PAGE NO. :
DATE : 12/1/22

2nd Internal ADA

Section A

1. **Best Case**

→

$$T(n) = \begin{cases} 0 & \\ T(n/2) + T(n/2) + n & n > 1 \end{cases}$$

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

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

$$= 2[2T(n/4) + n/2] + n$$

$$= 4T(n/4) + n + n$$

$$= 4[2T(n/8) + n/4] + 2n$$

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

$$= 8T(n/8) + 3n$$

$$= 2^3 T(n/2^3) + 3n$$

⋮

Replace with i

$$= 2^i T(n/2^i) + in$$

$$= n.T(n/n) + in$$

$$= n.T(1) + in$$

$$= n.0 + in$$

$$= in$$

$$= \log n . n$$

$$T(n) = \Omega(n \log n$$

Substitution

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

$$T(n/2) = 2T(n/4) + n/2$$

$$T(n/4) = 2T(n/8) + n/4$$

Let $n = 2$

log on both sides

$$\log 2 = \log n$$

$$i \log 2 = \log n$$

$$i.1 = \log n$$

$$i = \log n$$

PAGE NO. :
DATE : / /

worst case

->

$$T(n) \begin{cases} 0 & n=0 \\ 0 & n=1 \\ T(0)+T(n-1)+n & n>1 \end{cases}$$

$T(n) = T(0) + T(n-1) + 2$

$T(n) = T(n-1) + n$

$= T(n-2) + n-1 + n$

$= T(n-3) + n-2 + n-1 + n$

⋮

$= T(n-n) + n-(n-1) + n-(n-2) + \dots + n$

$= T(0) + 1 + 2 + \dots + n$

$= 1 + 2 + \dots + n$

$= \frac{n(n-1)}{2}$

$= \frac{n^2+n}{2}$

$T(n) = O(n^2)$

Average case

-> This time complexity is $O(n \log n)$.

solve the recurence relation. :-

$$Tn = \begin{cases} 1 & n=1 \\ T(n/2)+n & n>1 \end{cases}$$

PAGE NO. :
DATE : / /

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

$$= T(n/4) + n/2 + n$$

$$= T(n/8) + n/4 + n/2 + n$$

$$= T(n/2^3) + n/2^2 + n/2^1 + n_{2^1}$$

⋮

Replace with i

$$= T(n/2^i) + n/2^{i-1} + n/2^{i-2} + n/2^0 \qquad \left[\because \frac{n}{2^{i-1}} = \frac{n}{2^i * 2^{-1}}\right]$$

Let $2^i = n$

$$= T(n/n) + \frac{n}{n.2^{-1}} + \frac{n}{n.2^{-2}} + \frac{n}{n.2^{-3}} + \dots + n$$

$$= T(1) + n\left(\frac{1}{n-2^{-1}} + \frac{1}{n.2^{-2}} + \dots + \frac{1}{2} + 1\right)$$

$$= T(1) + n(1+1)$$

$$T(n) = O(n)$$

$\therefore 1/2 + 1/4 + 1/8 \dots = 1$

⟶ P.T.O

PAGE NO. :
DATE : / /

3. Control abstraction of Greedy method

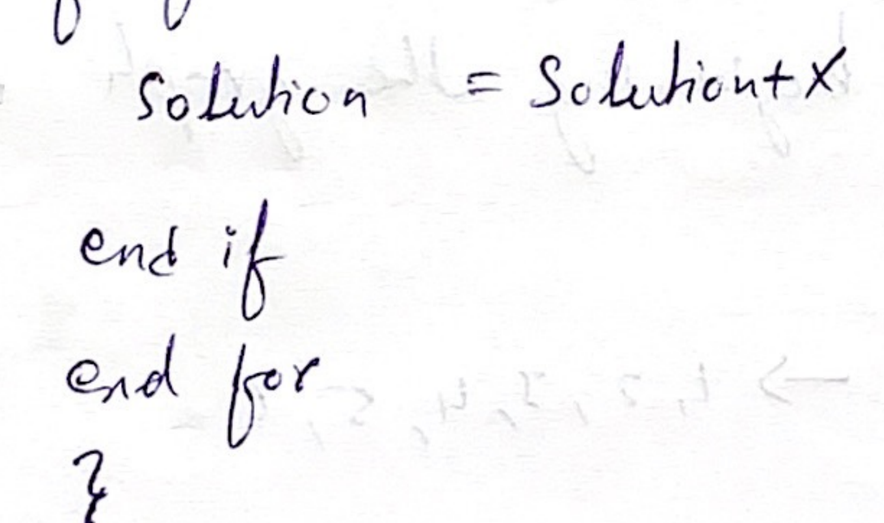
→ Greedy method says problem should be solved in diff. stages, each stage will be considered as input for the feasible sol. From the feasible sol list, we will get the optional sol.

Algo :

```
Greedy method (A, n)
{
for i = 1 to n do
    x = select (A[i])
if feasible (x) then
    solution = solution + x
end if
end for
}
```

Section B

2.



PAGE NO. :
DATE : / /

Step 1 : Arrange edges weight in increasing order.

Edge	weight / cost
1 - 2	5
1 - 3	10
2 - 6	20
3 - 6	25
4 - 5	35
2 - 4	45
5 - 6	45
4 - 6	50
2 - 5	55

Step 2 : Let in beginning the graph should be empty.

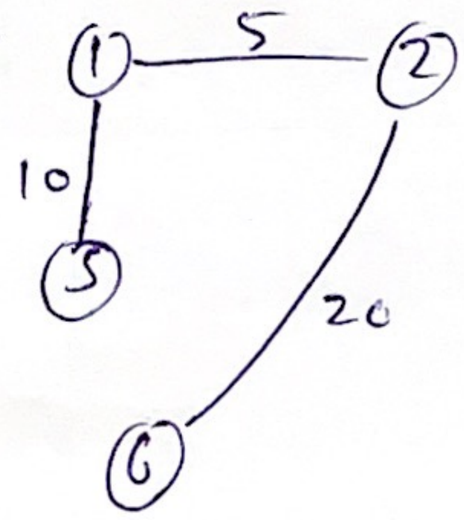
need visit → 1, 2, 3, 4, 5, 6

Step 3 : (1) —5— (2) , no circle formation, Edge Accepted

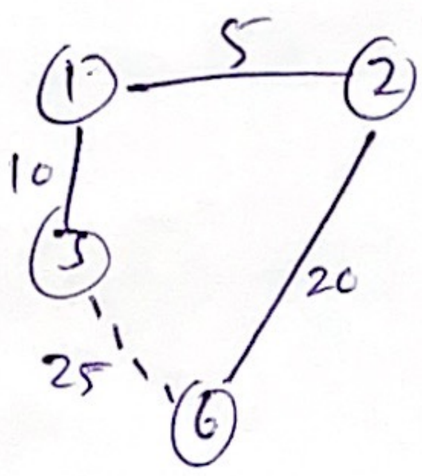
[1, 2] [3, 4, 5, 6]

Step 4 : (1) —5— (2) → Edge Accepted, no circle formed [1, 2, 3] [4, 5, 6]
10 |
(3)

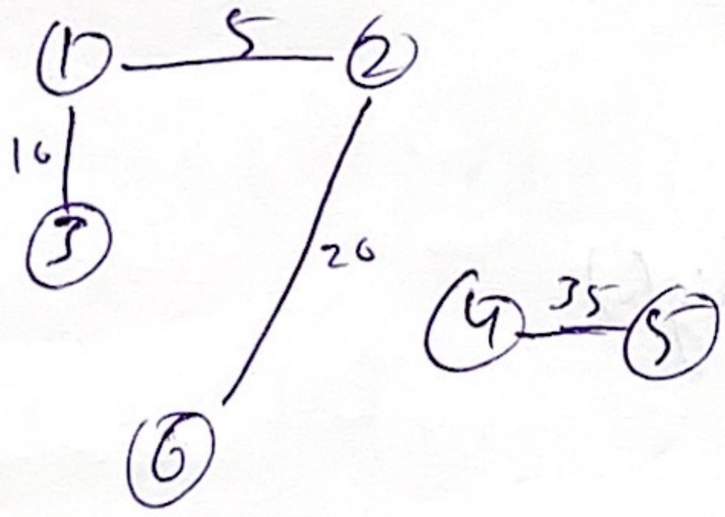
PAGE NO.:
DATE: / /

Step 5 : 

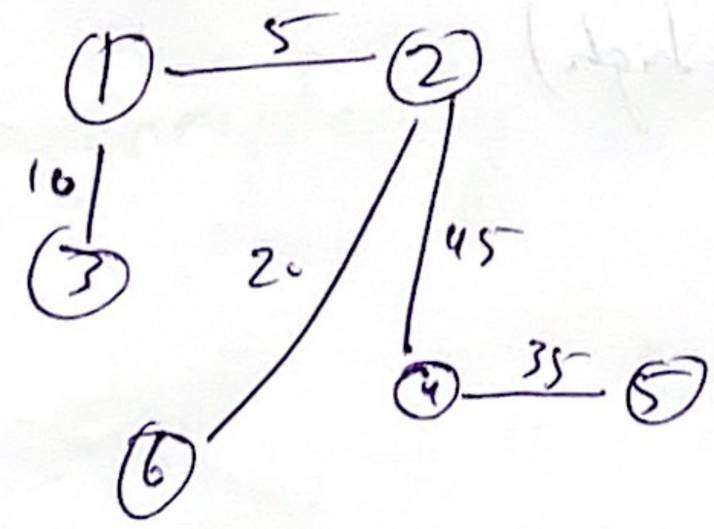
=> no circle formation,
Edge Accepted
[1, 2, 3, 6] [4, 5]

Step 6 : 

-> circle is formed, Edge is not Accepted
[1, 2, 3, 6] [4, 5]

Step 7 : 

=> No circle is formed
Edge Accepted
[1, 2, 3, 4, 5, 6]

Step 8 : 

=> No circle is formed
Edge Accepted
[1, 2, 3, 4, 5, 6]

Step 9 : $\langle 5, 6 \rangle$ = Rejected

Step 10 : $\langle 4, 6 \rangle$ => Rejected

Step 11 $\langle 2, 5 \rangle$ = Rejected

∴ Total minimum cost spanning tree => 5+10+20+45+35
=> 115 //

PAGE NO. :
DATE : / /

1) Merge Sort algorithm

```
-> mergesort (a, low, high)
   {
    if (low < high)
    {
     mid = (low+high)/2
mergesort (a, low, mid)
mergesort (a, mid+1, high)
merge(a, low, mid, high)
   }
}
merge (a, low, mid, high)
{
 i = low, j = mid + 1, k = low
while (i <= mid && j <= high)
  {
  if (a[i] < a[j])
c[k] ← a[i]
  k++
  i++
else
  c[k] ← a[j]
  k++
  i++
  }
```

PAGE NO. :
DATE : / /

```
while (i <= mid)
{
   c[k] ← a[i]
     k++
     i++
}
while (j <= high)
 {
  c[k] ← a[j]
    k++
    j++
 }
for (i=0; i<n-1; i++)
   {
    a[i] ← c[i]
}
```

Time Complexity :-

$$T(n) = \begin{cases} 0 & n = 1 \\ T(n/2) + T(n/2) + n & n > 1 \end{cases}$$

↑ Left ↑ compare & combine

↑ Self

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

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

$$= 2[2T(n/4) + n/2] + n$$

$$= 4T(n/4) + n + n$$

PAGE NO. :
DATE : / /

$= 4\left[2T(n/8) + n/4\right] + 2n$

$= 8T(n/8) + 3n$

$= 2^3 T(n/2^3 + 3n$

⋮

Replace constant with i

$= 2^i T(n/2i) + in$

$= nT(n/n) + in$

$= nT(1) + in$

$= n.0 + in$

$= in$

$= \log n . n$

$T(n) = O(n \log n)$

Substitution

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

$T(n/2) = 2T(n/4) + n/2$

$T(n/4) = 2T(n/8) + n/4$

Let $2^i = n$

Subsitute log both sides

$\log 2^i = \log n$

$i \log 2 = \log n \quad [\log 2 = 1]$

$i.1 = \log n$

$i = \log n$