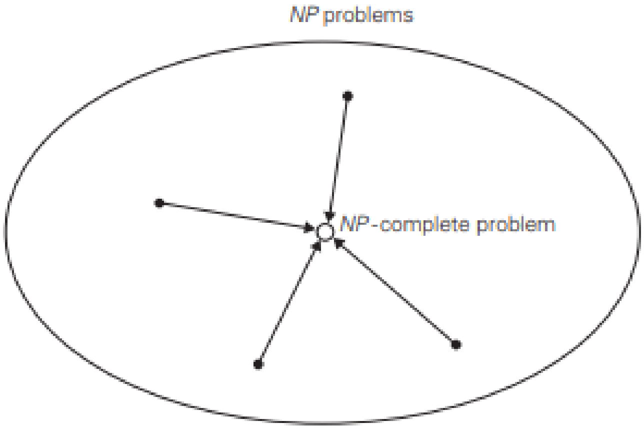
A decision problem D1 is said to be polynomially reducible to a decision problem D2, if there exists a function t that transforms instances of D1 to instances of D2 such that:

1. t maps all yes instances of D1 to yes instances of D2 and all no instances of D1 to no instances of D2
2. t is computable by a polynomial time algorithm

Definition 6:

A decision problem D is said to be NP-complete if:

1. it belongs to class NP
2. every problem in NP is polynomially reducible to D



Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/data_mining/dm_dti.htm

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L37

IT

II/IV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**
Course Teacher :**Mr.T.Manivel**
Unit :**V-Backtracking ,Branch And Bound And Approximation Algorithm**

Date of Lecture:

Topic of Lecture:Backtracking

Introduction :

- Backtracking is a more intelligent variation of this approach. The principal idea is to construct solutions one component at a time and evaluate such partially constructed candidates as follows. If a partially constructed solution can be developed further without violating the problem's constraints, it is done by taking the first remaining legitimate option for the next component.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Backtracking is a more intelligent variation of this approach. The principalidea is to construct solutions one component at a time and evaluate such partiallyconstructed candidates as follows. If a partially constructed solution can be developed further without violating the problem's constraints, it is done by takingthe first remaining legitimate option for the next component.
If there is no legitimate option for the next component, no alternatives for any remaining componentneed to be considered.
In this case, the algorithm backtracks to replace the lastcomponent of the partially constructed solution with its next option.It is convenient to implement this kind of processing by constructing a treeof choices being made, called the state-space tree. Its root represents an initialstate before the search for a solution begins.
The nodes of the first level in thetree represent the choices made for the first component of a solution, the nodesof the second level represent the choices for the second component, and soon. A node in a state-space tree is said to be promising if it corresponds to a
partially constructed solution that may still lead to a complete solution; otherwise it is called nonpromising.
Leaves represent either nonpromising dead ends or complete solutions found by the algorithm. In the majority of cases, a statespace tree for a backtracking algorithm is constructed in the manner of depthfirst search. If the current node is promising, its child is generated by adding the first remaining legitimate option for the next component of a solution, and the processing moves to this child. If the current node turns out to be nonpromising, the algorithm backtracks to the node's parent to consider

the next possible option for its last component;
if there is no such option, it backtracks one more level up the tree, and so on. Finally, if the algorithm reaches a complete solution to the problem, it either stops (if just one solution is required) or continues searching for other possible solutions.

Video Content / Details of website for further learning (if any):
https://mathworld.wolfram.com/SimplexMethod.html
https://www.hec.ca/en/cams/help/topics/The_steps_of_the_simplex_algorithm.pdf

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L38

IT

II/IV

Course Name with Code :**19ITC11/Design And Analysis Of Algorithm**
Course Teacher :**Mr.T.Manivel**
Unit :**V-Backtracking ,Branch And Bound And Approximation Algorithm**

Date of Lecture:

Topic of Lecture: n-Queens problem

Introduction :

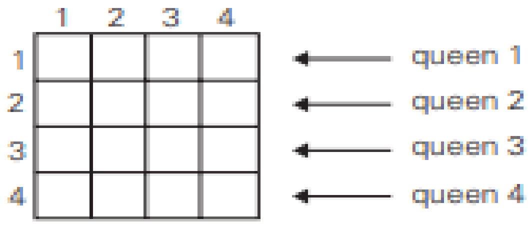
- The problem is to place n queens on an n × n chessboard so that no two queens attack each other by being in the same row or in the same column or on the same diagonal

Prerequisite knowledge for Complete understanding and learning of Topic:

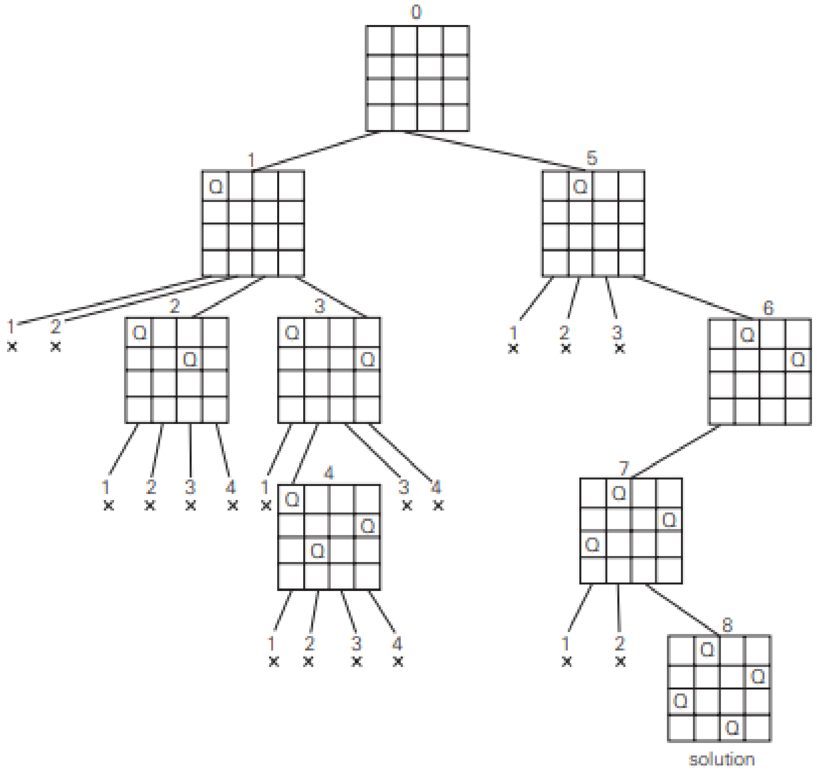
1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

The problem is to place n queens on an n × n chessboard so that no two queens attack each other by being in the same row or in the same column or on the same diagonal. For n = 1, the problem has a trivial solution, and it is easy to see that there is no solution for n = 2 and n = 3. So let us consider the four-queens problem and solve it by the backtracking technique. Since each of the four queens has to be placed in its own row, all we need to do is to assign a column for each queen on the board presented in Figure 12.1. We start with the empty board and then place queen 1 in the first possible position of its row, which is in column 1 of row 1. Then we place queen 2, after trying unsuccessfully columns 1 and 2, in the first acceptable position for it, which is square (2, 3), the square in row 2 and column 3. This proves to be a dead end because there is no acceptable position for queen 3. So, the algorithm backtracks and puts queen 2 in the next possible position at (2, 4). Then queen 3 is placed at (3, 2), which proves to be another dead end. The algorithm then backtracks all the way to queen 1 and moves it to (1, 2). Queen 2 then goes to (2, 4), queen 3 to (3, 1), and queen 4 to (4, 3), which is a solution to the problem.



Solution:



Video Content / Details of website for further learning (if any):

https://www.hackerearth.com/practice/algorithms/graphs/maximum-flow/tutorial/

https://www.geeksforgeeks.org/max-flow-problem-introduction/

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L39

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :V-Backtracking ,Branch And Bound And Approximation Algorithm

Date of Lecture:

Topic of Lecture: Hamiltonian Circuit Problem

Introduction :

- Hamiltonian circuit problem is used to find the optimal solution of a problem.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

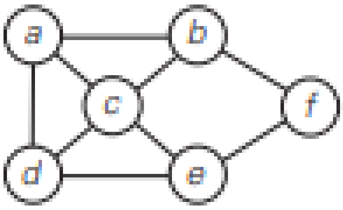
let us consider the problem of finding a Hamiltonian circuit in the graph. Without loss of generality, we can assume that if a Hamiltonian circuit exists, it starts at vertex a. Accordingly, we make vertex a the root of the state-space tree.

The first component of our future solution, if it exists, is a first intermediate vertex of a Hamiltonian circuit to be constructed.

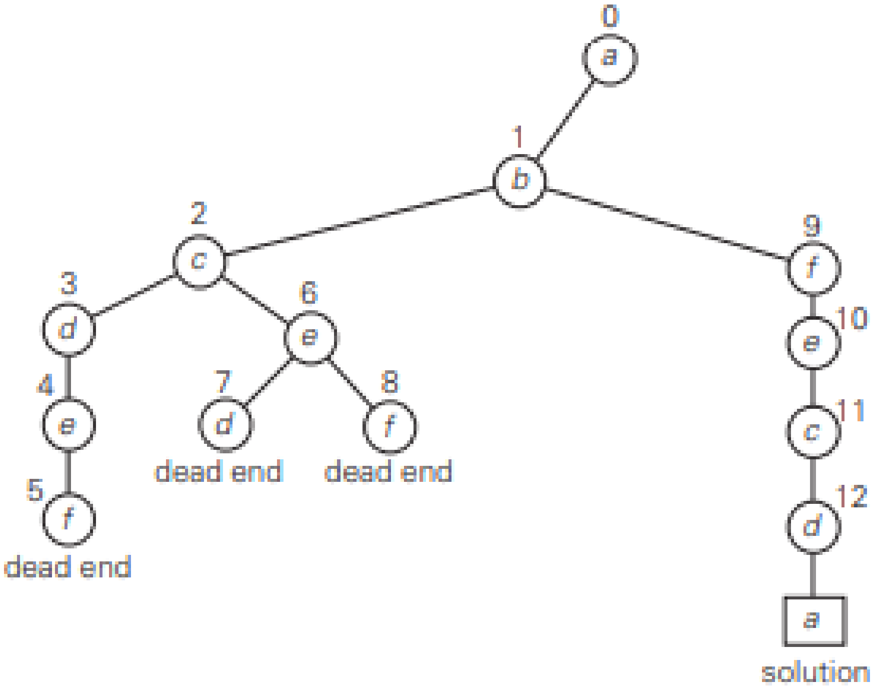
Using the alphabet order to break the three-way tie among the vertices adjacent to a, we select vertex b. From b, the algorithm proceeds to c, then to d, then to e, and finally to f, which proves to be a dead end.

So the algorithm backtracks from f to e, then to d, and then to c, which provides the first alternative for the algorithm to pursue. Going from c to e eventually proves useless, and the algorithm has to backtrack from e to c and then to b.

From there, it goes to the vertices f , e, c, and d, from which it can legitimately return to a, yielding the Hamiltonian circuit a, b, f , e, c, d, a. If we wanted to find another Hamiltonian circuit, we could continue this process by backtracking from the leaf of the solution found.



Solution:



Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/Maximum-Bipartite-Matchingwww.tutorialspoint.com/design_and_analysis_of_algorithms/index.html
https://www.geeksforgeeks.org/maximum-bipartite-matching/

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L40

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :V-Backtracking ,Branch And Bound And Approximation Algorithm

Date of Lecture:

Topic of Lecture: Subset Sum Problem

Introduction:

- Subset sum problem is used to identify an optimal solution for a problem.

Prerequisite knowledge for Complete understanding and learning of Topic:

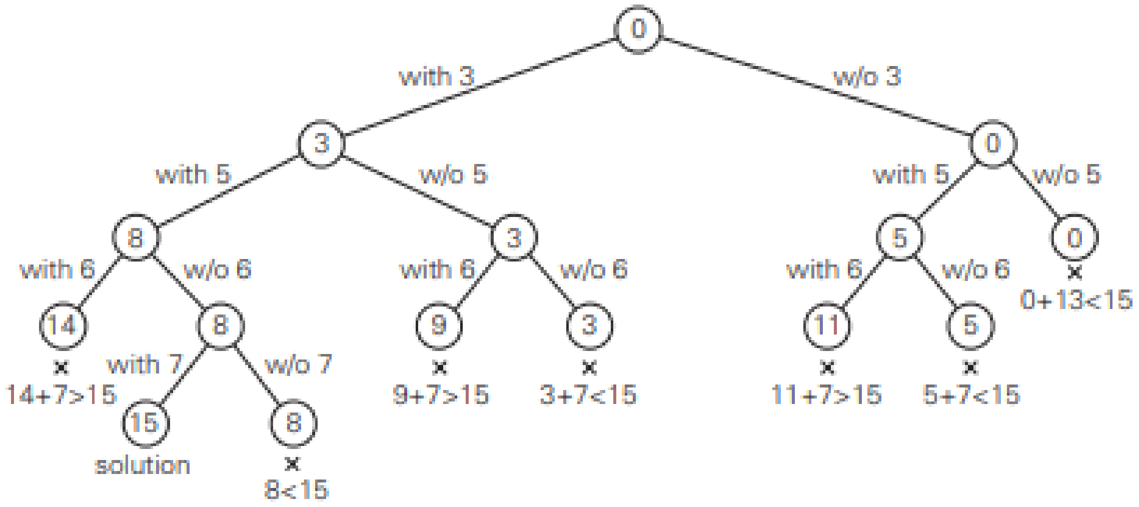
1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

consider the subset-sum problem: find a subset of a givenset A = {a1,...,an} of n positive integers whose sum is equal to a given positiveinteger d. For example, for A = {1, 2, 5, 6, 8} and d = 9, there are two solutions:{1, 2, 6} and {1, 8}. Of course, some instances of this problem may have no solutions.

It is convenient to sort the set's elements in increasing order. So, we willassume that

a1 < a2 < ... < an.



The state-space tree can be constructed as a binary tree like that for the instance A = {3, 5, 6, 7} and d = 15. The root of the tree represents the starting point, with no decisions about the given elements made as yet. Its left and right children represent, respectively, inclusion and exclusion of a1 in a set being sought. Similarly, going to the left from a node of the first level corresponds to inclusion of a2 while going to the right corresponds to its exclusion, and so on. Thus, a path from the root to a node on the ith level of the tree indicates which of the first i numbers have been included in the subsets represented by that node. We record the value of s, the sum of these numbers, in the node. If s is equal to d, we have a solution to the problem. We can either report this result and stop or, if all the solutions need to be found, continue by backtracking to the node's parent. If s is not equal to d, we can terminate the node as nonpromising if either of the following two inequalities holds:

$$s + a_{i+1} > d \quad \text{(the sum } s \text{ is too large),}$$

$$s + \sum_{j=i+1}^{n} a_j < d \quad \text{(the sum } s \text{ is too small).}$$

Video Content / Details of website for further learning (if any):

https://www.geeksforgeeks.org/stable-marriage-problem/Memory%20Function.htm
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html
http://www.cs.cmu.edu/~arielpro/15896s16/slides/896s16-16.pdf

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L41

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :V-Backtracking ,Branch And Bound And Approximation Algorithm

Date of Lecture:

Topic of Lecture: Branch and Bound

Introduction :

- This idea can be strengthened further if we deal with an optimization problem. An optimization problem seeks to minimize or maximize some objective function (a tour length, the value of items selected, the cost of an assignment, and the like), usually subject to some constraints. Note that in the standard terminology of optimization problems, a feasible solution is a point in the problem's search space that satisfies all the problem's constraints.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

Compared to backtracking, branch-and-bound requires two additional items:

- a way to provide, for every node of a state-space tree, a bound on the best value of the objective function1 on any solution that can be obtained by adding further components to the partially constructed solution represented by the node
- the value of the best solution seen so far

If this information is available, we can compare a node's bound value with the value of the best solution seen so far. If the bound value is not better than the value of the best solution seen so far—i.e., not smaller for a minimization problem and not larger for a maximization problem—the node is nonpromising and can be terminated (some people say the branch is "pruned"). Indeed, no solution obtained from it can yield a better solution than the one already available. This is the principal idea of the branch-and-bound technique.

In general, we terminate a search path at the current node in a state-space tree of a branch-and-bound algorithm for any one of the following three reasons:

- The value of the node's bound is not better than the value of the best solution seen so far.
- The node represents no feasible solutions because the constraints of the problem are already violated.
- The subset of feasible solutions represented by the node consists of a single point (and hence no further choices can be made)—in this case, we compare the value of the objective function for this feasible solution with that of the best solution seen so far and update the latter with the former if the new solution is better.

Video Content / Details of website for further learning (if any):
https://www.slideshare.net/muthukrishnavinayaga/coping-with-the-limitations-of-algorithm-power
www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:
Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS **L42**

IT **II/IV**

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :V-Backtracking ,Branch And Bound And Approximation Algorithm

Date of Lecture:

Topic of Lecture: Assignment problem

Introduction :

Assignment problem are used to identify the job to be assigned for each person to complete the task in efficient.

Prerequisite knowledge for Complete understanding and learning of Top

1.Basics of Algorithm

2.Programming Knowledge

3.Data Structure

4.Mathematical knowledge

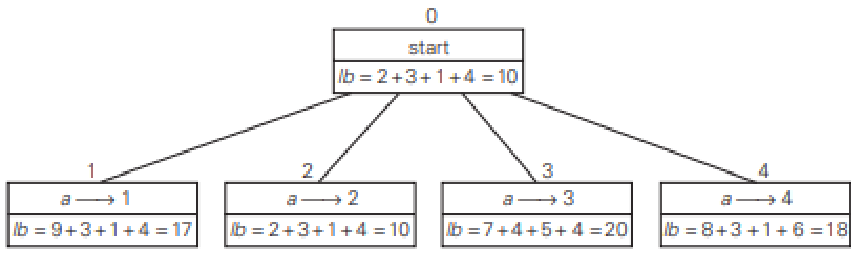
Detailed content of the Lecture:

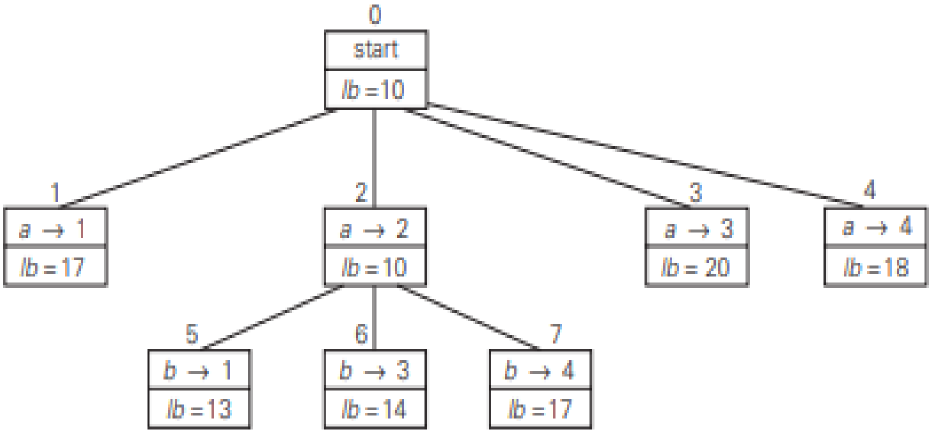
Let us illustrate the branch-and-bound approach by applying it to the problem of assigning n people to n jobs so that the total cost of the assignment is as small as possible. We introduced this problem in Section 3.4, where we solved it by exhaustive search. Recall that an instance of the assignment problem is specified by an n × n cost matrix C so that we can state the problem as follows: select one element in each row of the matrix so that no two selected elements are in the same column and their sum is the smallest possible. We will demonstrate how this problem can be solved using the branch-and-bound technique by considering the same small instance of the problem.

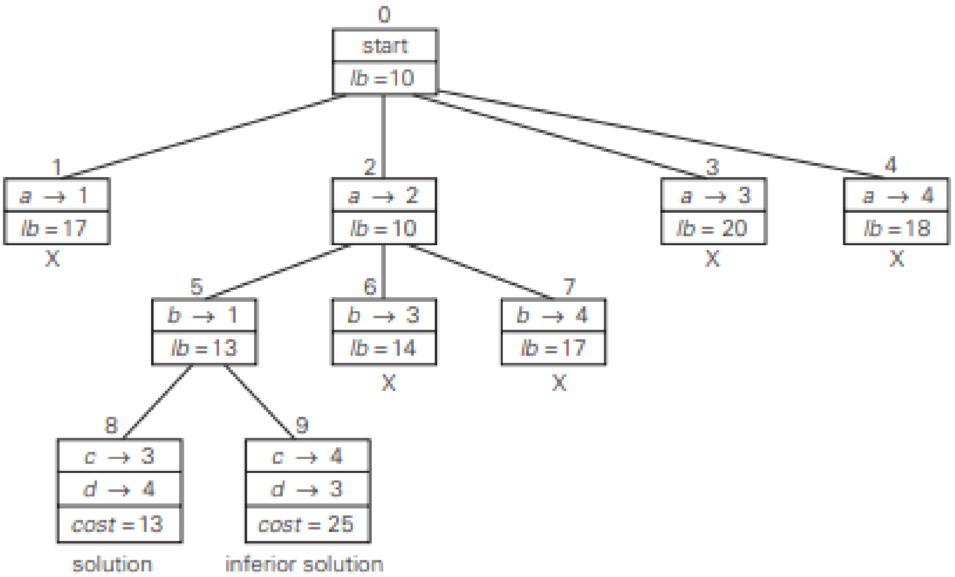
$$C = \begin{array}{ccccl} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} & \\ 9 & 2 & 7 & 8 & \text{person } a \\ 6 & 4 & 3 & 7 & \text{person } b \\ 5 & 8 & 1 & 8 & \text{person } c \\ 7 & 6 & 9 & 4 & \text{person } d \end{array}$$

Solution:

Here, this sum is 2 + 3 + 1 + 4 = 10. It is important to stress that this is not the cost of any legitimate selection (3 and 1 came from the same column of the matrix); it is just a lower bound on the cost of any legitimate selection. We can and will apply the same thinking to partially constructed solutions. the lower bound will be 9 + 3 + 1 + 4 = 17.Of the six live leaves—nodes 1, 3, 4, 5, 6, and 7—that may contain an optimal solution, we again choose the one with the smallest lower bound, node 5. First, we consider selecting the third column’s element from c’s row.







Video Content / Details of website for further learning (if any):

https://www.javatpoint.com/daa-lower-bound-theory

https://www.brainkart.com/article/Lower-Bound-Arguments_8057/

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L43

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :V-Backtracking ,Branch And Bound And Approximation Algorithm

Date of Lecture:

Topic of Lecture: Knapsack Problem

Introduction :
The branch-and-bound technique is used to solving the knapsack problem.

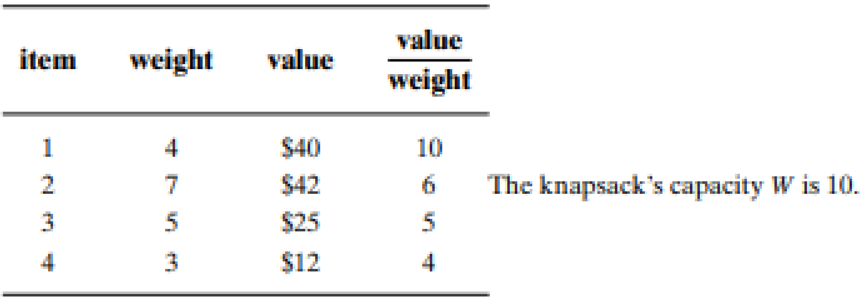
Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:
A simple way to compute the upper bound ub is to add to v, the total value of the items already selected, the product of the remaining capacity of the knapsack W − w and the best per unit payoff among the remaining items, which is vi+1/wi+1:

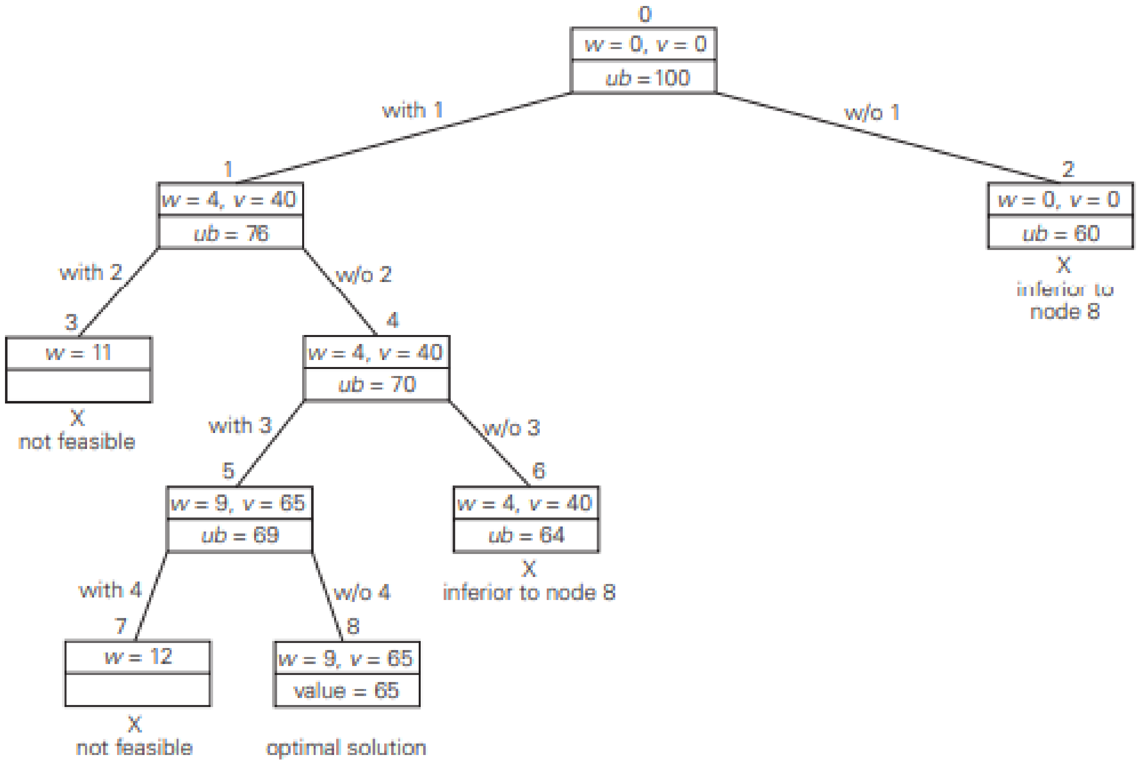
ub = v + (W − w)(vi+1/wi+1)

As a specific example, let us apply the branch-and-bound algorithm to the same instance of the knapsack problem we solved in Section 3.4 by exhaustive search. (We reorder the items in descending order of their value-to-weight ratios, though.)



item	weight	value	$\frac{\text{value}}{\text{weight}}$
1	4	\$40	10
2	7	\$42	6
3	5	\$25	5
4	3	\$12	4

The knapsack's capacity W is 10.



Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/data_mining/dm_dti.htm

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms
Design and Analysis of Algorithms

Course Teacher

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L44

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :V-Backtracking ,Branch And Bound And Approximation Algorithm

Date of Lecture:

Topic of Lecture: Traveling Salesman Problem

Introduction :

To apply the branch-and-bound technique to instances of the traveling salesman problem if we come up with a reasonable lower bound on tour lengths. One very simple lower bound can be obtained by finding the smallest element in the intercity distance matrix D and multiplying it by the number of cities n.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

One very simple lower bound can be obtained by finding the smallest element in the intercity distance matrix D and multiplying it by the number of cities n.

But there is a less obvious and more informative lower bound for instances with symmetric matrix D, which does not require a lot of work to compute. It is not difficult to show (Problem 8 in this section's exercises) that we can compute a lower bound on the length l of any tour as follows.

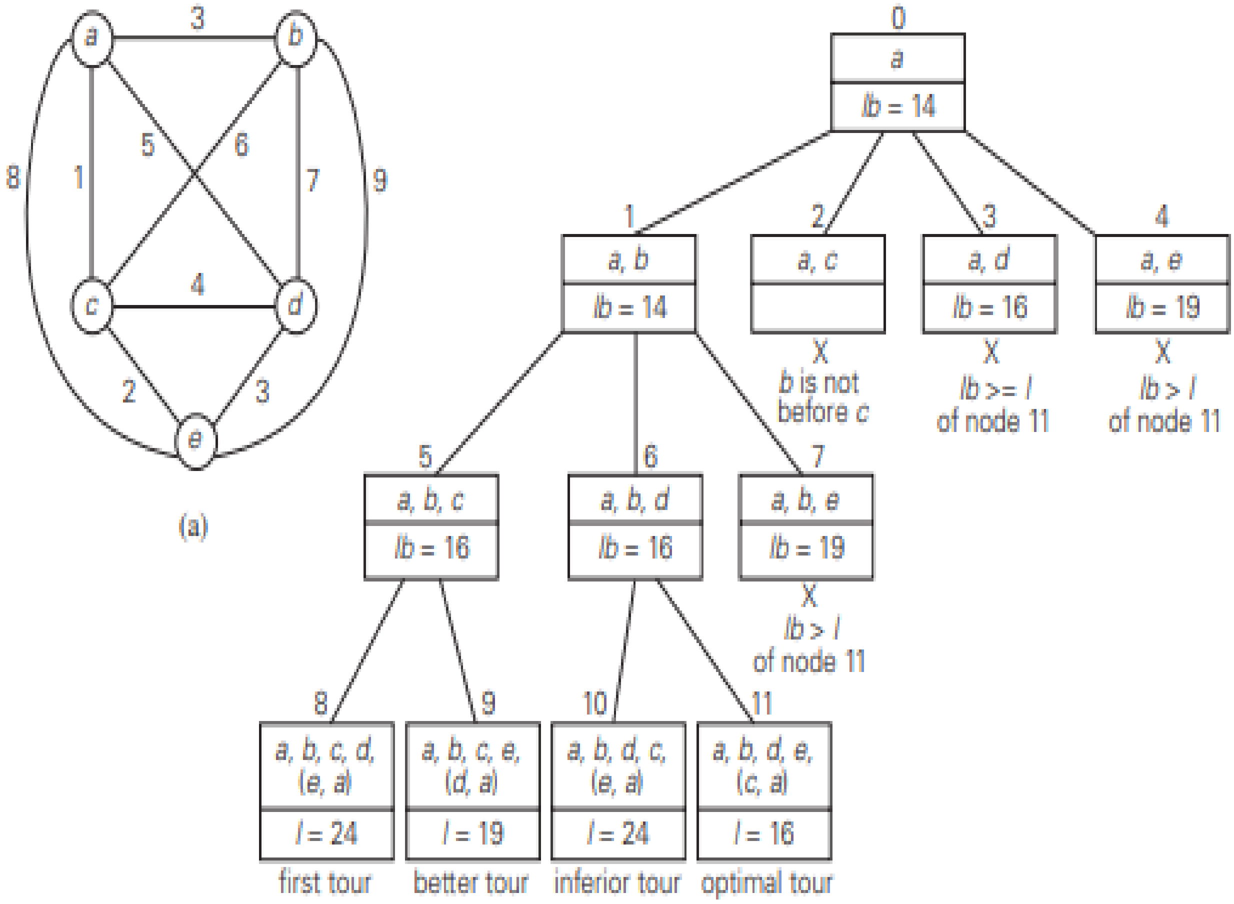
For each city i, $1 \leq i \leq n$, find the sum si of the distances from city i to the two nearest cities; compute the sum s of these n numbers, divide the result by 2, and, if all the distances are integers, round up the result to the nearest integer:

$$lb =[s/2]$$

For example, for the instance, formula yields

$$lb = [[(1 + 3) + (3 + 6) + (1 + 2) + (3 + 4) + (2 + 3)]/2] = 14$$

we get the following lower bound by summing up the lengths of the two shortest edges incident with each of the vertices, with the required inclusion of edges(a, d) and (d, a): $[(1 + 5) + (3 + 6) + (1 + 2) + (3 + 5) + (2 + 3)]/2] = 16$.



Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/data_mining/dm_dti.htm

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD

LECTURE HANDOUTS

L45

IT

II/IV

Course Name with Code :19ITC11/Design And Analysis Of Algorithm

Course Teacher :Mr.T.Manivel

Unit :V-Backtracking ,Branch And Bound And Approximation Algorithm

Date of Lecture:

Topic of Lecture: : Approximation Algorithms for NP-hard problems

Introduction :

NP-hard problems that are at least as hard as NP-complete problems. Hence, there are no known polynomial-time algorithms for these problems, and there are serious theoretical reasons to believe that such algorithms do not exist.

Prerequisite knowledge for Complete understanding and learning of Topic:

1.Basics of Algorithm
2.Programming Knowledge
3.Data Structure
4.Mathematical knowledge

Detailed content of the Lecture:

If an instance of the problem in question is very small, we might be able to solve it by an exhaustive-search algorithm.

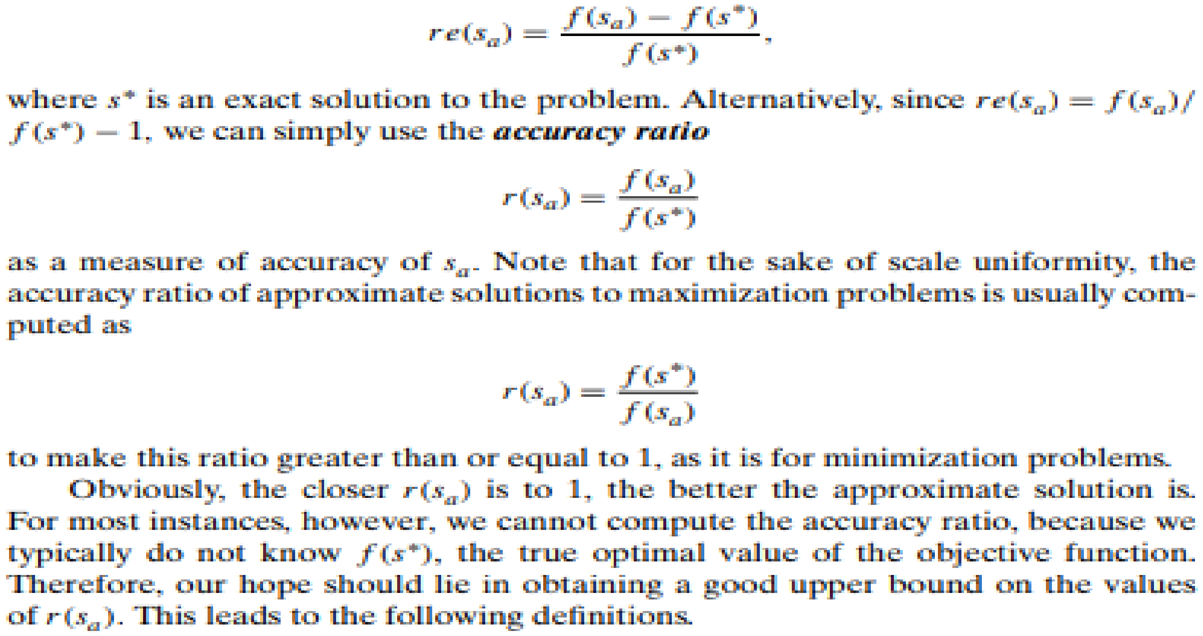
But even when this approach works in principle, its practicality is limited by dependence on the instance parameters being relatively small. The discovery of the branch-and-bound technique has proved to be an important breakthrough, because this technique makes it possible to solve many large instances of difficult optimization problems in an acceptable amount of time.

However, such good performance cannot usually be guaranteed. There is a radically different way of dealing with difficult optimization problems: solve them approximately by a fast algorithm.

This approach is particularly appealing for applications where a good but not necessarily optimal solution will suffice. Besides, in real-life applications, we often have to operate with inaccurate data to begin with.

Under such circumstances, going for an approximate solution can be a particularly sensible choice. Although approximation algorithms run a gamut in level of sophistication, most of them are based on some problem-specific heuristic.

A heuristic is a common-sense rule drawn from experience rather than from a mathematically proved assertion. For example, going to the nearest unvisited city in the traveling salesman problem is a good illustration of this notion.



$$re(s_a) = \frac{f(s_a) - f(s^*)}{f(s^*)},$$

where s^* is an exact solution to the problem. Alternatively, since $re(s_a) = f(s_a)/f(s^*) - 1$, we can simply use the ***accuracy ratio***

$$r(s_a) = \frac{f(s_a)}{f(s^*)}$$

as a measure of accuracy of s_a. Note that for the sake of scale uniformity, the accuracy ratio of approximate solutions to maximization problems is usually computed as

$$r(s_a) = \frac{f(s^*)}{f(s_a)}$$

to make this ratio greater than or equal to 1, as it is for minimization problems.

Obviously, the closer $r(s_a)$ is to 1, the better the approximate solution is. For most instances, however, we cannot compute the accuracy ratio, because we typically do not know $f(s^*)$, the true optimal value of the objective function. Therefore, our hope should lie in obtaining a good upper bound on the values of $r(s_a)$. This leads to the following definitions.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/data_mining/dm_dti.htm

www.khanacademy.org/computing/computer-science/algorithm

www.tutorialspoint.com/design_and_analysis_of_algorithms/index.html

Important Books/Journals for further learning including the page nos.:

Data Structures and Algorithms

Design and Analysis of Algorithms

Course Teacher

Verified by HOD