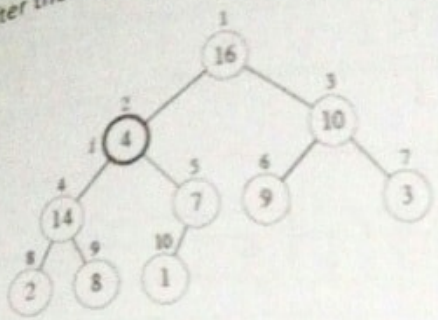
Question 1: 10 points (3 + 4 + 3)

a. Run Max-Heapify at i on the following Heap. All nodes of the Heap should maintain Heap property after the heapification is completed.



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

At i = 4 → no change since 14 is greater than 2 and 8

At i = 3 → no change since 10 is greater than 6 and 7

At i = 2 - check max among child — 4 is less than 7 and 14

At i = 1 no heapification already maintain heap property

b.. 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.enqueue(k);

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

a, c → backward

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 rotations that must be performed.

-0.5

Question 2: 12 points (6 + 3 + 3)

a. Consider a 0-1 Knapsack problem with 5 items and with a max weight of 11. The benefits and weight values are shown below.

Item	Value	Weight
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

Find an optimal solution using Dynamic Programming. Show all Table values. Be sure to state both the value of the maximum benefit as well as the item(s) selected. Explain how you determine the items selected (you can also use a column for Keep-items / selected items).

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

Running time of 0-1 knapsack problem is O(kn) where k is max weight and n is no. of item. For each item it calculate the value of upto k+1 so, overall time consuming is nxk

(-1)

c. 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 represented as a0x0 + a1x1 + ... where xi values are less than m and ai values are selected randomly from {0, 1, 2.....m-1}.

we know

$h(x) = \sum a_i x_i \bmod m$

If m is universal hash function and m is total number

ans → back page.

Question 3: 13 points (5 + 2 + 6))

a. The Single source shortest path using Dijkstra's algorithm is shown in the pseudo code below. Calculate the running time. Assume Adjacency List to represent the graph. Consider Heap is used for Q. Explain your answer clearly by **considering running time for each line.**

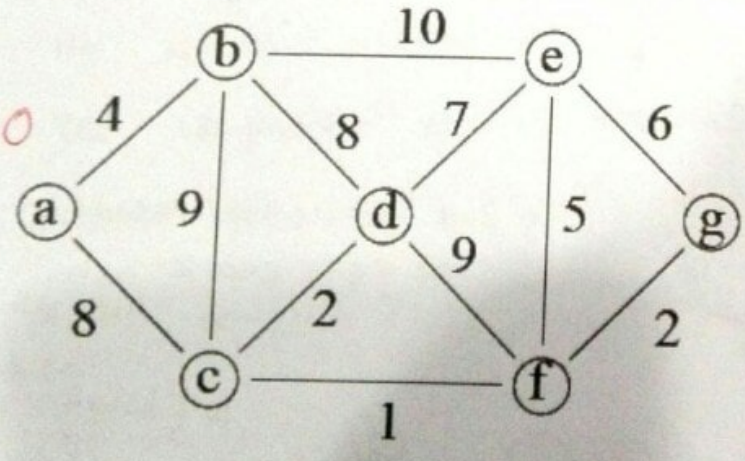
```
DIJKSTRA(G, w, s)                                   Runtime
1  INITIALIZE-SINGLE-SOURCE(G, s)                   V
2  S ← ∅                                            1
3  Q ← V[G]                                         V      → (It take V time to assign in Q
4  while Q ≠ ∅                                      V        (It loop to each vertex)
5       do u ← EXTRACT-MIN(Q)                       VlogV  [:- logV for extract-min(Q)
6          S ← S ∪ {u}                              VlogV
7          for each vertex v ∈ Adj[u]               VE
8               do RELAX(u, v, w)                   VElogV
```

∴ total running time $= O((V+E)\log V) \approx O(E \log V)$

b. Suppose the Line 4 in the pseudo code above 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 work until the negative weight is not in the graph. when there is negative weight in graph then, the algorithm doesnot work.

b. Consider the following graph.



Find the Minimum Spanning Tree using Prim's algorithm. Clearly show all the steps.

(8) → (6) +1 → (7)

Question 4: 11 points (7 + 4)

a. 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.

Assume the following 3-CNF which is NP-Complete.

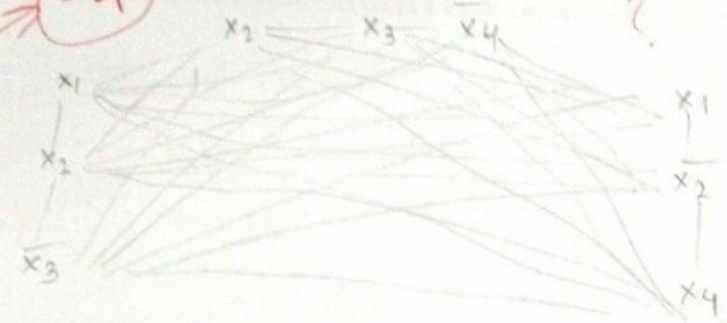
(x1 v x2 v ~x3) ^ (x2 v x3 v ~x4) ^ (x1 v ~x2 v x4) where ~ means not.

Show that Independent set in a graph corresponding to the 3-CNF is NP-Complete.

What is the size of the Independent set S (i.e. maximum number of nodes in S that are not adjacent)? Explain your answer.

[Hint: Reduce 3-CNF to a graph. You can consider a node (e.g. x2) is connected to (~x2) etc.]

(3) → (4)



?

$x_1 = 1$
$x_2 = 1$
$x_3 = 1$
$x_4 = 1$

ans in back

Soln: in back page.

b. Can all NP-Hard problems be reduced to NP problems? Can all NP problems be reduced to NP-Hard problems? Does Power Set problem belong to NP-Complete problem? Explain your answers.

ans. No NP hard problem cannot be NP- problem ✓

Yes NP-problem can be reduced to NP- hard problem because NP- hard problems are at least ~~same~~ as hard as NP- complete problems (where all NP- problems can be reduced). ✓

NO powset doesnot belong to NP-Complete because it is solved in dynamic programming which is exponential time. and it is verified in exponential ✓ time.

(4) → (3)

Question 5: 5 (3 + 2) points [Bonus Question]

(a)

Suppose that we have a hash table with n slots, with collisions resolved by chaining, and suppose that n keys are inserted into the table. Each key is equally likely to be hashed to each slot. Let M be the maximum number of keys in any slot after all the keys have been inserted.

Argue that the probability Q_k that exactly k keys hash to a particular slot is given by

$$Q_k = \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \binom{n}{k}.$$

(n k) means the number of ways k items can be taken from n items.

Expected no of collision = $\Sigma P[h(x) = h(y)] \binom{n}{k}$?

here n no. of slot and n keys are to be inserted

M - max no. of slot in any slot

$Q_k = \left(\frac{1}{n}\right)^k \left(1 - \frac{1}{n}\right)^{n-k} \binom{n}{k}$? ✓

→ 7

b)

MST algorithms by Prim or Kruskal use Greedy strategy. Greedy strategy in general does not guarantee an optimal result. Explain how Greedy strategy in the case of Prim or Kruskal still produce an optimal result.

Prim or Kruskal still produce optimal result because H choose minimum value of edges n vertexes. and also we decision to choose minimum path. In case of Prim it visit through each node and select minimum one ✓