**CS 3401- ALGORITHMS**

**UNIT I**

**INTRODUCTION**

**Algorithm analysis:** Time and space complexity - Asymptotic Notations and its properties Best case, Worst case and average case analysis – Recurrence relation: substitution method - Lower bounds – **searching:** linear search, binary search and Interpolation Search, **Pattern search:** The naïve string- matching algorithm - Rabin-Karp algorithm - Knuth-Morris-Pratt algorithm. **Sorting:** Insertion sort – heap sort

# PART - A

## What do you mean by algorithm?

An algorithm is a sequence of unambiguous for solving a problem i.e., for obtaining a required output for any legitimate input in a finite amount of time. In addition, all algorithms must satisfy the following criteria:

* 1. Input
  2. Output
  3. Definiteness
  4. Finiteness
  5. Effectiveness.

## What is performance measurement?

Performance measurement is concerned with obtaining the space and the time requirements of a particular algorithm.

## Give the diagram representation of Notion of algorithm.

PROBLEM

ALGORITHM

INPUT  OUTPUT

COMPUTER

## What are the types of algorithm efficiencies?

The two types of algorithm efficiencies are

Time efficiency: indicates how fast the algorithm runs.

Space efficiency: indicates how much extra memory the algorithm needs.

## What is space complexity?

Space Complexity indicates how much extra memory the algorithm needs. Basicallyit has three components. They are instruction space, data space and environment space.

## What is time complexity?

Time Complexity indicates how fast an algorithm runs. T (P) =Compile Time + Run Time.(Tp) ,Where Tp is no of add, sub, mul...

## What is an algorithm design technique?

An algorithm design technique is a general approach to solving problems algorithmically that is applicable to a variety of problems from different areas of computing.

## What is pseudo code?

A pseudo code is a mixture of a natural language and programming language constructs to specify an algorithm. A pseudo code is more precise than a natural language and its usage often yields more concise algorithm descriptions.

## What are the types of algorithm efficiencies?

The two types of algorithm efficiencies are

Time efficiency: indicates how fast the algorithm runs

Space efficiency: indicates how much extra memory the algorithm needs.

## What do you mean by “worst-case efficiency” of and algorithm?

The worst case efficiency of an algorithm, its efficiency for the worst-case input of size n, which is an input or inputs of size n for which the algorithm runs the longest among all possible inputs of that size.

## What is best-case efficiency?

The best-case efficiency of an algorithm is its efficiency for the best-case input of size n, which is an input or inputs for which the algorithm runs the fastest among all possible inputs of that size.

## What is average case efficiency?

The average case efficiency of an algorithm is its efficiency for an averagecase input of size n. It provides information about an algorithm behavior on a “typical” or “random” input.

## Define asymptotic notations. (R)

To choose best algorithm, we need to check efficiency of each algorithm the efficiency can be measured by computing time complexity of each algorithm. Asymptotic notation is shorthand way to represent the time complexity

The various notations are Big “Oh”, Big Omega, and Theta.

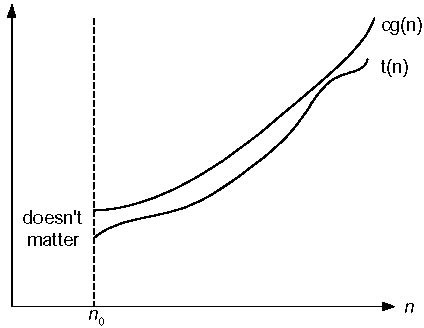
1. **Define the asymptotic notation “Big oh” (0)**

Let, f(n) and g(n) be two non-negative functions. Let, n0 and constant c are two integers such that no denotes some value of input similarly c is constant such that c > 0. We can write

F(n) <= c\*g(n) For all n ≥ n0

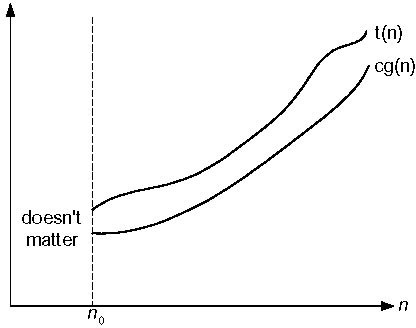
A function t(n) is said to be in O(g(n)) denoted t(n) ε O (g(n)), if t(n) is bounded above by some constant multiple of g(n) for all large n, i.e., if there exist some positive constant c and some non-negative integer n0 such that

T (n) < c g (n) for n > n0

****

## Define the asymptotic notation “Omega” ( Ω). (R)

A function f(n) is said to be in Ω (g(n)) if f(n) is bounded below by some positive constant multiple of g(n) such that

f(n) ≥ c \* g(n) For all n ≥ n0

**UNIT II**

# GRAPH ALGORITHMS

Graph algorithms: Representations of graphs - Graph traversal: DFS – BFS - applications

- Connectivity, strong connectivity, bi-connectivity - Minimum spanning tree: Kruskal’s and Prim’s algorithm- Shortest path: Bellman-Ford algorithm - Dijkstra’s algorithm - Floyd-Warshall algorithm Network flow: Flow networks - Ford-Fulkerson method – Matching: Maximum bipartite matching

## Define a graph. (April/May 2007) - U

A graph G=(V, E) consists of a set of vertices(V) and set of edges(E).

In general, A *graph* is a collection of *nodes* (also called *vertices*) and *edges* (also called

*arcs* or *links*) each connecting a pair of *nodes*.

V1

V2

**R**

V3

**Example:**

## What are the different types of Graph? -

There are two types of Graphs. They are:

1. Directed Graphs.
2. Undirected Graphs.

**Example for Directed Graph: Example for undirected Graph:**

## What is complete graph? - U

If an undirected graph of n vertices consists of n(n-1)/2 number of edges it is called as complete graph.

V1

V

V

V

## Example:

1. **What is sub graph? - U**

A subgraph G’ of graph G is a graph such that the set of vertices and set of edges of G’ are proper subset of the set of edges of G.

V2

V3

**Example:**

V1

V2

V3

**G Grapth G’ Sub Graph**

1. **What is connected graph? - U**

An directed graph is said to be connected if for every pair of distinct vertices Vi and Vj in V(G) there is a graph from Vi to Vj in G.

**Example:** V1 V2

V3 V4

## What are directed graphs? - U

Directed graph is a graph which consists of directed edges, where each edge in E is unidirectional. It is also referred as Digraph. If (v,w) is a directed edge then (v,w) ≠ (w,v

V1

V2

V3

(v1,v2) ≠ (v2,v1)

## What are undirected graphs? - U

An undirected graph is a graph, which consists of undirected edges. If (v,w) is an undirected edge then (v,w) = (w,v)

V

V

V

(v1,v2) = (v2,v1)

## Define cycle. - U

A cycle in a graph is a path in which first and last vertex are the same.

## Path is 1 -> 2 -> 3 -> 1

1 2

3

A graph which has a cycle is referred to as cyclic graph.

## Define Acyclic graph. - U

A directed graph is said to be acyclic when there is no cycle path in it. It is also called DAG(Directed Acyclic Graph).

## Example Path: A -> C -> D

A

B C

D E

## Define weighted graph. - U

A graph is said to be weighted graph if every edge in the graph is assigned a weight or value. It can be directed or undirected graph.

Ex: 1

1

2

3

1

1 1

2

3

2 1 2 1

## Define strongly connected graph. - U

A directed graph is said to be strongly connected, if for every pair of vertex, there exists a path

**Diagram**: V1 V2

V3



V4

V5

## What is weakly connected graph. (May/June 2012) - U

A digraph is weakly connected if all the vertices are connected to each other.

## Diagram:

V2

V4 V3



V5

V1

## Define path in a graph. - U

A **path** in a graph is a sequence of vertices such that from each of its vertices there is an edge to the next vertex in the sequence. A path may be infinite, but a finite path always has a first vertex, called its *start vertex*, and a last vertex, called its *end vertex*. Both of them are called *end or terminal vertices* of the path. The other vertices in the path are *internal vertices*. A **cycle** is a path such that the start vertex and end vertex are the same.

## What is the degree of a graph? - U

The number of edges incident on a vertex determines its degree. The degree of the vertex V is written as degree(V).

A

C

B

D

**Example:**

Degree (A) = 0 Degree (C) = 1 Degree (D) = 3

## Define indegree, outdegree in a graph. (April/May 2010) - U

The **indegree** is the numbers of edges entering in to the vertex V.

The **out degree** is the numbers of edges that are exiting or leaving from the vertex V.

V1 V2

Indegree(V1) = 1

Outdegree (V1) = 2

V3 V4

## What is adjacent node? - U

Adjacency node is the node which is connected to the other nodes in the graph.

**Example:**

A

C

E

B

D

**In the above diagram, C is an adjacent node to E**

1. **What are the applications of graphs? - R**

The various applications of graphs are:

* + In computer networking such LAN, WAN etc., internetworking of computer systems is done using graph concept.
  + In telephone cabling, graph theory is effectively used.
  + In job scheduling algorithms, the graph is used. Solution: a b c d

## Prove that the number of odd degree vertices in a connected graph should be even. (May/June 2007) - An



Consider a graph with odd degree vertices.



Degree =1

No. of vertices = 2 Degree = 3 No. of vertices = 4

## This implies that the no. of vertices is even for odd degree graphs.

1. **What is a single source shortest path problem? - U**

The single source shortest path problem finds the minimum cost from single source vertex to all other vertices. Dijkstra’s algorithm is used to solve this problem which follows the greedy technique.

## What is depth-first traversal? Give an example. -- U

**Depth-first search** (**DFS**) is a graph search algorithm that begins at the root and explores as far as possible along each branch before backtracking.

**Note**: Backtracking is finding a solution by trying one of several choices. If the choice proves incorrect, computation *backtracks* or restarts at the point of choice and tries another choice.

1

2

3

4

**Example:**

The Depth-first traversal is **1 – 2 – 4 – 3**

**UNIT III**

# ALGORITHM DESIGN TECHNIQUES

**Divide and Conquer methodology:** Finding maximum and minimum - Merge sort - Quick sort **Dynamic programming:** Elements of dynamic programming — Matrix-chain multiplication - Multi stage graph — Optimal Binary Search Trees. **Greedy Technique:** Elements of the greedy strategy - Activity-selection problem –- Optimal Merge pattern — Huffman Trees.

# PART A

## Give the general plan for divide-and-conquer algorithms.

## The general plan is as follows.

A problems instance is divided into several smaller instances of the same problem, ideally about the same size.

The smaller instances are solved, typically recursively.

If necessary the solutions obtained are combined to get the solution of the original problem.

* 1. **List the advantages of Divide and Conquer Algorithm.**

Solving difficult problems, Algorithm efficiency, Parallelism, Memory access, Round off control.

## Give the recurrence relation of divide-and-conquer? (R)

The recurrence relation is

*g*(*n*)



T(n) = *t*(*n*1)  *t*(*n*2)  .....  *t*(*nk*)  *f* (*n*) g(n)

* 1. **Define of feasibility.**

A feasible set (of candidates) is promising if it can be extended to produce not merely a solution, but an optimal solution to the problem.

* 1. **Define Quick Sort.**

Quick sort is an algorithm of choice in many situations because it is not difficult to implement, it is a good \"general purpose\" sort and it consumes relatively fewer resources during execution.

* 1. **List out the Disadvantages in Quick Sort**
     1. It is recursive. Especially if recursion is not available, the implementation is extremely complicated.
     2. It requires quadratic (i.e., n2) time in the worst-case.
     3. It is fragile i.e., a simple mistake in the implementation can go unnoticed and cause it to perform badly.
  2. **What is the difference between quicksort and mergesort?**

Both quicksort and mergesort use the divide-and-conquer technique in which the given array is partitioned into subarrays and solved. The difference lies in the technique that the arrays are partitioned. For mergesort the arrays are partitioned according to their position and in quicksort they are partitioned according to the element values.

## Define merge sort. (R)

1. Mergesort sorts a given array A[0..n-1] by dividing it into two halves a[0..(n/2)-1] and A[n/2..n-1] sorting each of them recursively and then merging the twosmaller sorted arrays into a single sorted one.
   1. **List the Steps in Merge Sort**

**Divide Step:** If given array A has zero or one element, return S; it is already sorted. Otherwise, divide A into two arrays, A1 and A2, each containing about half of the elements of A.

**Recursion Step:** Recursively sort array A1 and A2.

**Conquer Step:** Combine the elements back in A by merging the sorted arrays A1 and A2 into a sorted sequence

* 1. **List out Disadvantages of Divide and Conquer Algorithm**

Conceptual difficulty Recursion overhead Repeated subproblems

* 1. **List out the Advantages in Quick Sort**
* It is in-place since it uses only a small auxiliary stack.
* It requires only n log(n) time to sort n items.
* It has an extremely short inner loop
  1. **Define Optimal Binary Search Trees? (R)**

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. If probabilities of searching for elements of a set are known e.g., from accumulated data about past searches - it is natural to pose a question about an optimal binary search tree for which the average number of comparisons in a search is the smallest possible.

## List out the memory functions used under Dynamic programming. (MAY 2015) (R)

Memory functions solve in a top-down manner only sub problems that are necessary. Memory functions are an improvement of dynamic programming because they only solve sub problems that are necessary and do it only once. However they require more because it makes recursive calls which require additional memory.

* Greedy Algorithm
* Branch and Bound
* Genetic Algorithm

## State the general principle of greedy algorithm. (NOV/DEC 16) (R)

A greedy algorithm is an [algorithmic paradigm](https://en.wikipedia.org/wiki/Algorithmic_paradigm) that follows the [problem solving](https://en.wikipedia.org/wiki/Problem_solving) [heuristic](https://en.wikipedia.org/wiki/Heuristic_(computer_science)) of making the locally optimal choice at each stage with the hope of finding a [global optimum.](https://en.wikipedia.org/wiki/Global_optimum)

## Define Optimization function ?

Every set of s that satisfies the constraints is a feasible solution. Every feasible solution that maximizes is an optimal solution.

**UNIT IV**

# STATE SPACE SEARCH ALGORITHMS

Lower - Bound Arguments - P, NP NP- Complete and NP Hard Problems. Backtracking – n- Queen problem - Hamiltonian Circuit Problem – Subset Sum Problem. Branch and Bound – LIFO Search and FIFO search - Assignment problem – Knapsack Problem – Travelling Salesman Problem - Approximation Algorithms for NP-Hard Problems – Travelling Salesman problem – Knapsack problem.

# PART - A

## Differentiate backtracking and Exhaustive search. (AN)

|  |  |  |
| --- | --- | --- |
| S.No | Backtracking | Exhaustive search |
| 1. | Backtracking is to build up the solution vector one component at a time and to use modified criterion Function Pi(x1,..,xi) (sometimes called bounding function) to test whether the vector being formed has any chance of success. | Exhaustive search is simply a “brute – force” approach to combinatorial problems. It suggests generating each and every element of the problem’s domain, selecting those of them that satisfy the problem’s constraints, and  then finding a desired element. |
| 2. | Backtracking makes it possible to solve many large instances of NP- hard problems in an acceptable  amount of time. | Exhaustive search is impractical for large instances, however applicable for small instances of problems. |

1. **What are the factors that influence the efficiency of the backtracking algorithm?(APRIL/MAY 2008**) **(R)**

The efficiency of the backtracking algorithm depends on the following four factors. They are:

* 1. The time needed to generate the next xk
  2. The number of xk satisfying the explicit constraints.
  3. The time for the bounding functions Bk
  4. The number of xk satisfying the Bk.

## What is backtracking?(Nov/Dec 2011) (R)

Backtracking constructs solutions one component at a time and such partially constructed solutions are evaluated 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. .\_If there is no legitimate option for the next component, no alternatives for the remaining component need to be considered. In this case, the algorithm backtracks to replace the last component of the partially constructed solution with its next option.

## What is n-queens problem? (R)

The problem is to place ‘n’ queens on an n-by-n chessboard so thatno two queens attack each other by being in the same row or in the column orin the same

## Draw the solution for the 4-queen problem. (R)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Q |  |  |
|  |  |  | Q |
| Q |  |  |  |
|  |  | Q |  |

1. **Define the Hamiltonian cycle.(NOV/DEC 2012) (R)**

The Hamiltonian is defined as a cycle that passes through all the vertices of the graph exactly once. It is named after the Irish mathematician Sir William Rowan Hamilton (1805- 1865).It is a sequence of n+1 adjacentverticesvi0, vi1… vin-1, vi0 where the first vertex of the sequence is same as the last one while all the other n-1 vertices are distinct.

## What is the subset-sum problem?( Nov/Dec 2012) (R)

Find a subset of a given set S={s1,………,sn} of ‘n’ positive integers whose sum is equal to a given positive integer ‘d’.

## When can a node be terminated in the subset-sum problem?(NOV/DEC 2008) (R)

The sum of the numbers included are added and given as the value for the root as s’. The node can be terminated as a non-promising node if either of the two equalities holds:

s’+si+1>d (the sum s’ is too large)

*n*



*j* *i* \_ 1

*sj*  *d* (the sum s’ is too small)

## How can the output of a backtracking algorithm be thought of? (R)

The output of a backtracking algorithm can be thought of as an n-tuple(x1, …xn) where each coordinate xi is an element of some finite linearlyordered set Si. If such a tuple (x1, …xi) is not a solution, the algorithm finds thenext element in Si+1 that is consistent with the values of (x1, …xi) and theproblem’s constraints and adds it to the tuple as its (I+1)st coordinate. If suchan element does not exist, the algorithm backtracks to consider the next valueof xi, and so on.

## Give a template for a generic backtracking algorithm.(APRIL/MAY 2012) (R)

ALGORITHM Backtrack(X [1..i])

//Gives a template of a generic backtracking algorithm

//Input X[1..i] specifies the first I promising components of a solution

//Output All the tuples representing the problem’s solution if X[1..i] is a solution write X[1..i]

else

for each element x Si+1 consistent with X[1..i] and the constraints do X[i+1] = x

Backtrack(X[1..i+1])

## What is the method used to find the solution in n-queen problem by symmetry? (R)

The board of the n-queens problem has several symmetries so thatsome solutions can be obtained by other reflections. Placements in the lastn/2 columns need not be considered, because any solution with the first queen in square can be obtained by reflection from a solutionwith the first queen in square (1,n-i+1)

## State m color-ability decision problem.(NOV/DEC 2012) (R)

Let G be a graph and m be a given positive integer. We want to discover whether thenodes of G can be colored in such a way that no two adjacent nodes have the same color yet only m colors are used.

## What are the two types of constraints used in Backtracking? (R)

* Explicit constraints
* Implicit constraints

## Define implicit constraint.(APR/MAY 2010& 2012) (R)

They are rules that determine which of the tuples in the solution space of I satisfy the criteria function. It describes the way in which the xi must relate to each other.

## Define explicit constraint. (APR/MAY 2010& 2012) (R)

They are rules that restrict each xi to take on values only from a give set. They depend on the particular instance I of the problem being solved. All tuples that satisfy the explicit constraints define a possible solution space.

## What is a promising node in the state-space tree? (R) (Nov 17)

A node in a state-space tree is said to be promising if it corresponds toa partially constructed solution that may still lead to a complete solution.

## What is a non-promising node in the state-space tree? (R) (Nov 17)

A node in a state-space tree is said to be promising if it corresponds toa partially constructed solution that may still lead to a complete solution;

otherwise it is called non-promising

# UNIT V NP-COMPLETE AND APPROXIMATION ALGORITHM

Tractable and intractable problems: Polynomial time algorithms – Venn diagram representation - NP- algorithms - NP-hardness and NP-completeness – Bin Packing problem - Problem reduction: TSP – 3-CNF problem. **Approximation Algorithms**: TSP - **Randomized Algorithms**: concept and application - primality testing - randomized quick sort - Finding kth smallest number

**PART A**

## What are NP- hard and NP-complete problems? (R)

The problems whose solutions have computing times are bounded by polynomials of small degree.

## Define bounding. (R)

* + Branch-and-bound method searches a state space tree using any search mechanism in which all children of the E-node are generated before another node becomes the E-node.
  + Each answer node x has a cost c(x) and we have to find a minimum-cost answer node. Common strategies include LC, FIFO, and LIFO.
  + Use a cost function ˆc(·) such that ˆc(x)  c(x) provides lower bound on the solution

obtainable from any node x.

## List example of NP hard problem. (R)

NP hard graph problem

* + clique decision problem(CDP)
  + Node cover decision problem(NCDP).

## What is meant by NP hard and NP complete problem?(NOV/DEC 2011&NOV/DEC 2012) (R)

* + NP-Hard Problem: A problem L is NP-hard if any only if satisfy ability reduces to L.
  + NP- Complete: A problem L is NP-complete if and only if L is NP-hard and L є NP. There are NP-hard problems that are not NP-complete.

Halting problem is NP-hard decision problem, but it is not NP-complete.

## How NP-Hard problems are different from NP-Complete? (R)

These are the problems that are even harder than the NP-complete problems. Note that NP-hard problems do not have to be in NP, and they do not have to be decision problems.

The precise definition here is that a problem X is NP-hard, if there is an NP- complete problem Y, such that Y is reducible to X in polynomial time.

But since any NP-complete problem can be reduced to any other NP-complete problem in polynomial time, all NP-complete problems can be reduced to any NP-hard problem in polynomial time. Then, if there is a solution to one NP-hard problem in polynomial time, there is a solution to all NP problems in polynomial time.

## Define P and NP problems. (APR/MAY 2017) (R)

**P**- Polynomial time **solving** . Problems which can be solved in polynomial time, which take time like O(n), O(n2), O(n3). Eg: finding maximum element in an array or to check whether a string is palindrome or not. so there are many problems which can be solved in polynomial time.

**NP**- Non deterministic Polynomial time solving. Problem which can't be solved in polynomial time like TSP( travelling salesman problem)

## What are tractable and non-tractable problems? (APR/MAY 2018)

Generally we think of problems that are solvable by polynomial time algorithms as being tractable, and problems that require super polynomial time as being intractable.

## Define P and NP problems. (NOV/DEC 2018)

All problems in P can be solved with polynomial time algorithms, whereas all problems in NP - P are intractable.

It is not known whether P = NP. However, many problems are known in NP with the property that if they belong to P, then it can be proved that P = NP.

If P ≠ NP, there are problems in NP that are neither in P nor in NP-Complete.

The problem belongs to class P if it’s easy to find a solution for the problem. The problem belongs to NP, if it’s easy to check a solution that may have been very tedious to find.

## Define NP completeness and NP hard.(APR/MAY 2019)

NP-complete problems are the hardest problems in NP set. A decision problem L is NP- complete if:

1. L is in NP (Any given solution for NP-complete problems can be verified quickly, but

there is no efficient known solution).

1. Every problem in NP is reducible to L in polynomial time (Reduction is defined below). A problem is NP-Hard if it follows property 2 mentioned above, doesn’t need to follow property 1. Therefore, NP-Complete set is also a subset of NP-Hard set.

## What do you meant by primality testing?

The basic structure of randomized primality tests is as follows:

* Randomly pick a number a.
* Check equality (corresponding to the chosen test) involving a and the given number n. If the equality fails to hold true, then n is a composite number and a is a witness for the compositeness, and the test stops.
* Get back to the step one until the required accuracy is reached.

After one or more iterations, if n is not found to be a composite number, then it can be declared [probably prime.](https://en.wikipedia.org/wiki/Probable_prime)

## What is Kth smallest number?

Given an array and a number k where k is smaller than the size of the array, we need to find the k’th smallest element in the given array. It is given that all array elements are distinct.

## How quick sort using random pivoting?

In QuickSort we first partition the array in place such that all elements to the left of the pivot element are smaller, while all elements to the right of the pivot are greater than the pivot.

Then we recursively call the same procedure for left and right subarrays.

Unlike [merge sort](https://www.geeksforgeeks.org/merge-sort/), we don’t need to merge the two sorted arrays. Thus Quicksort requires lesser auxiliary space than Merge Sort, which is why it is often preferred to Merge Sort. Using a randomly generated pivot we can further improve the time complexity of QuickSort.