CS-206

END-SEMESTER

- Tarusi Mittal
- 1901CS65
- Tarusi Mittal

Ans 1:-

(a) Divide and conquer recurrence relation for the number of modular multiplication required to compute $a^n \bmod m$,

→ When,

$n = 1$, $a^n \bmod m = a \bmod m$

Case 1: If n is even:

We start by calculating $a^{n/2} \bmod m$

Then we multiply the result (let us say x) with itself and with a (modulo m)

And thus 1 more multiplication occurs.

Case 2: If n is odd:

We start by calculating $a^{(n-1)/2} \bmod m$

Then we multiply the result x with itself and with a (modulo m)

And thus 2 more multiplications occurs.

Therefore; in each recursive step, the number of multiplication is the number of multiplications for $\frac{n}{2}$ (rounded down to nearest interest) and then increased by at most 2 multiplications.

So, if $f(n)$ is the quantity of augmentations required, then basically

$$\boxed{f(n) = f\left(\frac{n}{2}\right) + 2}$$

(b) Using the recurrence relation to construct a big-O estimation.

→ Master Theorem:

→ $f(n) = a f(n/b) + cn^d$

then;

$$f(n) = \begin{cases} O(n^d) & ; \text{ if } a < b^d \\ O(n^d \log n) & ; \text{ if } a = b^d \\ O(n^{\log_b a}) & ; \text{ if } a > b^d \end{cases}$$

The given recurrence relation:

$$f(n) = f\left(\frac{n}{2}\right) + 2$$

for which:

$a = 1$

$b = 2$

$c = 2$

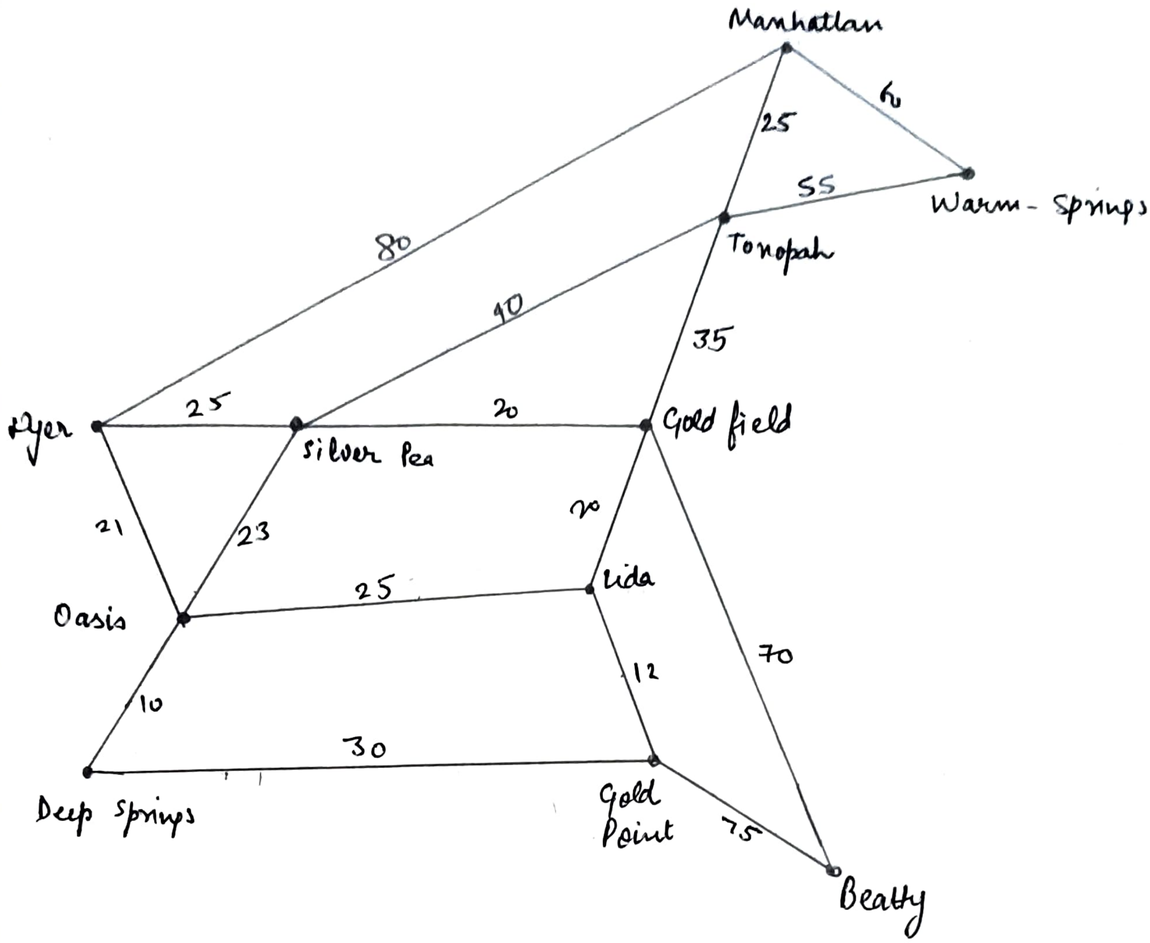
$d = 0$; $a = 1 = 2^0 = b^d$.

Using the master theorem.

$$f(n) = O(n^d \log n) = O(n^0 \log n) = O(\log n)$$

$$\Rightarrow \boxed{f(n) = O(\log n)}$$

Ans 2:-



As given in the question, considering cities as nodes and roads as edges; The question turns out to find the minimum spanning tree of the graph for which the Kruskal's Algorithm will be used.

In Kruskal's algorithm, any edge that doesnot complete a simple circuit ie a closed path who starting and ending vertex are same can be added.

In short the Minimum Spanning Tree should not be containing any cycles.

Steps for Kruskal's Algorithm:

1) All the edges will be sorted in the non decreasing order of their weight.

2) After choosing the smallest edge, while moving to the other edges it will be checked whether they make a cycle or not; If there is no cycle, the edge will be included, otherwise won't.

3) The second step will be repeated there are (n-1) edges in the spanning tree [n is the no. of vertices of the graph].

1st step: Arranging in non-decreasing order of weights.

10, 12, 20, 20, 21, 23, 25, 25, 25, 30, 35, 40, 45, 50, 60, 70, 80.

2nd step: The smallest edge → 10

Edge 1: 10 - Oasis to Deep springs → Success (Paved)

Edge 2: 12 - Lida to Gold Point → Success (Paved)

Edge 3: 20 - Lida to Gold feild → Success (Paved)

Edge 4: 20 - Gold feild to Silver Peak → Success (Paved)

Edge 5: 21 - Oasis to Dyer → Success (Paved)

Edge 6: 23 - Oasis to Silver Pea → Success (Paved)

Edge 7: 25 - Oasis to Lida → Failure (cycle formed) (Not paved)

Edge 8: 25 - Dyer to Silver Pea → Failure (cycle formed) (Not Paved)

Edge 7: 25 - Manhattan to Tonopah - Succen (Paved)

Edge 8: 30 - Deepsprings to gold point - Failure (cycle formed) (Not paved)

Edge 8: 35 - Goldfeild to Tonopah - Succes (Paved)

Edge 9: 40 - Silver Pea to Tonopah - Failure (cycle formed) (Not paved)

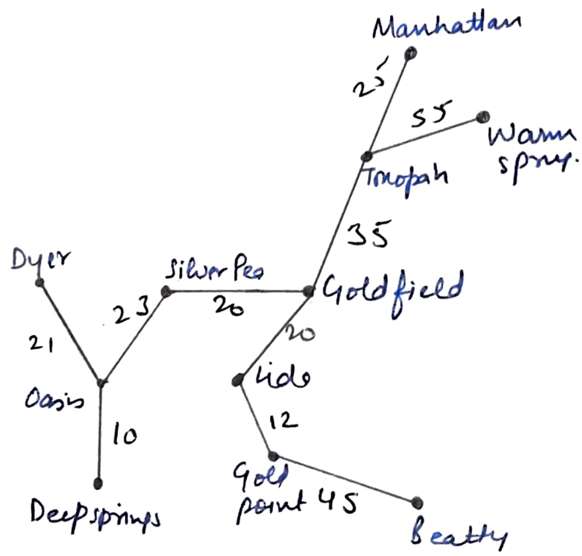
Edge 9: 45 - gold Point to Beatty - Succes (Paved)

Edge 10: 55 - Tonopah to Warm spring - Succes (Paved)

No. of edges connected = 10 = 11-1 = no of vertices -1.

⇒ So length of the required path:

Oasis ↔ Deepspring	=	10
Oasis ↔ Dyer	=	21
Oasis ↔ Silver Pea	=	23
Silver Pea ↔ Gold field	=	20
Gold field ↔ Lida	=	20
Lida ↔ Goldpoint	=	12
Goldpoint ↔ Beatty	=	45
Gold field ↔ Tonopah	=	35
Tonopah ↔ Manhattan	=	25
Tonopah ↔ Warmspring	=	55
		266



The Minimum spanning Tree to Be Paved.

Total path to be paved = 266