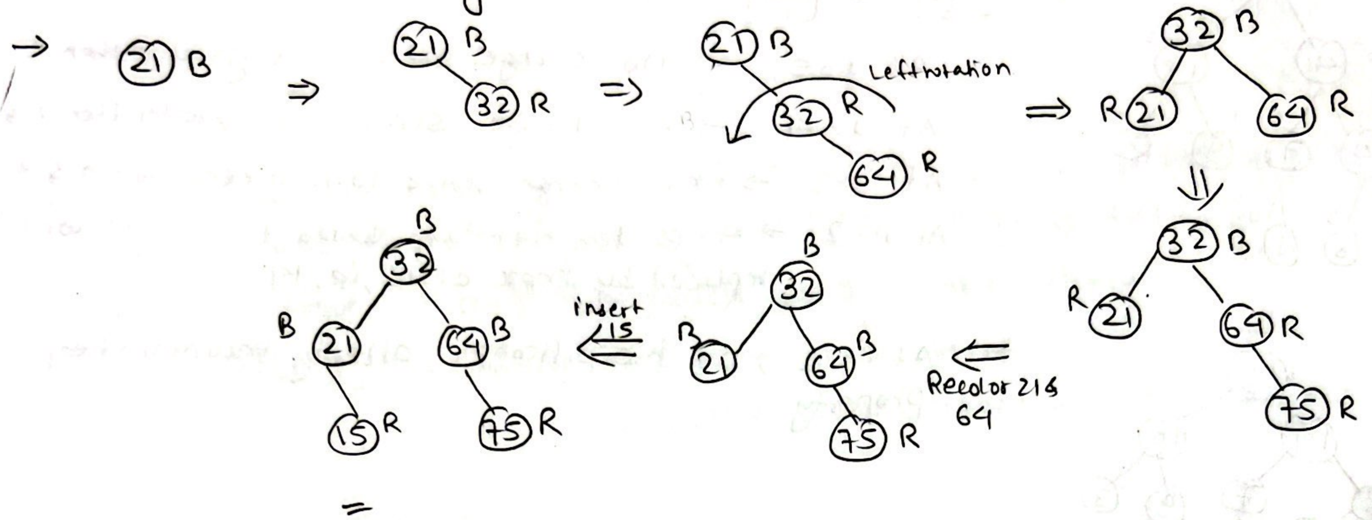
(c) Show the red-black tree that results after each of the integer keys 21, 32, 64, 75, and 15 are inserted in that order, into an initially empty red-black tree. Clearly show the tree that results after each insertion (indicating the color of each node), and make clear any rotation that must be performed.



Q2 a) Consider a 0-1 Knapsack problem with 5 items and with max weight 11. The benefits and weight values are shown below. Find the optimal solution that uses DP. Show all table values. Be sure to state both the value of maximum benefit as well as item selected. Explain how you determine the items selected (you can also use column for keep items/selected items).

Item	value	wt	w_0 0	w_1 1	w_2 2	w_3 3	w_4 4	w_5 5	w_6 6	w_7 7	w_8 8	w_9 9	w_{10} 10	w_{11} 11	selected items
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	1	1	0	1	1	1	1	1	1	1	1	1	1	1	
2	6	2	0	1	6	7	7	7	7	7	7	7	7	7	
3	18	5	0	1	6	7	7	18	19	24	25	25	25	25	
4	22	6	0	1	6	7	7	18	22	23	28	29	29	40	
5	28	7	0	1	6	7	7	18	22	28	29	34	35	35	

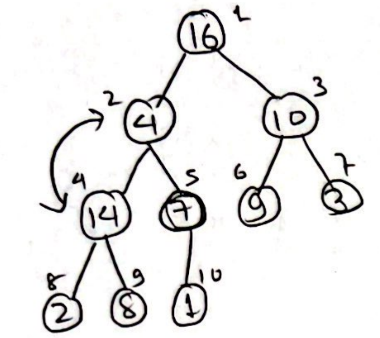
Selected Item 1 2 3 4 5
1 1 0 0 1

wt = 11 − 7 = 4
wt = 4 − 2 = 2
2 − 1 = 1 =

value 1

Previous Question

Qa Run max heapify at i on the following heap. All nodes of the Heap should maintain Heap property after the heapification is completed.



$i = \lfloor \frac{n}{2} \rfloor = \lfloor \frac{10}{2} \rfloor = 5$

At i = 5, → no change since 7 is greater then 1

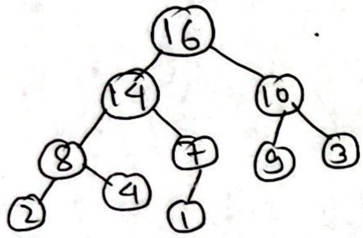
At i = 4 → no change since 14 is greater then 2 & 8

At i = 3 → no change since 10 is greter then 9 & 3

At i = 2 → 4 is less then its Child 14, so it will be replaced by max child, i.e, 14

At i = 1, no heapification, already maintain heap property

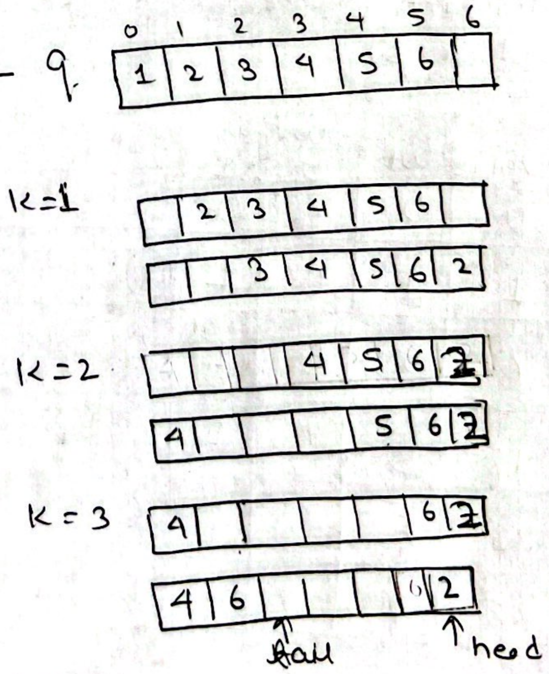
⇓



Qb Given an efficient circular array based queue q capable of holding 7 objects. Show the steps and the final contents of the array after the following code is executed.

```
for(int k=1; k≤6; k++){
    q.enquee(k)
}

for (int k=1 ; k ≤ 3; k++) {
    q.dequeue();
    q.enqueue(q.dequeue);
```



Maximum benefit over the set of all items is 35 and $\{1, 2, 5\}$ are the set of items that gives us maximum benefit (35)

Q What is the running time of 0-1 Knapsack problem using DP used in #a above? Explain your answer in good details.

→ Running time of 0-1 knapsack used in #a above is
$O(nw)$, where n = number of items
w = the limit/capacity of the weight.

$Size = \log_b w$

$w = 2^{size}$ Since, size = number of bits to represent w and b = base.

$\therefore T(n) = O(nw) = O(n \cdot 2^{size})$
$\Rightarrow O(n \cdot w)$

Q Consider Universal Hash function $h(x) = \sum a_i x_i \bmod$. prove that the probability of collision is $1/m$ where there are m slots in the hash table. Key X is presented as $a_0x_0 + a_1x_1 + \ldots$, where x_i values are less than m and a_i values are selected randomly from $\{0, 1, 2, \ldots, m-1\}$

Q3

DIJKSTRA(G, w, s)

1. Initialize-single-source(G, s) ———— V

$S \leftarrow \emptyset$ ———— 1

$Q \leftarrow V[G]$ ———— V (It takes V time to assign in Q)

while $Q \neq \emptyset$ ———— V (It loop to each vertex)

do $u \leftarrow$ Extract-MIN(Q) ————> V Log V [logV to Extract min(Q)]

$S \leftarrow S \cup \{u\}$ ————> 1

7. for each vertex $v \in Adj[u]$ —— $V(\Delta E)$ ——> $V \Delta E \approx 2E$, $O(2E) = O(E)$

do Relax(u, v, w) ————> E log V

→ Everytime it enters to line 7, we get smaller number of vertices at neighbour

→ Hence it accounts for ΔE

$\therefore$ It takes $V(\Delta E)$

$\simeq 2E$

$= O(E)$

So, total running time is

$O(V \log V + E \log V)$

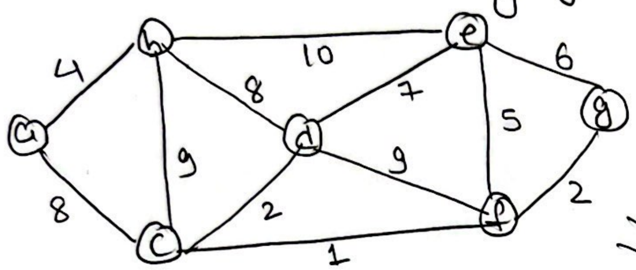
$= O((V+E)\log V)$

$\approx O(E \log V)$

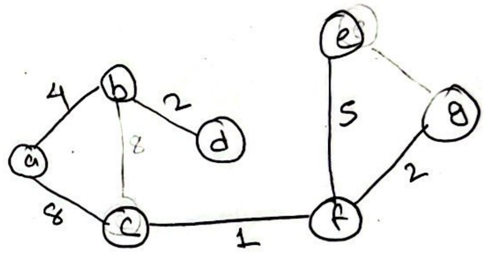
(b) suppose the Line 4 in the pseudo code is changed to "while $|Q| > 1$". This will cause the loop to run $|V| - 1$ times instead of $|V|$ times. will the algorithm work? Explain your answer.

⇒ Yes this will word until there is no negative weight cycle present in the graph. when there is negative weight in graph then, the algorithm does not work.

(5) Consider the following graph. Find the minimum spanning Tree using Prim's algorithm. Clearly show all the steps.



⇒



minimum spanning tree

	a	b	c	d	e	f	g
key	0	∞	∞	∞	∞	∞	∞
π	N	N	N	N	N	N	N

min = a = 4

	b	c	d	e	f	g
K	4	8	∞	∞	∞	∞
π	a	a	N	N	N	N

min = b = 4

	c	d	e	f	g
K	8	8	10	∞	∞
π	a	b	b	N	N

min = c = 4

⇒

	d	e	f	g
K	2	10	1	∞
π	b	b	c	N

min = f = 4

⇒

	d	e	g
K	2	5	2
π	b	f	f

min = d

⇒

	e	g
K	5	2
π	f	f

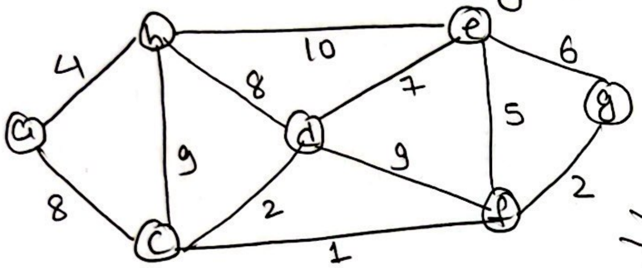
min = g = 4

	e
K	5
π	f

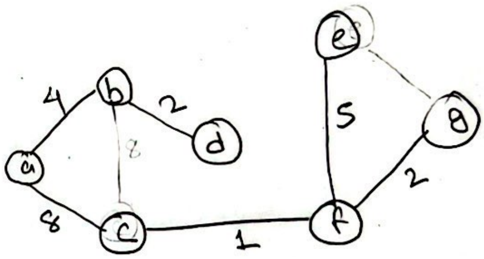
(b) suppose the Line 4 in the pseudo code is chenged to "while $|Q| > 1$". This will cause the loop to run $|V|-1$ times insteed of $|V|$ times. Will the algorithm work? Explain your answer.

⇒ Yes this will word until there is no negative weight cycle present in the graph. When there is negative weight in graph then, the algorithm does not work.

(5) Consider the following graph. Find the minimum Spenning Tree using Prim's algorithm. Clearly show all the steps.



⇒



minimum spanning tree

	a	b	c	d	e	f	g
key	0	∞	∞	∞	∞	∞	∞
π	N	N	N	N	N	N	N

min = a = 4

	b	c	d	e	f	g
K	4	8	∞	∞	∞	∞
π	a	a	N	N	N	N

min = b = 4

	c	d	e	f	g
K	8	8	10	∞	∞
π	a	b	b	N	N

min = c = 4

⇒

	d	e	f	g
K	2	10	1	∞
π	b	b	c	N

min = f = 4

⇒

	d	e	g
K	2	5	2
π	b	f	f

min = d

⇒

	e	g
K	5	2
π	f	f

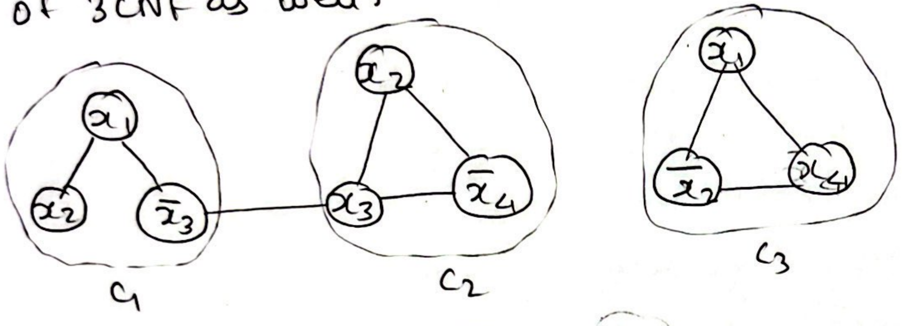
min = g = 4

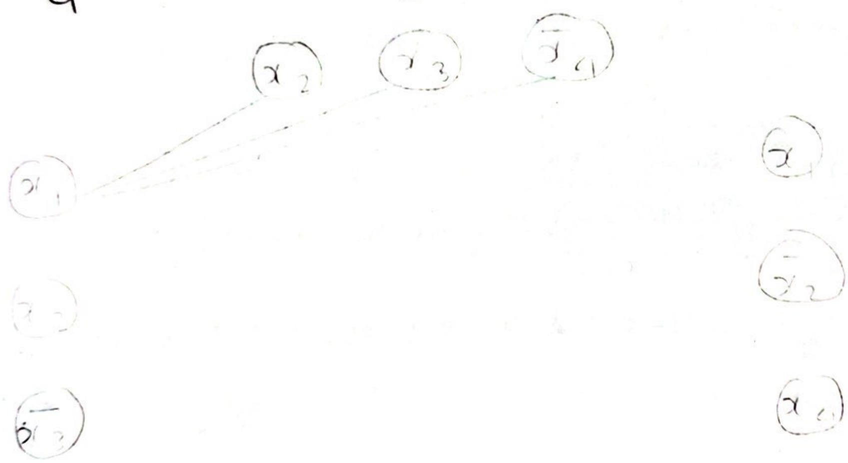
⇓

	e
K	5
π	f

Q 4. An independent set in a Graph is defined as an independent set of vertices in a graph, no two of which are adjacent, That is it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two. The size of an independent set is the number of vertices it contains. Assume that independent set problem is in NP.

⇒ we have to show that the solution for 3-CNF is solution of Independent set and the solution for independent set is solution of 3CNF as well.





Q Can you use Bellmon-ford shortest path algorithm on undirected graph? Explain your answer, will you be able to deal with negative weight?

=> Yes, BF shortest path algorithm can be used for undirect graph because undirected graph is directed graph with 2 ways on each edge.

Generally BF can be used in graph with negative weight but cannot be used with negative cycle.

BF algorithm can handle negative weight in an undirected graph as long as there is no negative weight cycles. If there is negative weight cycle in the graph (a cycle where the sum of edge weight is negative), the algorithm will not work correctly because it will keep reducing the distance values along the cycle indefinitely.

Q Suppose that A and B, and that $A \leq_p B$. Circle one answer for each statement below. Also briefly Justify your answer.

Ans 1. If B is NP-Hard, then A is NP-Hard

True ~~false~~

=> All NP complet, NP, P problems can be reduced to NP hard. So, ...B is not able to reduce A?

2. If A is Exp-Complete then B is Exp complete

True False

=> A reduced to B and its solution is Exp-Complete means B is exp complete

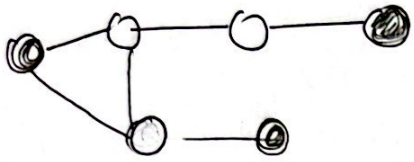
③ Does $A \leq_p B$ implies $B \leq_p A$ in general

True false

=> For example, NP problem 1 can be reduced to NP complete 1 but NP complete problem cannot be reduced to NP prob

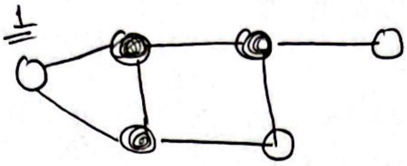
→ concerns finding a maximum dominating set

Q In graph theory, a dominating set for a graph $G = (V, E)$ is a subset D of V such that every vertex not in D is adjacent to at least one member of D. An example of a Dominating set is the set of the 3 nodes shown as bold for the graph G shown below.

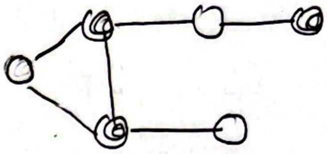


→

1



2



Example Dominating set example

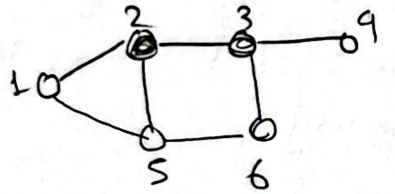


fig 1.

In figure 1, the vertices 2 and 3 are clearly members of a dominating set as every vertex that is not in dominating set {2,3} is adjacent to either 2 or 3.