International Islamic University Chittagong

Department of Computer Science and Engineering

B. Sc. in CSE Final Examination, Autumn 2018

Course Code: CSE 2403 **Course Title: Computer Algorithms**

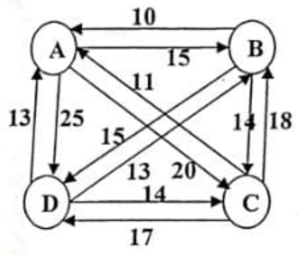
*Total marks: **50*** *Time: 2 hours 30 minutes*

[Answer any ***two*** questions from **Group-A** and any ***three*** questions from **Group-B**;
Separate answer script must be used for Group-A and Group-B;
Figures in the right hand margin indicate full marks.]

Group-A

1.

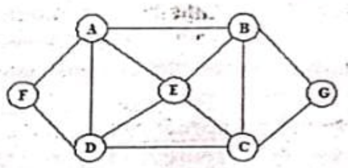
a) Define traveling salesperson problem. Apply the traveling salesperson problem on the following figure to find the optimal tour. **4**



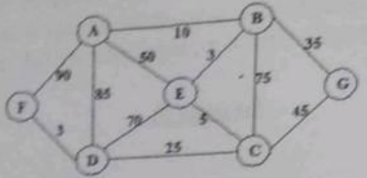
b) What is an optimal Huffman code for the following set of frequencies? **3**
a:5 b:8 c:45 d:25 e:17 f:30

c) Prove that in activity selection problem, the activity that finishes first is always part of some optimal solution. **3**

2. a) Show how Depth First Search works on following graph if E is the source **3**



b) Define spanning tree and minimum spanning tree. Construct minimum spanning tree from the graph using Kruskal Algorithm if E is the source. 4



c) In what order do the vertices will be visited if the graph in **Figure 1** is visited using BFS and DFS. Start from vertex ***G*** and assume that the adjacency list is in lexicographic order. You only need to name the vertices. 3

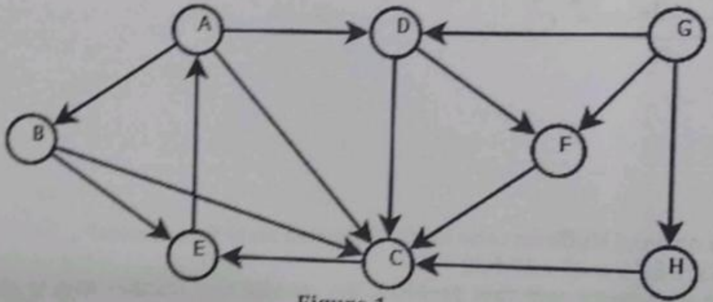


Figure 1

3.

a) Consider the graph with the following weights: w(a, b) = 1, w(a, c) = 2 , w(a, d) = 6, w(a, e) = 9, w(b,c)= 3, w(b, d) = 4, w(b, e) = 10, w(c, d) = 5, w(c, e) = 8, w(d, e) = 7. 5

a) What minimum spanning tree would *Kruskal*'s algorithm produce? Write the edges in the order that the algorithm would add them to its result.

b) What minimum spanning tree would *Prim*'s algorithm produce, starting at vertex b? Write the edges in the order that the algorithm would add them to its result.

b) Describe activity-selection problem. Find all possible optimal set for the following data set using activity selection problem: 5

i	1	2	3	4	5	6	7	8	9	10	11
S_i	1	3	0	5	3	5	6	8	8	2	12
f_i	4	5	6	7	8	9	10	11	12	13	14

Group-B

4. a) What are the four variants of shortest path algorithm? Please describe. 3

b) Illustrate the operation of *Dijkstra's algorithm* for finding shortest path on the directed graph shown in **Figure 2**. Assume **a** as the source. Write down the **d** and π value of each vertex after each iteration. 5

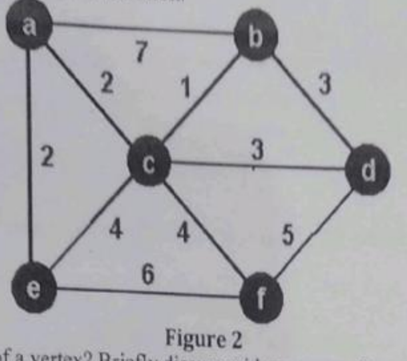


Figure 2

c) What is *relaxation* of a vertex? Briefly discuss with necessary figure. 2

5.

a) Using suitable example show how *branch-and-bound* technique can be applied to solve *travelling salesman problem.* 4

b) What is the significance of **NP-Complete** class in complexity theory? 3

c) Use *recurrence tree* to solve the following recurrence $T(n) = 3T(n/3) + cn^2$. 3

6.

a) Write down Graham's scan algorithm of convex hull construction 3

b) Write down the algorithm to calculate x^n mod m. 4

c) Graham's scan solves the convex hull problem by sorting the points using polar angle with respect to some point P_0 in counter clockwise. Suppose, you are given two points $P_1(4,5)$ and $P_2(2,3)$. Determine which one has greater polar angle with respect to point $P_0(0,0)$. 3

7.

a) What are the applications of Branch and Bound? 4

b) Define convex hull. Consider the following points and find convex hull using Graham's scan algorithm: 3

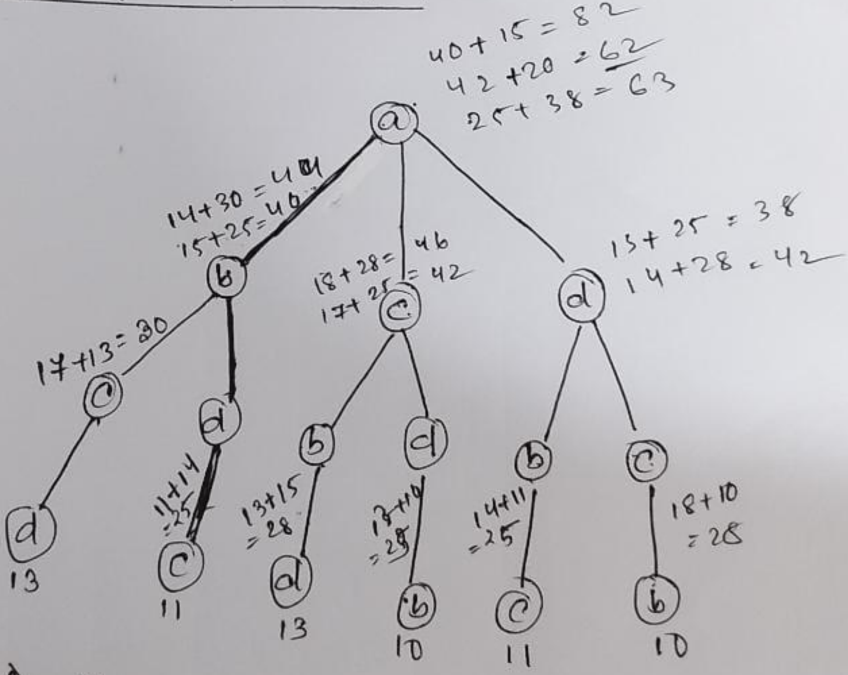
P1(0.7, 2.7), P2(1.8, 3.2), P3(2.6, 0.8), P4(0.9, 2.5), P5(0.6, 0.7), P6(1.5, 0.6), P7(1.5, 2.5), P8(2, 1.5), P9(1.0, 2.0), P10(0.8, 1.5).

c) State for each of the following algorithms whether they are *greedy algorithm* or *dynamic programming (DP) algorithm* or *neither greedy nor DP*. 3

i) Bellman-Ford's algorithm for single-source shortest path
ii) Dijkstra's algorithm for single-source shortest path
iii) Floyd-Warshall's algorithm for all-pairs shortest path
iv) Graham's scan algorithm for finding convex hull
v) Kruskal's algorithm for finding minimum spanning tree
vi) Huffman's algorithm for finding optimal prefix code

1 (a)

	a	b	c	d
a	0	15	20	25
b	10	0	14	15
c	11	18	0	17
d	13	13	14	0



$g(b, \phi) = 10$

$g(c, \phi) = 11$

$g(d, \phi) = 13$

$g(b, \{c\}) = \cancel{30}\ 25$

$g(b, \{b\}) = 28$

$g(c, b) = 28$

$g(c, d) = 30$

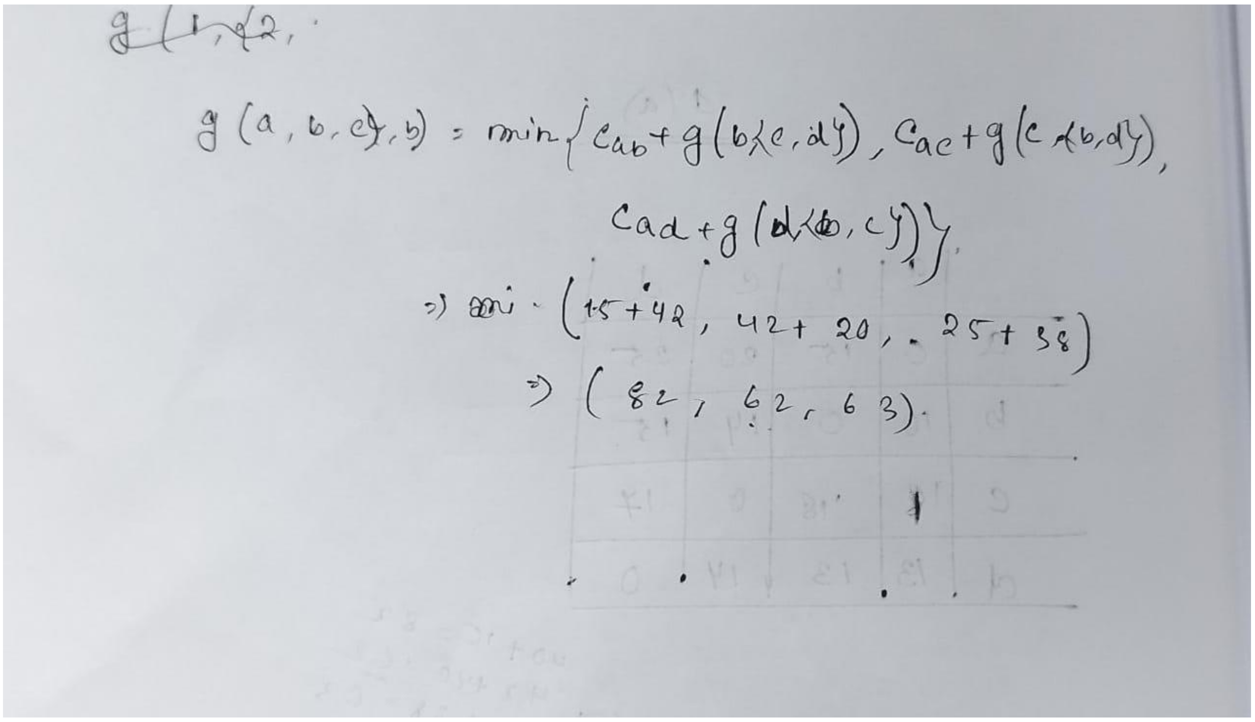
$g(d, b) = 25$

$g(d, c) = 25$

$g(b, d, c) = 40$

$g(c, d, b) = 42$

$g(d, b, c) = 38$



~~g (1, {2,~~

$g(a, \{b, c, d\}) = \min\{c_{ab} + g(b, \{c, d\}),\ c_{ac} + g(c, \{b, d\}),$

$c_{ad} + g(d, \{b, c\})\}$

$\Rightarrow \min\,(15 + 42,\ 42 + 20,\ 25 + 38)$

$\Rightarrow (82,\ 62,\ 63)$

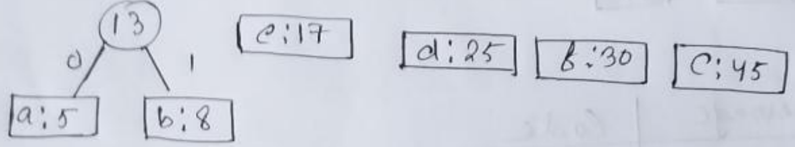
1 (b)

$\boxed{a:5}$ $\boxed{b:8}$ $\boxed{c:45}$ $\boxed{d:25}$ $\boxed{e:17}$ $\boxed{f:30}$

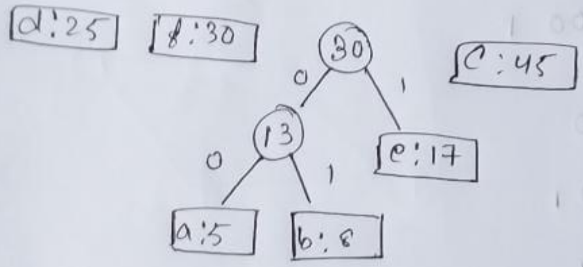
1

$\boxed{a:5}$ $\boxed{b:8}$ $\boxed{e:17}$ $\boxed{d:25}$ $\boxed{f:30}$ $\boxed{c:45}$

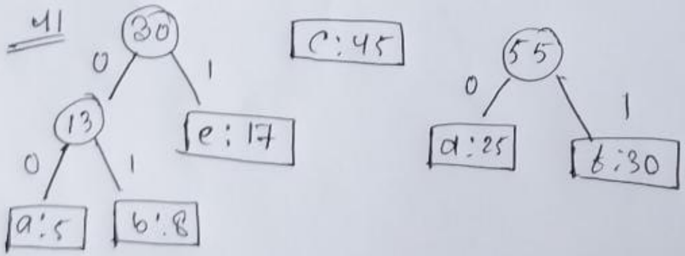
2



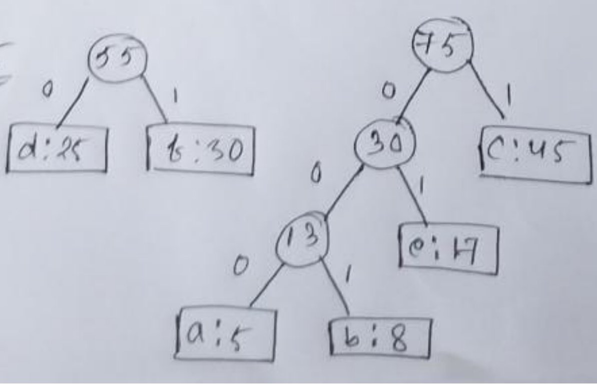
3



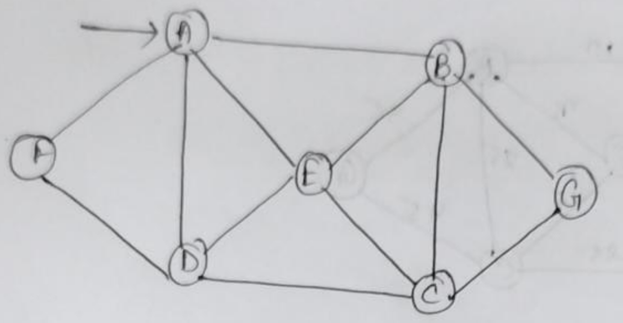
4



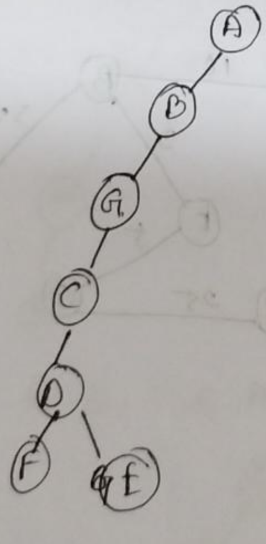
5

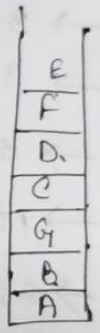


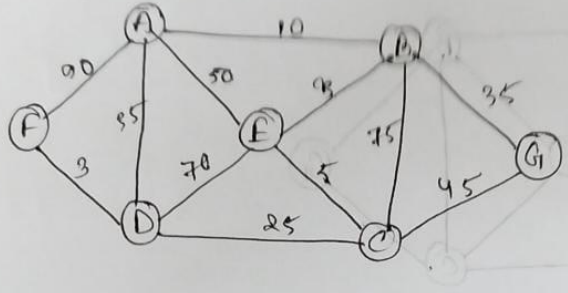
2(a)



DFS :- A, B, G, C, D, F, E







$BE \rightarrow 3$

$FD \rightarrow 3$

$CE \rightarrow 5$

$AB \rightarrow 10$

$DC \rightarrow 25$

$AD \rightarrow 35$

$BG \rightarrow 35$

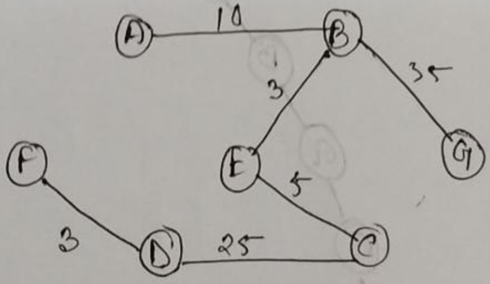
$CG \rightarrow 45$

$AE \rightarrow 50$

$DE \rightarrow 70$

$BC \rightarrow 75$

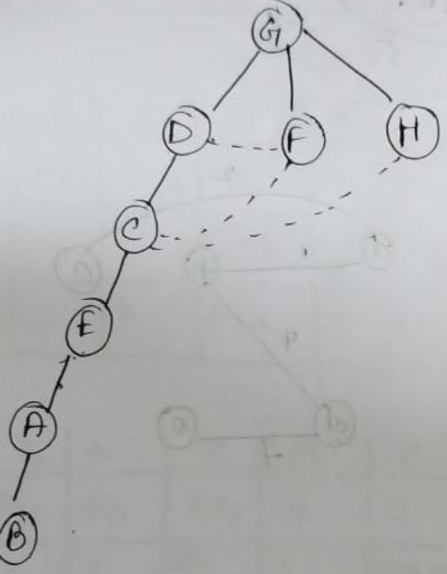
$AF \rightarrow 90$



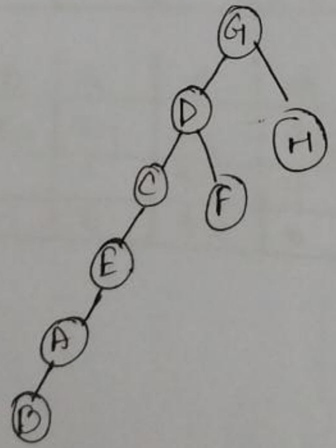
C

BFS ⇒ G, D, F, H, C, E, A, B

~~G~~	~~D~~	~~F~~	~~H~~	~~C~~	~~B~~ ~~E~~	~~A~~	~~B~~.	

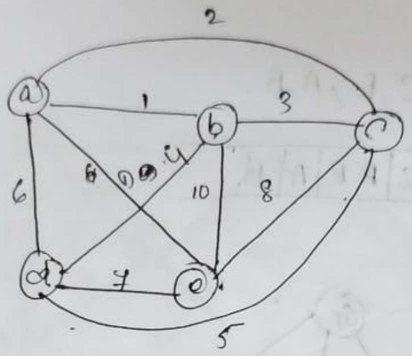


DFS: G, D, C, E, A, B, ~~C~~ F, H

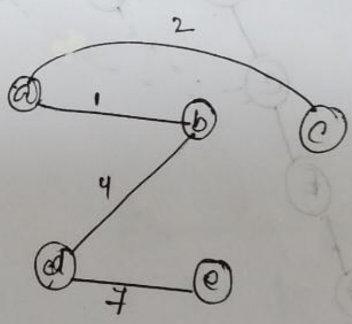


~~F~~
~~B~~
~~A~~
~~E~~
~~C~~
~~D~~
~~G~~

3(a)



ab	→ 1
ac	→ 2
bc	→ 3
bd	= 4
dc.	= 5
ad	= 6
de	= 7
ec	= 8
~~d.e=9~~ ae	→ 9
be	= 10



b

b

	b	a	c	d	e
T	0/1	0	0	0	0
key	–	1	3	4	10
Pπ	–	b	b	b	b

a

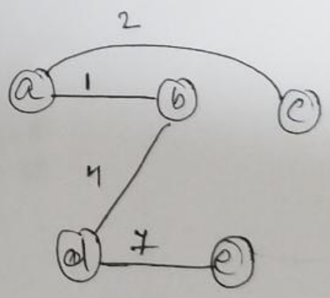
	b	a	c	d	e
T	0/1	0/1	0	0	0
key	–	1	3/2	4	10/9
Pπ	–	b	b/a	b	b/a

c

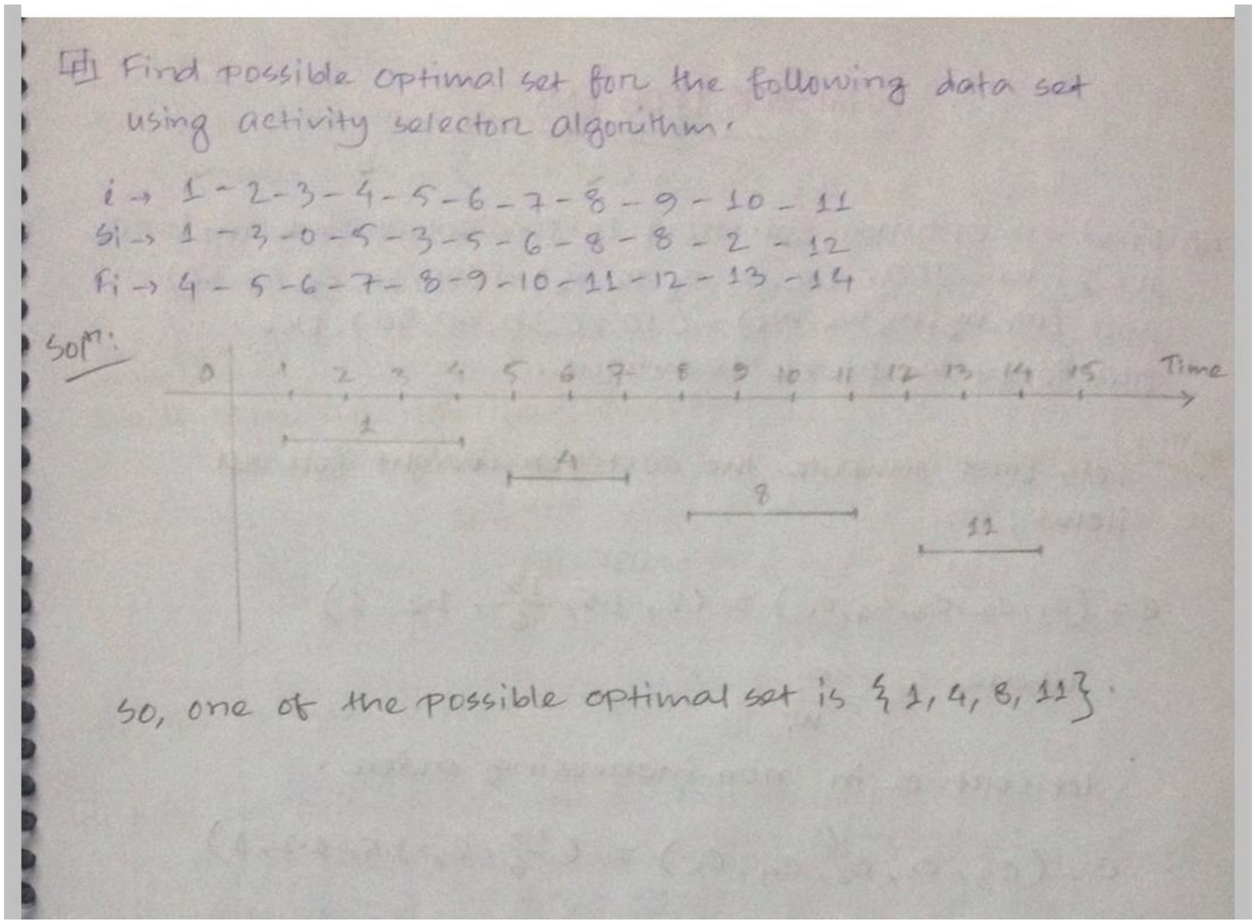
	b	a	c	d	e
T	0/1	0/1	0/1	0	0
key	–	1	3/2	4	10/9/8
Pπ	–	b	b/a	b	b/a/c

d

	b	a	c	d	e
T	0/1	0/1	0/1	0/1	0/1
K	–	1	3/2	4	10/9/8/7
Pπ	–	b	b/a	b	b/a/c/d



b.



4 Find possible optimal set for the following data set using activity selector algorithm:

i → 1 - 2 - 3 - 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11

Si → 1 - 3 - 0 - 5 - 3 - 5 - 6 - 8 - 8 - 2 - 12

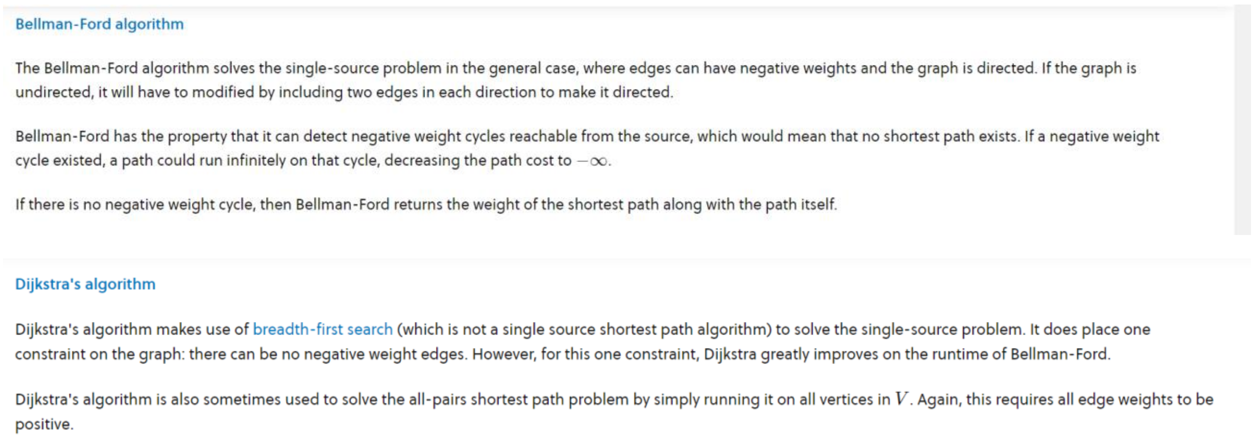
Fi → 4 - 5 - 6 - 7 - 8 - 9 - 10 - 11 - 12 - 13 - 14

Soln:

0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
Time
1
4
8
11

So, one of the possible optimal set is {1, 4, 8, 11}.

4.a



Bellman-Ford algorithm

The Bellman-Ford algorithm solves the single-source problem in the general case, where edges can have negative weights and the graph is directed. If the graph is undirected, it will have to modified by including two edges in each direction to make it directed.

Bellman-Ford has the property that it can detect negative weight cycles reachable from the source, which would mean that no shortest path exists. If a negative weight cycle existed, a path could run infinitely on that cycle, decreasing the path cost to $-\infty$.

If there is no negative weight cycle, then Bellman-Ford returns the weight of the shortest path along with the path itself.

Dijkstra's algorithm

Dijkstra's algorithm makes use of breadth-first search (which is not a single source shortest path algorithm) to solve the single-source problem. It does place one constraint on the graph: there can be no negative weight edges. However, for this one constraint, Dijkstra greatly improves on the runtime of Bellman-Ford.

Dijkstra's algorithm is also sometimes used to solve the all-pairs shortest path problem by simply running it on all vertices in V. Again, this requires all edge weights to be positive.

Floyd-Warshall algorithm

The Floyd-Warshall algorithm solves the all-pairs shortest path problem. It uses a dynamic programming approach to do so. Negative edge weight may be present for Floyd-Warshall.

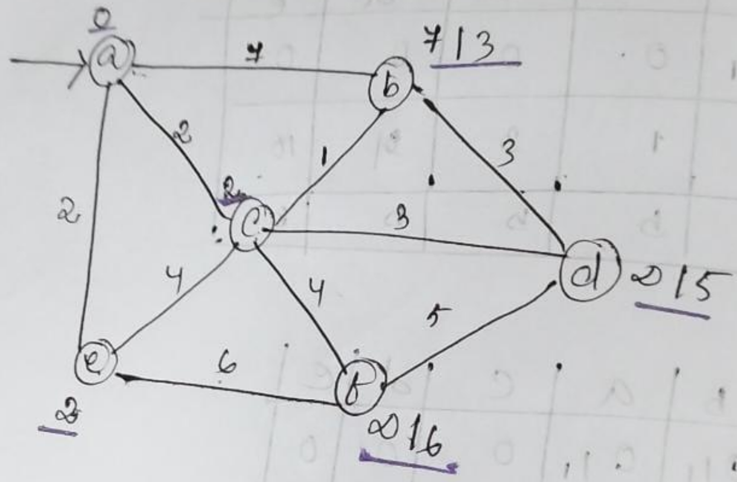
Floyd-Warshall takes advantage of the following observation: the shortest path from A to C is either the shortest path from A to B plus the shortest path from B to C *or* it's the shortest path from A to C that's already been found. This may seem trivial, but it's what allows Floyd-Warshall to build shortest paths from smaller shortest paths, in the classic dynamic programming way.

Johnson's Algorithm

While Floyd-Warshall works well for dense graphs (meaning many edges), Johnson's algorithm works best for sparse graphs (meaning few edges). In sparse graphs, Johnson's algorithm has a lower asymptotic running time compared to Floyd-Warshall.

Johnson's algorithm takes advantage of the concept of reweighting, and it uses Dijkstra's algorithm on many vertices to find the shortest path once it has finished reweighting the edges.

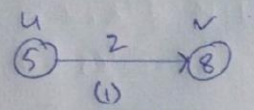
b

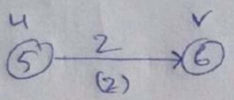


b	3
c	2
d	5
e	2
f	6

4.c

[I] What do you know about relaxation? Show the result of RELAX for an edge (u,v) with weight w(u,v) for the graphs below:





Ans: The process of relaxing an edge (u,v) consists of testing whether we can improve the shortest path to v found so far by going through u and, if so, updating v.d and v.π. A relaxation step may decrease the value of the shortest path estimate v.d and update v's predecessor attribute v.π. The following code performs a relaxation step on edge (u,v) in O(1) time:

```
RELAX (u, v, w)
if v.d > u.d + w(u,v)
        v.d = u.d + w(u,v)
        v.π = u
```

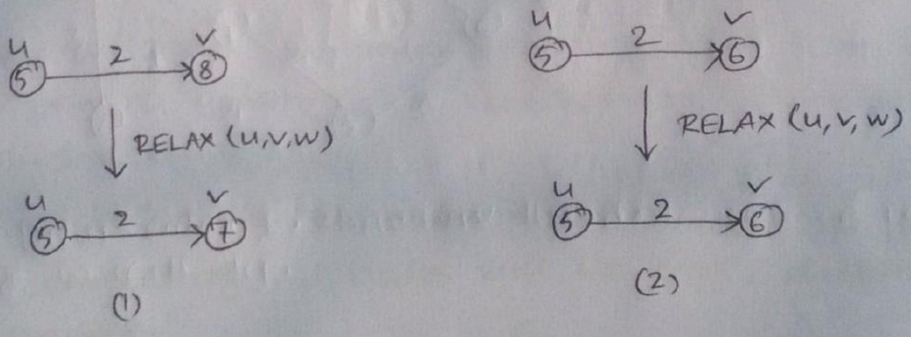
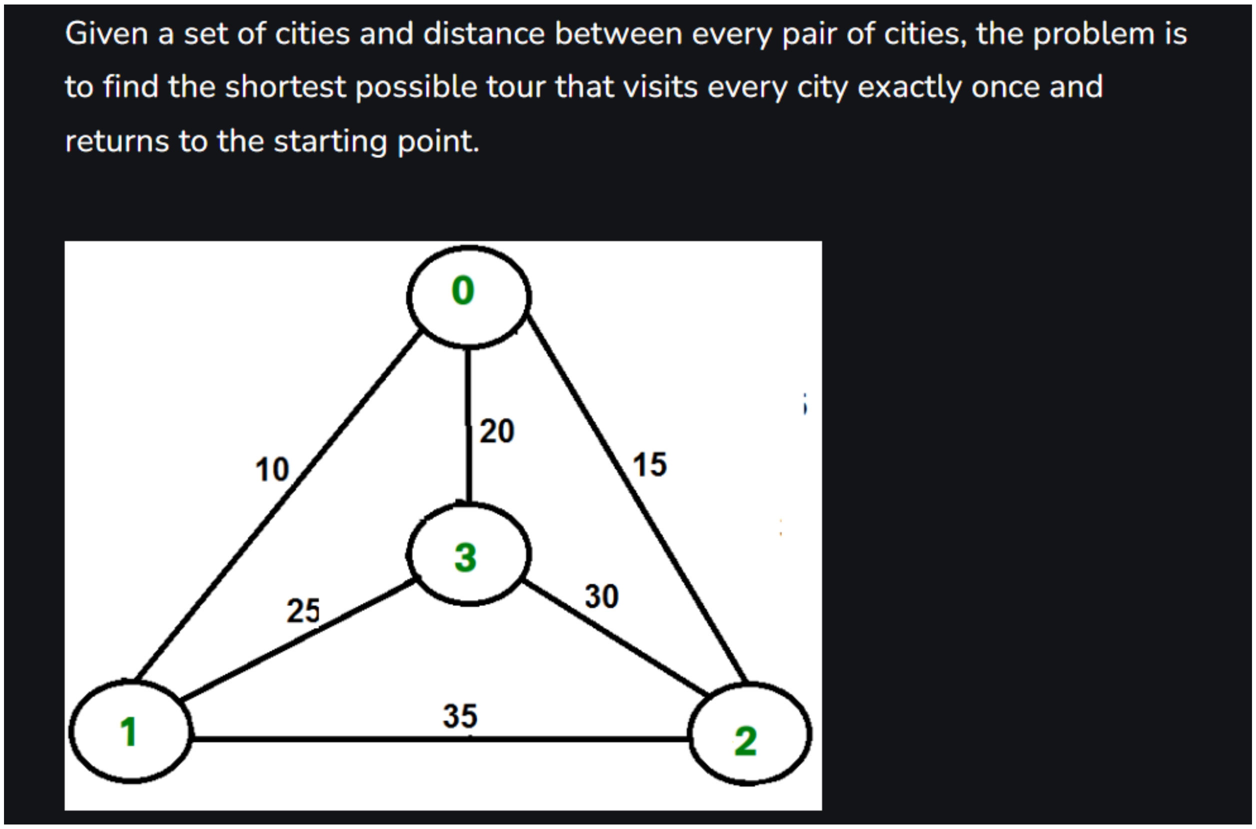


Fig: Result after relaxation

5.a.



Given a set of cities and distance between every pair of cities, the problem is to find the shortest possible tour that visits every city exactly once and returns to the starting point.

For example, consider the graph shown in figure on right side. A TSP tour in the graph is 0-1-3-2-0. The cost of the tour is 10+25+30+15 which is 80.

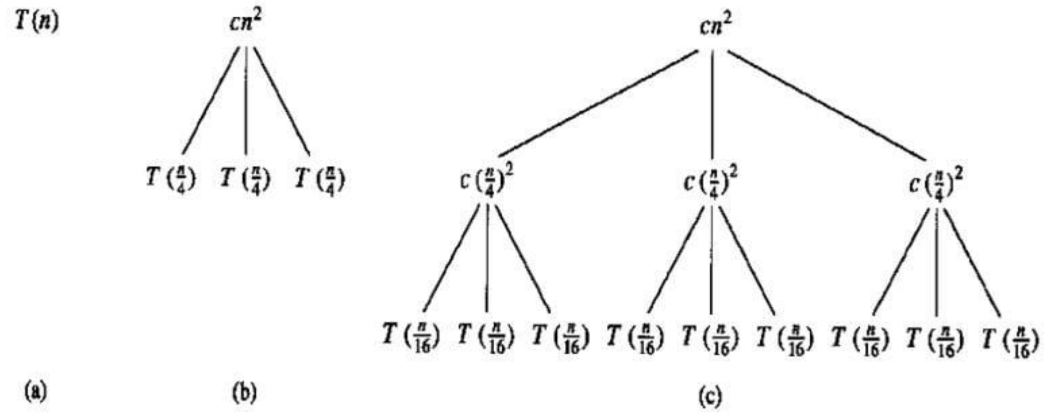
5b.

In computational complexity theory, a problem is NP-complete when: It is a decision problem, meaning that for any input to the problem, the output is either "yes" or "no". When the answer is "yes", this can be demonstrated through the existence of a short (polynomial length) solution.

5c

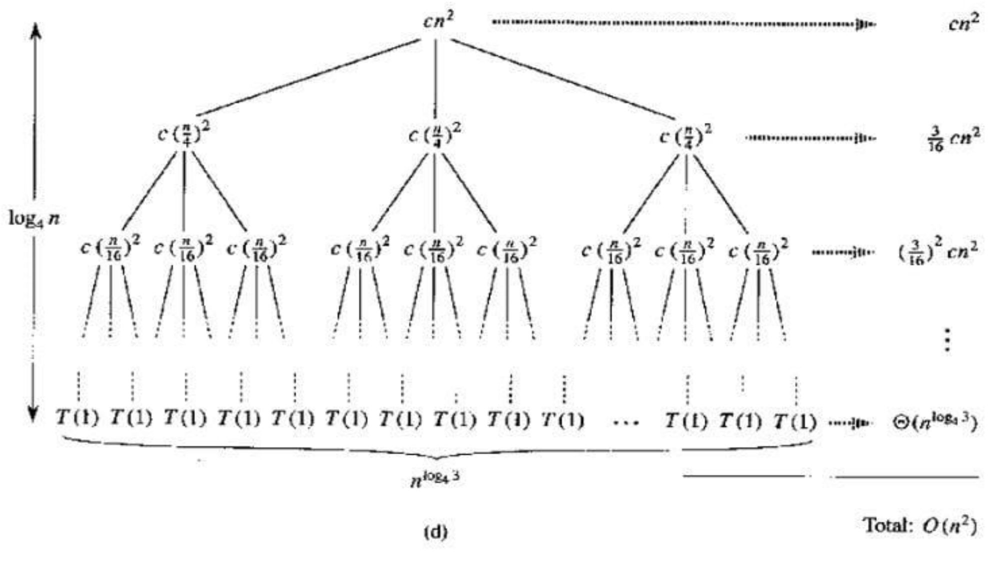
Recursion-tree: Example 1

$$T(n) = 3T(\lfloor n/4 \rfloor) + \Theta(n^2)$$



$T(n)$
cn^2
$T(\frac{n}{4})$ $T(\frac{n}{4})$ $T(\frac{n}{4})$
$c(\frac{n}{4})^2$ $c(\frac{n}{4})^2$ $c(\frac{n}{4})^2$
$T(\frac{n}{16})$
(a)
(b)
(c)

Recursion-tree method



6(a)

■A more detailed algorithm

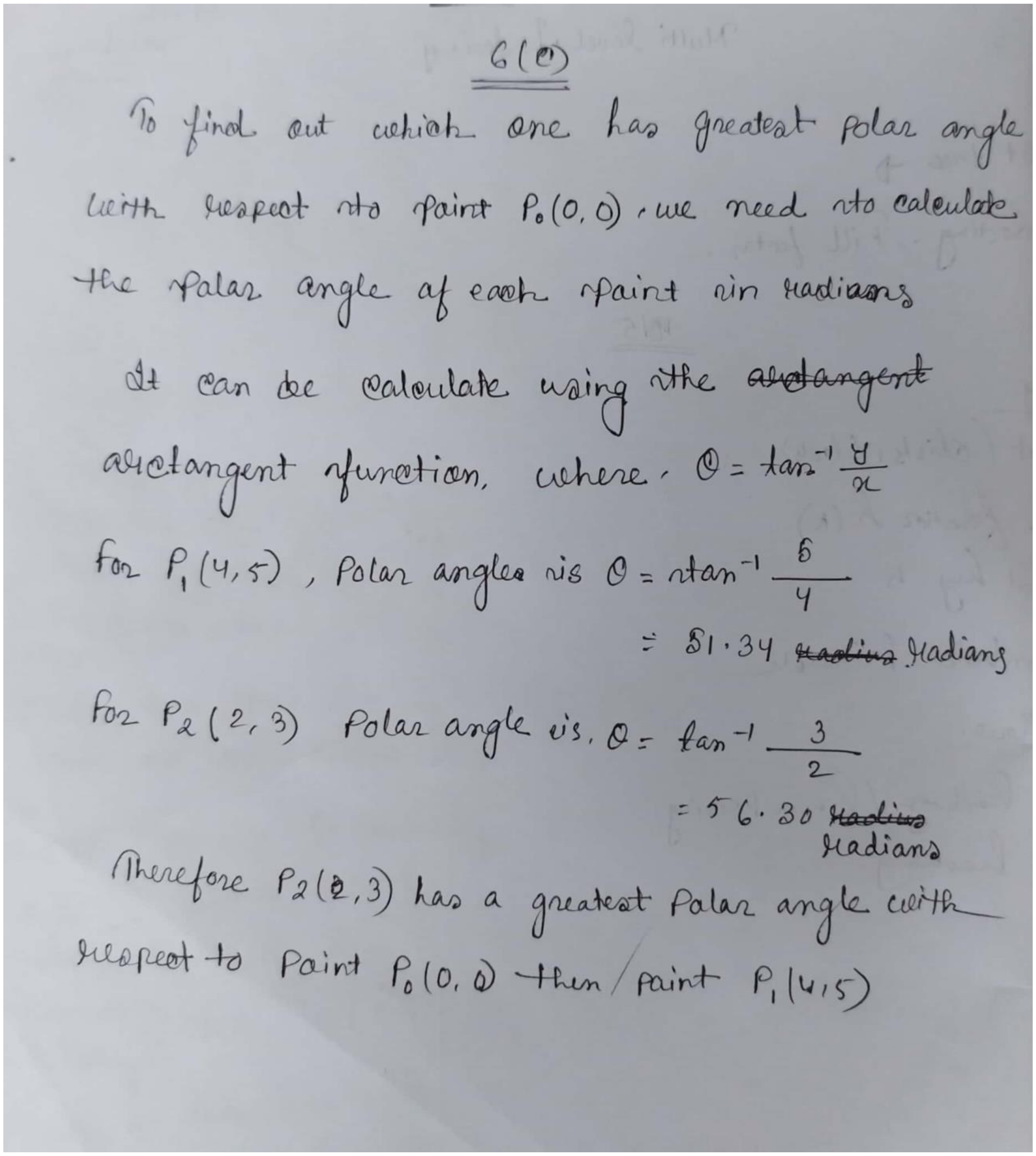
```
GRAHAM-SCAN($Q$)
1  let $p_0$ be the point in $Q$ with the minimum $y$-coordinate,
           or the leftmost such point in case of a tie
2  let $\langle p_1, p_2, \ldots, p_m \rangle$ be the remaining points in $Q$,
           sorted by polar angle in counterclockwise order around $p_0$
           (if more than one point has the same angle, remove all but
           the one that is farthest from $p_0$)
3  PUSH($p_0$, $S$)
4  PUSH($p_1$, $S$)
5  PUSH($p_2$, $S$)
6  for $i \leftarrow 3$ to $m$
7       do while the angle formed by points NEXT-TO-TOP($S$), TOP($S$),
                    and $p_i$ makes a nonleft turn
8               do POP($S$)
9          PUSH($p_i$, $S$)
10 return $S$
```

6(b)

Algorithm for modular exponentiation
To Compute x^n *mod m*

```
Initialise $y=1$, $u=x \bmod m$
Repeat
    if $n \bmod 2=1$ then $y=(y*u) \bmod m$
    $n=n$ div $2$
    $u=(u*u) \bmod m$
Until $n=0$
Output $y$
```

6c.



6(c)

To find out which one has greatest polar angle with respect to point $P_0(0,0)$, we need to calculate the polar angle of each point in radians

It can be calculate using the ~~arctangent~~ arctangent function, where. $\theta = \tan^{-1}\frac{y}{x}$

for $P_1(4,5)$, Polar angles is $\theta = \tan^{-1}\frac{5}{4}$

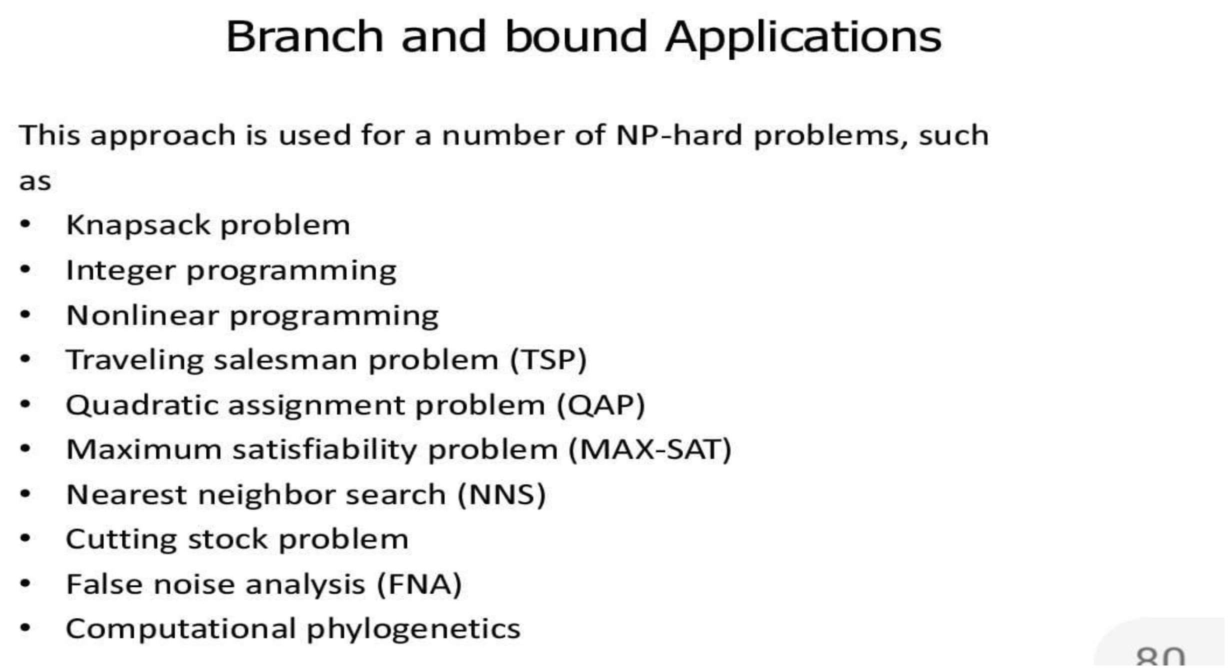
$\doteq 51.34$ ~~radians~~ radians

for $P_2(2,3)$ Polar angle is. $\theta = \tan^{-1}\frac{3}{2}$

$= 56.30$ ~~radians~~ radians

Therefore $P_2(2,3)$ has a greatest polar angle with respect to point $P_0(0,0)$ then / point $P_1(4,5)$

7A.



Branch and bound Applications

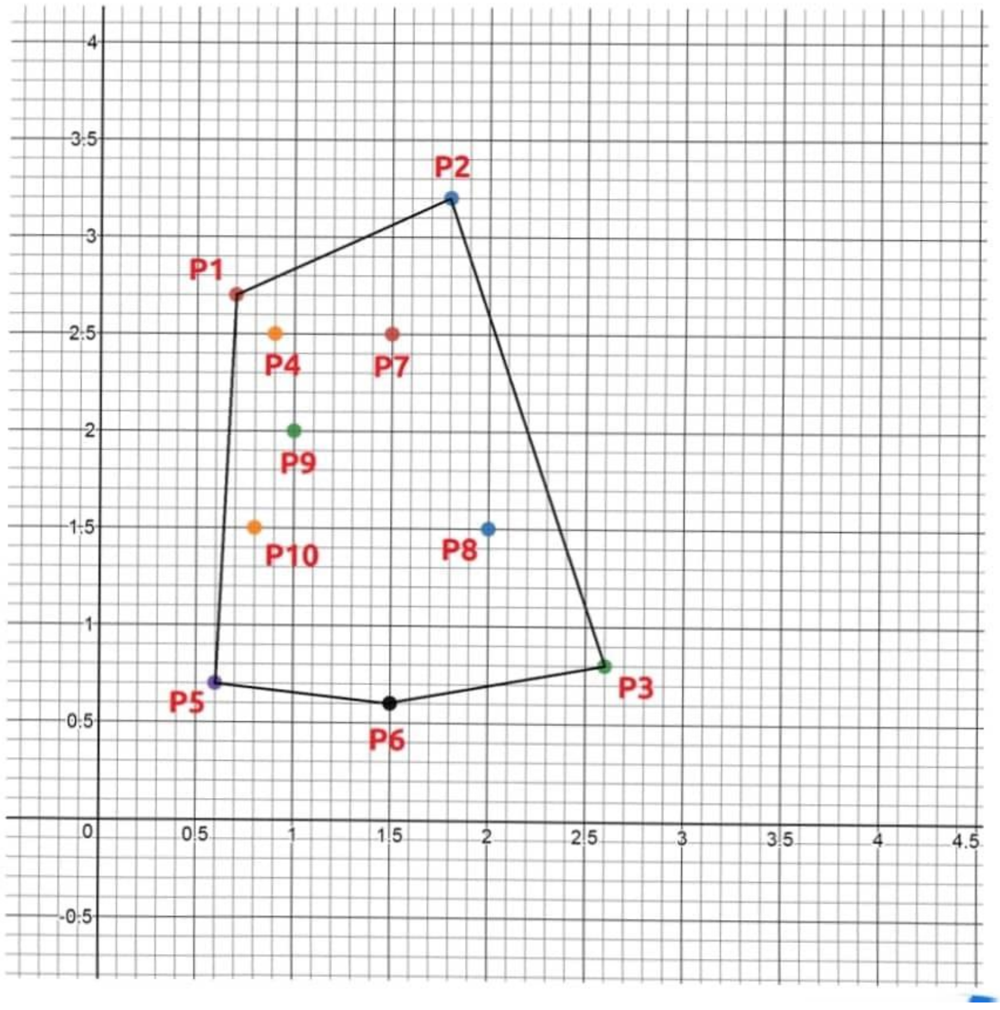
This approach is used for a number of NP-hard problems, such as

- Knapsack problem
- Integer programming
- Nonlinear programming
- Traveling salesman problem (TSP)
- Quadratic assignment problem (QAP)
- Maximum satisfiability problem (MAX-SAT)
- Nearest neighbor search (NNS)
- Cutting stock problem
- False noise analysis (FNA)
- Computational phylogenetics

7(b)

Question: Consider the following points and find convex hull using Graham's Scan algorithm:
P1(0.7, 2.7), P2(1.8, 3.2), P3(2.6, 0.8), P4(0.9, 2.5), P5(0.6, 0.7), P6(1.5, 0.6), P7(1.5, 2.5), P8(2, 1.5), P9(1.0, 2.0), P10(0.8, 1.5)

Answer:



7C.

i) Bellman-Ford's algorithm for single-source shortest path:

Bellman-Ford's algorithm is not a greedy algorithm. It is a dynamic programming algorithm that solves the problem of finding the shortest path from a single source to all other vertices in a weighted graph, even in the presence of negative edge weights.

ii) Dijkstra's algorithm for single-source shortest path:

Dijkstra's algorithm is not a greedy algorithm. It is also a dynamic programming algorithm that finds the shortest path from a single source to all other vertices in a non-negative weighted graph.

iii) Floyd-Warshall's algorithm for all-pairs shortest path:

Floyd-Warshall's algorithm is not a greedy algorithm. It is a dynamic programming algorithm that computes the shortest paths between all pairs of vertices in a weighted graph, including both positive and negative edge weights.

iv) Graham's scan algorithm for finding convex hull:

Graham's scan algorithm is a greedy algorithm. It is used to find the convex hull of a set of points in a plane by selecting points in a specific order.

v) Kruskal's algorithm for finding minimum spanning tree:

Kruskal's algorithm is a greedy algorithm. It finds the minimum spanning tree in a connected, weighted graph by repeatedly adding the smallest weighted edge that does not form a cycle.

vi) Huffman's algorithm for finding optimal prefix code:

Huffman's algorithm is a greedy algorithm. It constructs an optimal prefix code, known as Huffman code, by assigning shorter codes to more frequent symbols and longer codes to less frequent symbols based on their probabilities or frequencies.