Page No.
Date

Q. 13 → Job sequencing problem with deadline :

This problem consist of n jobs each associated with a deadline & profit and our objective is to earn maximum profit we will earn profit only when job is completed on or before deadline we assume that each job will take unit time to complete.

- In this problem we have n jobs $J_1, J_2, J_3 \dots J_n$ each has an associated deadline $d_1, d_2, d_3 \dots d_n$ and profit $P_1, P_2 \dots P_n$
- profit will only be awarded or earned if the job is completed on or before the deadline.
- we assume that each job takes unit time to complete.
- The objective is to earn maximum profit when only one job can be scheduled or processed at any given time.

Given:

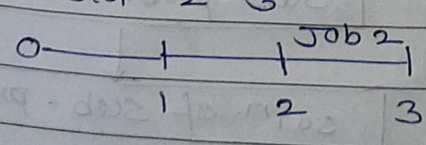
Job ID	1	2	3	4	5
Deadline	2	3	2	1	3
profit	20	38	16	10	30

The sort in maximum profit

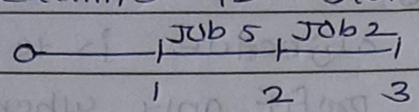
Job ID	2	5	1	3	4
deadline	3	3	2	2	1
profit	38	30	20	16	10

sort in decreasing order profit

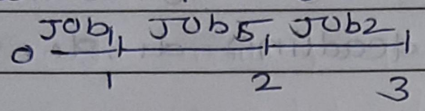
1) The last empty slot available for job 2 before deadline is slot 2 - 3



2) This last empty slot available for Job 5 before deadline is slot 1 – 2



3) The last empty slot available for Job 1 before deadline is slot 0 – 1



All the slot are full the sequence of job 1:5:2

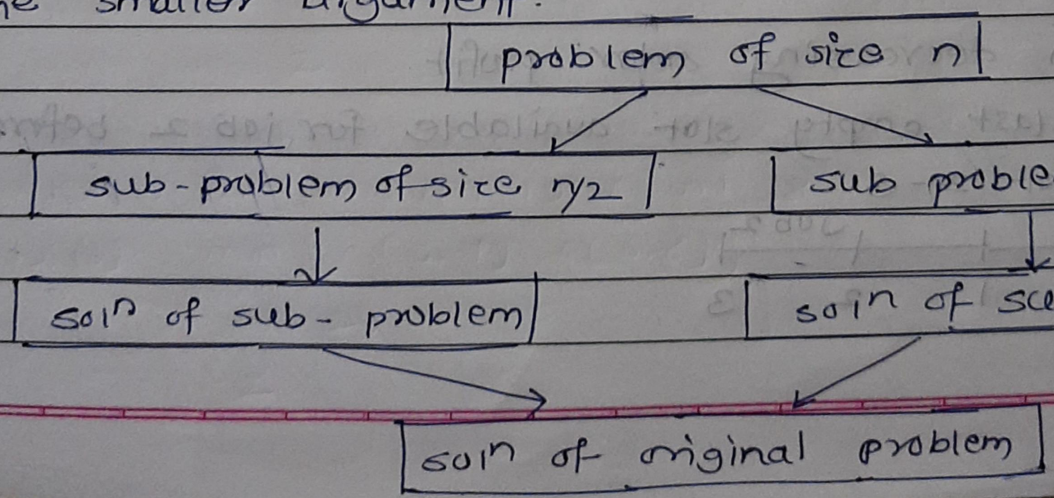
Q. 9) Explain short note on divide and conquer method?

→ Divide and conquer method: This is the most widely applicable technique for designing efficient algorithms. It works in three stages as shown in fig

1) Divide – Recurssively divide the bigger problem of size n into smaller sub-problem of size n/2

2) solve – sub-problem are solved independently

3) Combine – combine solution of smaller sub-problem with arguments. Hence such problem can be solved easily using recursion. Divide & conquer is multi-branched. Top-down recursive approach. Each branch indicates one sub-problem and it calls itself with the smaller argument.



above diagg. show the working of divide & conquer method. sub-problem may or may not be of size n/2

Q.22) Explain multistage graph problem with example?

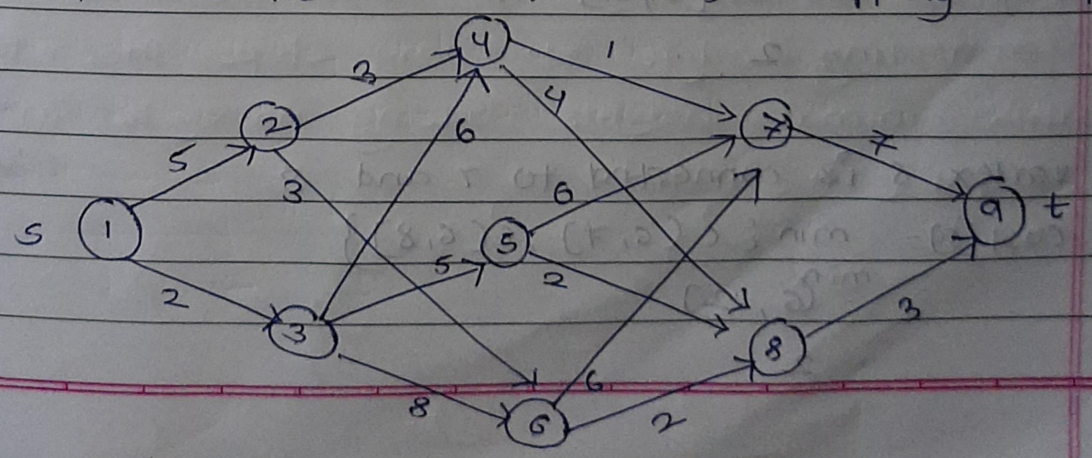
→ 1) A mutistage graph is a directed graph in which the nodes can be divided into set of stages such that all edges are from a stage to next stage only in other words there is no edge betn vertices of same stage and from a vertex of current stage to previous stage

2) The brute force method of finding all possible path betn source & Destination and then finding the minimum That's the worst possible strategy.

3) Dijkstra's algorithm of single source shorted path. This method will find shortest path from source to all other nodes which is not required in this case. so it will take a lot of time & it doesn't even use the special feature that this multi-stage graph has.

4) simple Greedy method - At each node, choose the shortest outgoing path. if we apply this approach to the example graph give above we get soln as 1+4+18 = 23, But a quick look at the graph will show much shorter path available than 23, so the greedy method fails. The best option is Dynamic programming so, we need to find optimal sub-structure. Recursive equation & overlapping sub-problem.

e.g.



Stage - 1:

vertex 1 is connected to 2 & 3

$cost[1] = \min\{c[1,2], c[1,3]\}$

$= \min\{5, 2\}$

$= 2$

stage - 2:

vertex 2 is connected to 4 and 6

$cost[2] = \min\{c[2,4], c[2,6]\}$

$= \min\{3, 8\}$

$= 3$

vertex 3 is connected to 4, 5, and 6

$cost[3] = \min\{c[3,4], c[3,5], c[3,6]\}$

$= \min\{6, 5, 8\}$

$= 5$

stage 3:

vertex 4 is connected 7 and 8

$cost[4] = \min\{c[4,7], c[4,8]\}$

$= \min[1, 4]$

$= 1$

vertex 5 is connected to 7 and 8

$cost[5] = \min\{c[5,7], c[5,8]\}$

$= \min[6, 2]$

$= 2$

vertex 6 is connected to 7 and 8

$cost[6] = \min\{c[6,7], c[6,8]\}$

$= \min[6, 2]$

$= 2$

stage - 4:

vertex 7 is connected to 9

cost [7] = min {c[7,9]

= min 7

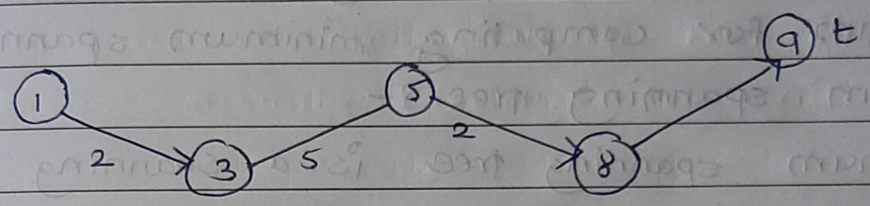
= 7

cost [8] = min {c[8,9]

= min 3

= 3.

min cost path s to t



Q.5 write a short note on Rabin karp algorithm

→ 1) It is based on hashing technique.

2) It first compute the hash value of pattern and text.

3) If has values are same i.e if hash(s) = hash(t). we check each charcter if character are same pattern is found. if hash value are not same no need of comparing string.

3) string are compared using brute force approach. if pattern is found then it is called as Hit. otherwise it is called spurious Hit. Time complexity = O(n) For worst case sometime it is O(m.n) when prime no. is used very small.

Algorithm Rabin-karp (T, P)

n = T. length

m = P. length

h^p = hash(T

h^t = hash (T) (0 ... m-1)

for s = 0 to n-m

if (h^p = h^t)

if (P (0 ... m-1) == T (0 ... m-1))

print "pattern found"

if (s < n-m)

h^t = hash (s+1 .... s+m-1)

Q. 23 what is minimum spanning tree? Explain prim's algorithm for computing minimum spanning tree

→ minimum spanning tree :-

Miminum spanning tree is a spanning tree with minimum cost.

prims algorithm :-

prims algorithm is a greedy algorithm that is used to find the minimum spanning tree from a graph. prims algorithm finds the subset of edges that includes every vertex of the graph such that the sum of weights of the edges can be minimized.

prims algorithm start with the single node & explores all the adjacent nodes with all the connecting edges at every steps. The edges with the minimal weights causing no cycles in the graph got selected.

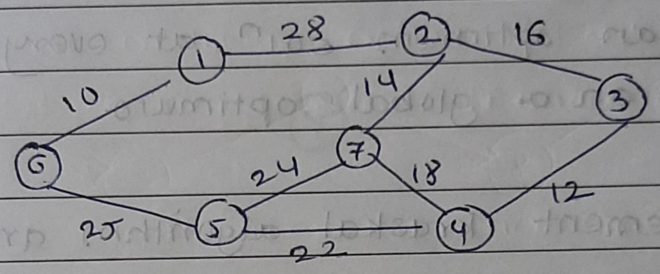
prim's algorithm are given steps are follows:

1) first we have to initalize an MST with randomly choose vertex.

2) Now, we have to find all the edges that connect the tree in the above step with the new vertices. from the edges found. select the minimum edges and add it to the tree.

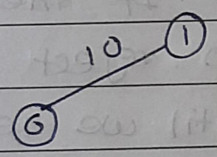
3) Repeat step 2 until the minimum spanning tree is formed.

e.g.

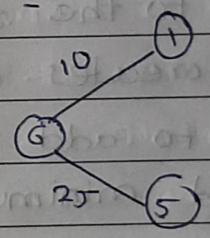


Sol^n

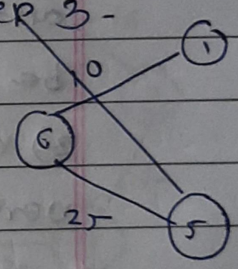
step - 1



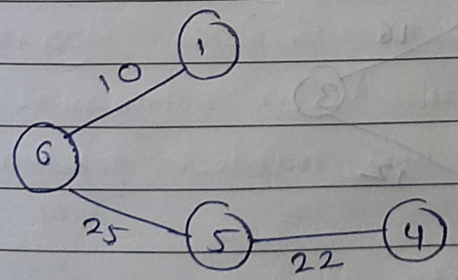
step 2 -



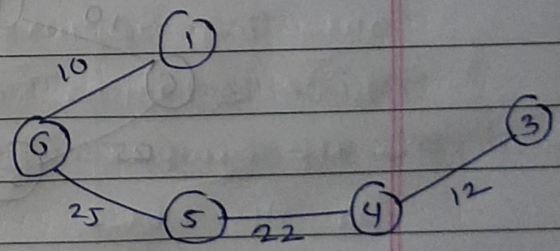
step 3 -



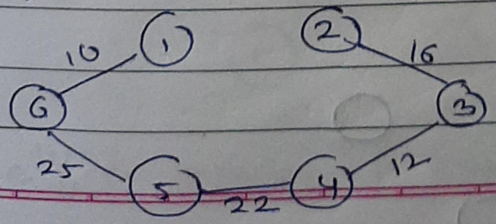
step - 3



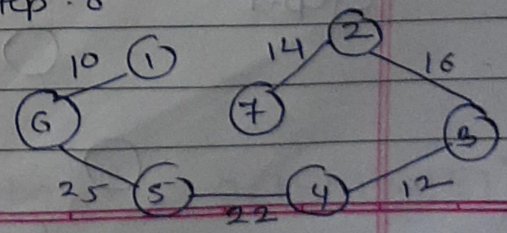
step - 4



step - 5



step - 6



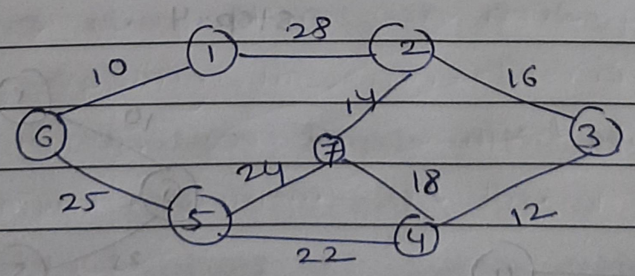
Now cost of minimum spanning tree -
= sum of all edges weight
= 10 + 25 + 22 + 22 + 16 + 14 = 99 units.

Q. 19) what the kruskal algorithm for minimum spanning tree. what is complexity of kruskal algorithm.

→ Kruskal algorithm is used to find the minimum spanning tree for a connected weighted graph. The main target of the algorithm is to find the subset of edges by using which we can traverse every vertex. of the graph. It follow the greedy approach that finds an optimum soln at every stage instead of focusing on a global optimum.

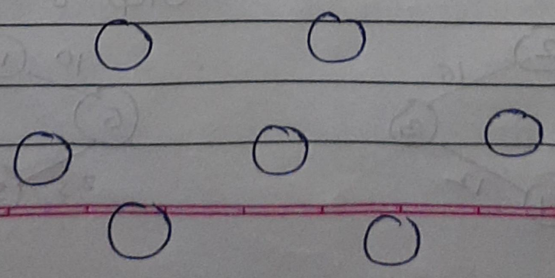
step to implement kruskal algorithm are listed:
1) first sort all the edges from low weight to high
2) Now, take the edge with the lowest weight and add it to the spanning tree. If the edge to be added creates a cycle then reject the edge.
3) continue to add the edges until we reach all vertices and minimum spanning tree is created.

e.g.

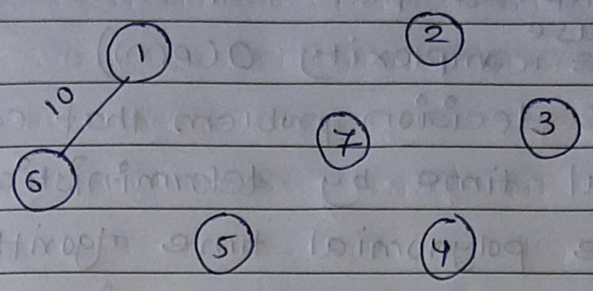


soln:

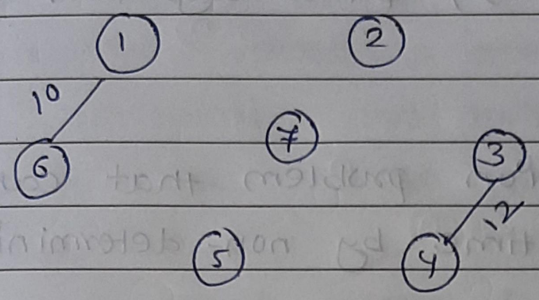
step-1



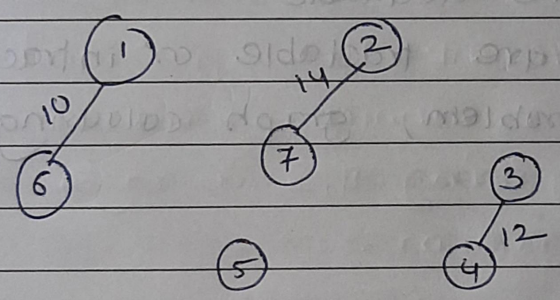
step - 2



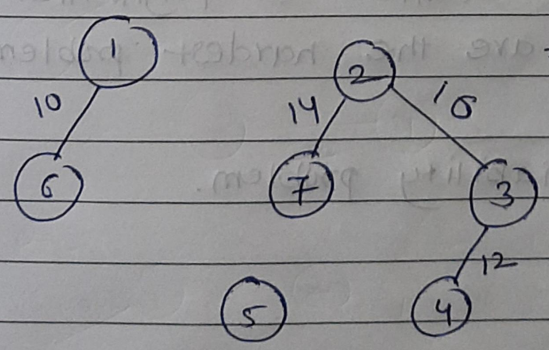
step - 3



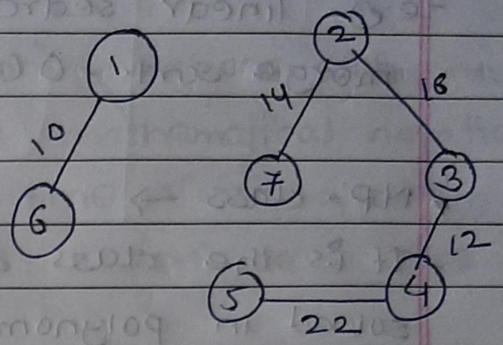
step - 4



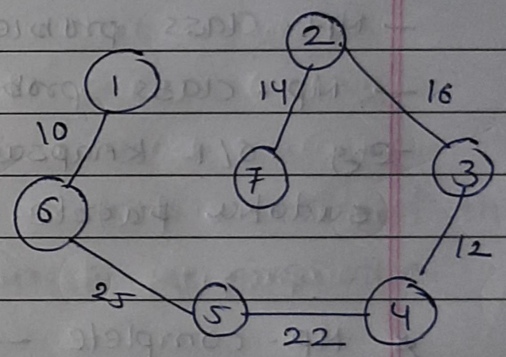
step - 5



step - 6



step - 7



since all the vertices have been connected in MST.

= sum of all edges weights

= 10 + 25 + 22 + 12 + 16 + 14 = 99 unit.

The time complexity of kruskal algorithm is $O(|E| \log |E|)$

Q.10) Define P, NP, NP-complete, NP-Hard.

→ * p-class → ~~time~~ worst case complexity $O(P(n))$

- It is the class of decision problem that can be solved in the polynomial time by deterministic algorithm
- P stands for the polynomial-time algorithm.
- p-class problem are decidable & tractable problem
- e.g. linear search $O(n)$, binary search $O(\log n)$, merge sort - $O(n \log n)$, prims algorithm $O(n^2)$

* NP-class →

- It is the class decision problem that can be solved in polynomial time by non-deterministic algorithm.
- NP stands for Non-deterministic polynomial time algorithms.
- NP-Class problem are decidable
- NP-class problem are tractable or intractable.
- e.g 0/1 knapsack problem, graph colouring problem, sudoku puzzle.

* NP-complete →

- NP-complete problem can have possible solution only by appling the non-deterministic algorithm & their solution can be verified in polynomial time.
- NP-complete problem are the hardest problems in NP-class.
- e.g. Cirait satisfiafiability problem.

- NP - Hard class -

- A decision problem P_1 is NP - Hard if each NP - class problem is polynomially reducible to P_1
- All NP - complete problem are NP - Hard problem but all NP - Hard problem are not NP - complete problem.
- NP - Hard problem are decidable or undecidable
- e.g. vertex - cover decision problem.

Q. 15) Explain the different asymptotic notations with graphs.

→ Asymptotic notations are the mathematical notations used to describe the running time of an algorithm when the input tends towards a particular value or a limiting value.

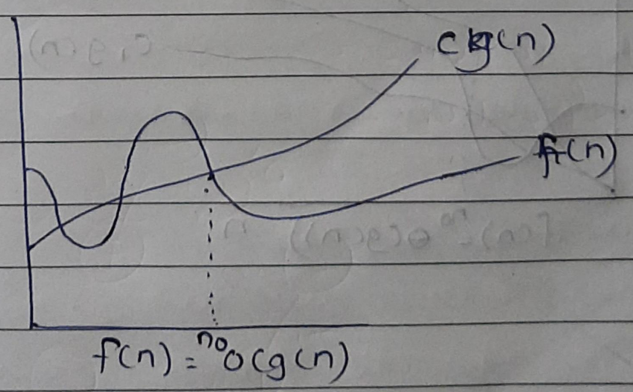
There are three types of notations:

1) Big - o - notation
2) Omega notation
3) Theta notation.

1) Big-o notation -

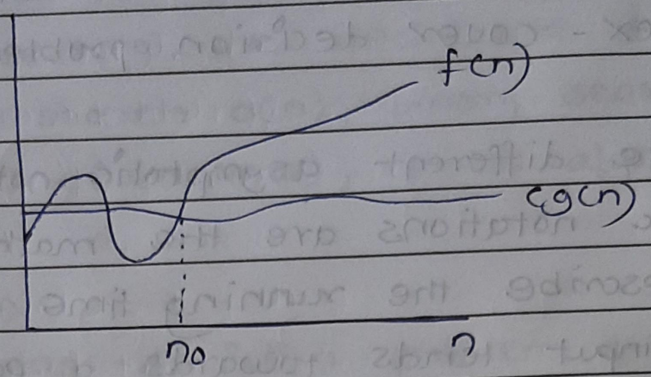
Big - o - notation represent the upper bound of the running time of an algorithm, thus, it gives the worst - case complexity of an algorithm.

The Funⁿ - $h(n) = O(k(n))$



2) Omega Notation

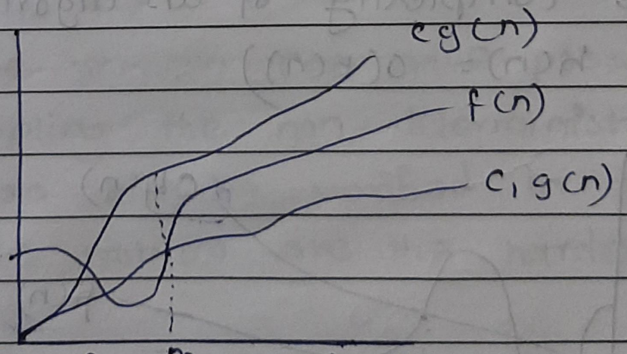
omega notation represent the lower bound of the running time of an algorithm, Thus, it provides the best case complexity of an algorithm.



$f(n) = \Omega(g(n))$

3) Theta Notation-

Theta notation enclose the Funⁿ from above & below, since, it represent the upper and the lower bound of the running time of an algorithm it is used for analysiring the average-case complexity of an algorithm.



$f(n) = \Theta(g(n))$

Q. 32) Explain assembly line scheduling problem with example.

→ 1) Assembly line scheduling is a manufacturing problem. In automobile industries assembly lines are used to transfer parts from one station to another station

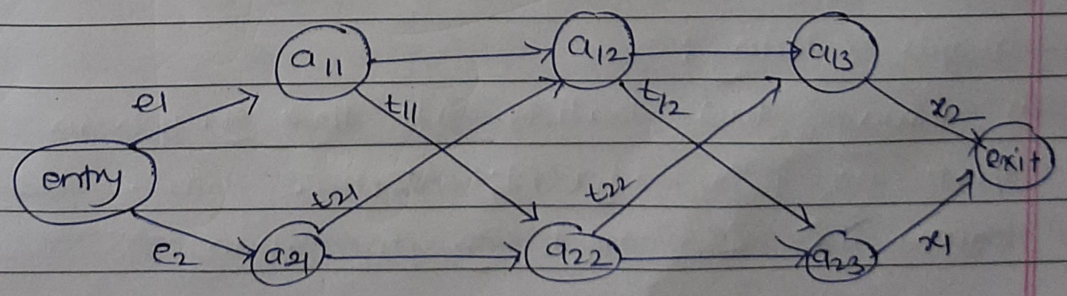
2) Manufacturing of large items like car, trucks, etc. generally undergoes through multiple station. where each station is responsible for assembling particular part only. Entire product be ready after it goes through predefined in stations in sequence.

3) Manufacturing of car may be done through several stages like engine fitting, coloring, light fitting fixing of controlling system. gates, seats and many other things

4) The particular take is carried out at the station dedicated to that task only. Based on the requirement there may be more than one assembly line.

5) In case of two assembly lines if the load at station j at assembly 1 is very high, then components are transfer to station j of assembly line 2 the converse is also true. This technique helps to speed ups the manufacturing process.

5) The time to transfer partial product from one station to next station on the same assembly line a regligible. During rush factory may transfer partially completed. auto from one assembly line to another, complete the manufactering as quickly as possible.



Q.24) write algorithm for binary search. Explain the algorithm with example.

→ Algorithm BINARY_SEARCH (A, key)

```
low ← 1
high ← n
while low < high do
 mid ← (low + high)/2
 if A[mid] == key then
 return mid
else if A[mid] < key then
  low ← mid +1
else
 hig ← mid-1
end
end
return 0
```

Q.28) write short note on Bellman-Ford Algorithm.

→ 1) like Dijkstra's algorithm, Bellman-Ford algorithm also follow the optimal substructure property of the shortest path & the concept of relaxation.

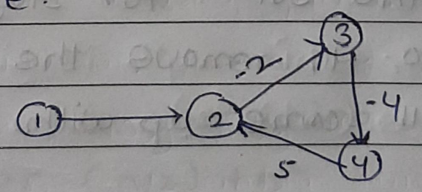
2) The Bellman-ford algorithm solves the single source shortest path problem using dynamic programming.

3) It check for the principle of optimality, relaxes the edges to determine the nearest neighbour and then estimates the shortest path from a single source to all other nodes in a given graph.

4) It can solve the single source shortest problem for a graph with negative-weight edges. It can detect a negative weight cycle in a given graph but cannot handle it.

5) A negative weight cycle in a graph defines a set of edges that form a cycle and the sum of their weights results in a negative value.

6) consider a directed weighted graph depicted in a cycle formed by the nodes 2 → 3 → 4 has resultant length = -2 + (-4) + 5 = -1 so it is negative weight cycle.



7) since the objective of an algorithm is to find the shortest path, the algorithm moves around the negative weight cycle for the infinite no. of times to reduce the length to $-\infty$. Thus the algorithm does not terminate for a graph with a negative weight cycle.

8) Therefore the Bellman-Ford algorithm cannot ~~give~~ solve the single source shortest path problem if the graph consist of negative weight cycle, but it identifies the presence of a negative-weight cycle in a given graph.

Q. 14) What is backtracking? Explain how it is applicable to graph coloring problem?

→ Backtracking approach

1) Backtracking is the process where the entire problem is divide into several stages. The algorithm then attempt to find the soln to the problem by constructing partial soln remain consistent with the requirements of the problem.

2) However when an inconsistency with the requirement of the problem occurs the algorithm backs up by removing the most recently constructed part of the soln and trying another possibility.

3) For e.g. algorithm will first find soln for stage one & stage two and so on till stage N. However if it cannot find the soln for stage N+1, then it will go back to N, remove the soln it found for stage N & will come up with a new soln for stage N.

4) e.g - Graph coloring & N-Queens problems.

Graph coloring:

1) coloring the vertices of the graph in such a way that no two adjacent vertices have the same color. This is called graph coloring.

2) It is also called as vertex coloring or k-coloring
3) The smallest no. of color required for coloring the graph is called as chromatic no.
4) e.g.

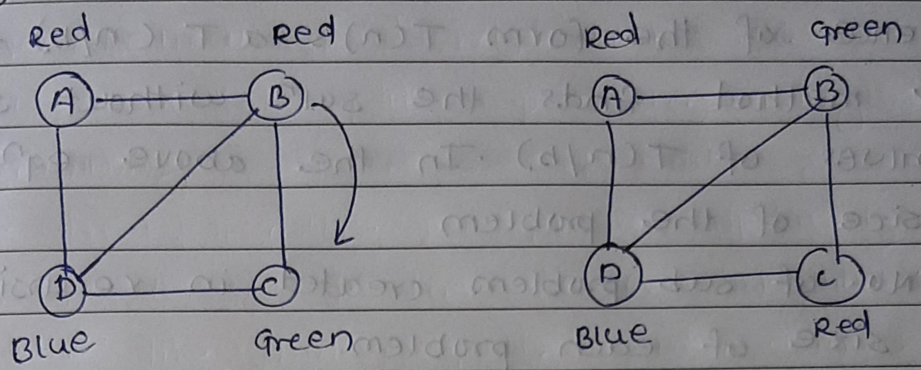


fig A fig B

In fig A same color (Red) is assigned to adjacent vertices (A & B) which is against rule so we Backtrack & change the color as shown in fig B. we cannot assign another color on the place of Red because chromatic color should be minimum. If we assign another color on the place of Red chromatic no. will be 4 for fig. A. and if we see fig B. chromatic no. is 3 which is minimum fig B is Right soln for graph coloring.

Q.25) solve the following using master method.
i) $T(n) = 8T(n/2) + n^2$
ii) $T(n) = 4T(n/2) + n \log n$.

Definition -
Divide & conquer strategy uses recursion. The time complexity of the recursive program is describe using recurrence. The master theorem

is often suitable to find the time complexity of recursive problem.

- Master method is used to quickly solve the recurrence of the form $T(n) = a \cdot T(n/b) + f(n)$. Master method finds the solⁿ without substituting the values of $T(n/b)$. In the above eqⁿ

n = size of the problem

a = No. of sub problem created in recursive solⁿ

n/b = size of each problem.

$f(n)$ = work done outside recursive call.

This include cost of division of problem & merging of the solⁿ

Ex.

i) Given

$T(n) = 8T(n/2) + n^2$

compare this eqⁿ with $T(n) = aT(n/b) + f(n)$

Here $a = 8$ $b = 2$ and $f(n) = n^2$

chuking the relation betⁿ a and b^2

As $a > b^2$

i.e. $8 > 2^2$, solⁿ for this eqⁿ is given as

$T(n) = (n^{\log_b a}) = \Theta(n^{\log_2 8}) = \Theta(n^{\log_2 2^3}) = \Theta(n^{3\log_2 2}) = \Theta(n^3)$

ii) $T(n) = 4T(n/2)\, n\log n$

Q. 26) Explain the diff. betn greedy approach & dynamic programming approach.

Greedy approach	Dynamic approach
1) Greedy method does not gurantee an optimal soln	1) Dynamic programming guarn. tees an optimal soln
2) sub-problems do not overlap	2) sub-problems overlap
3) It does little work	3) It does more work
4) only considers the current choices	4) consider the future choice.
5) construct the soln from the set of feasible soln	5) There is no spealized set of feasible choices
6) select choice which is locally optimum.	6) select choices which is globally optimum.
7) There is no concept of memorization.	7) Employ memorization
8) e.g. knapsack problem, Job scheduling with deadline, kruskal, prims.	8) e.g. All pair shortest path, 0/1 knapsack, travelling saleman problem, LCS. (longest common sequence)