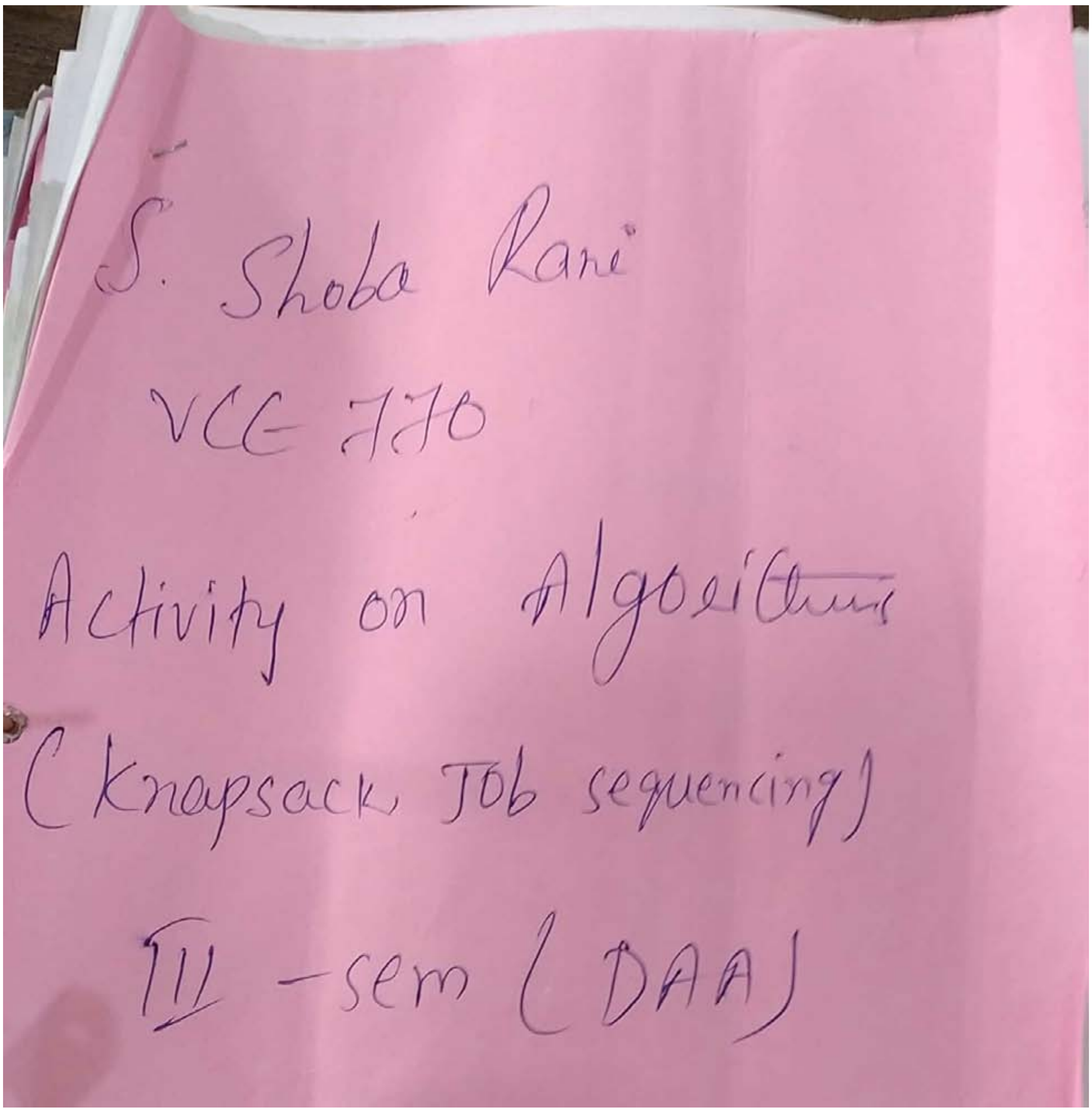
Department of Computer Science & Engineering

Dynamic Activities Conducted by the Faculty Members



Team 6: ~~5H4~~, 5L9, 5KL7, 5NA, 5K6, 5P3.

apsack Problem: Consider 'n' objects with profits P_1 to P_n and weights (w_1 to w_n) and Maximum capacity 'm'.
he main objective of knapsack problem is to fill the knapsack with maximum profit.

No of Objects n = 7

Profits $(P_1, P_2, \ldots P_7)$ = 10, 5, 15, 7, 6, 18, 3

Weights $(w_1, w_2, \ldots w_7)$ = 2, 3, 5, 7, 1, 4, 1

P_i	w_i	P_i/w_i
10	2	5
5	3	1.67
15	5	3
7	7	1
6	1	6
18	4	4.5
3	1	3

$\frac{P_5}{w_5} > \frac{P_1}{w_1} > \frac{P_6}{w_6} > \frac{P_3}{w_3} > \frac{P_7}{w_7} > \frac{P_2}{w_2} > \frac{P_4}{w_4}$

Rules:
1) Calculate P_i/w_i ratio
2) Arrange them in decending Order

m = 15

$P_5 = 6$ $w_5 = 1$
m = 15 − 1 = 14

$P_1 = 10$ $w_1 = 2$
m = 14 − 2 = 12

$P_6 = 18$ $w_6 = 4$
m = 12 − 4 = 8

$P_3 = 15$ $w_3 = 5$
m = 8 − 5 = 3

$P_7 = 3$ $w_7 = 1$
m = 3 − 1 = 2

$P_2 = 5$ $w_2 = 3$
$m = 2 - \frac{2}{3}(3) = 0$

Profit $= P_5 + P_1 + P_6 + P_3 + P_7 + \frac{2}{3}(P_2)$
$= 6 + 10 + 18 + 15 + 3 + \frac{2}{3}(5)$
= 55.3

objects kept in knapsack

$x_1, \frac{2}{3}x_2, x_3, x_5, x_6, x_7$

Algorithm: Knapsack (m, n)

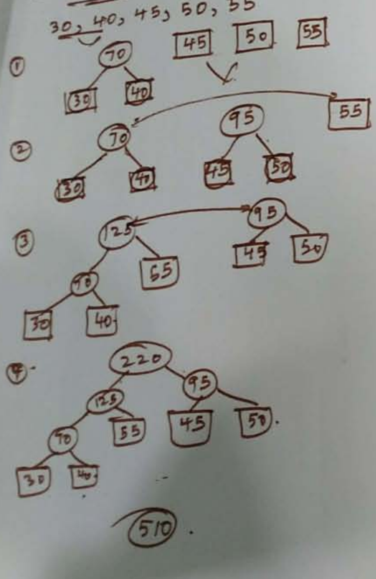
```
{
  // p[1:n] and w[1:n] are
  Profits and weights are stored in
  decending order of their ratios
  Pi/wi
  Profit := 0.0;
  for i:=1 to n do
  {
     if (w[i] ≤ m)
     {  m := m - w[i];
        Profit := profit + p[i];
     }
     else
     {
        Profit = profit + (m / w[i] * p[i])
        m = 0;
        break;
     }
  }
  return profit;
}
```

①

Sachin
Niharika
Karishma
Ashish
Leela prasad.

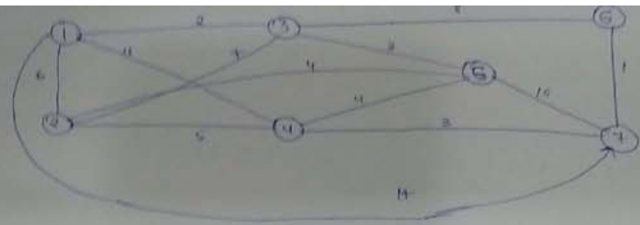
How many min no. of edges req for merging of different files $(x_1, x_2, x_3, x_4, x_5) = (40, 30, 50, 45, 55)$;

Sorted order,
30, 40, 45, 50, 55



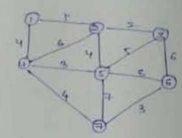
Algorithm (

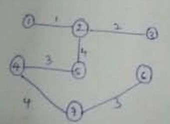
```
Algorithm (minedges (n, a
{
 // n is no. of vertices
 // a[1,n] in ascending order
 Merge first two elements.
 for (i := 1 to n do.
 {
   pt = new. tree node.
   (pt -> lchild) = least (list);
   (pt -> rchild) := least (list);
   (pt -> weight) := (pt -> lchild -> weight) + (pt -> rchild -> weight)
   insert (list, pt);
 }
 tree node := return least(list);
}
 tree node := Record:
{
  tree node * lchild:
  treenode * rchild:
  int weight:
}.}
```



Intermidiate nodes	Vertex selected	1	2	3	4	5	7
6	—	+α	+α	8	+α	+α	1
{6, 7}	7	15	+α	8	4	11	1
{6, 7, 4}	4	15	9	8	4	8	1
{6, 7, 4, 5}	5	15	9	8	4	8	1
{6, 7, 4, 5, 3}	3	10	9	8	4	8	1
{6, 7, 4, 5, 3, 2}	2	10	9	8	4	8	1
{6, 7, 4, 5, 3, 2, 1}	1						

Algorithm





Total cost = 1+2+3+3+4+4

⇒ 17

Algorithm - Kruskal's (cost, weight, e, n)

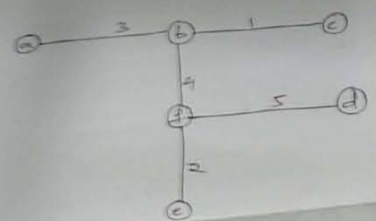
team - 7

```
{
    t[r,i] = k ;   t[i,0] = j ;
    for i:=0 to n-1 do
      ~~parent[i] =~~ ; i = 0;
      mincost = 0;
    for i:= to n-1 do
    {while((i<n-1) && heap is not empty) do
     {
       delete min cost edge [u,v] from heapfy and reheapify using adjust;
       if (j ≠ k) then
          j = find [u];
          k = find [v];
           i = i+1;
           t [i,1] = j;
           t [i,2] = k;
           mincost = mincost + [u,v]
            union (k,j] ;
      } if (n ≠ n-1) write no spanning tree
    }   else ~~written~~ min cost
             return
}
```

Evaluated by team - 8

3 ~~How many Find the minimum~~

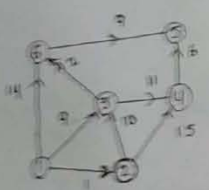
Minimum Spanning Tree :-



(b,c) = 1 ✓
(b,a) = 3 ✓
(b,f) = 4 ✓
(c,d) = 6
(c,f) = 4 ✓
(f,e) = 2 ✓
(f,a) = 5
(f,d) = 5 ✓

Team :- 8
JN3, 5N5, N7, N6, PD, N8, N9

1- shortest path :-



Iteration	S	Action Taken	[1]	[2]	[3]	[4]	[5]	[6]
-	-	{1}	0	1	9	∞	∞	14
1	{1}	2	0 ✓	1 ✓	9	16	∞	14
2	{1,2}	3	0 ✓	1 ✓	9 ✓	16	∞	11
3	{1,2,3}	6	0 ✓	1 ✓	9 ✓	16	20	11 ✓
4	{1,2,3,6}	4	0 ✓	1 ✓	9 ✓	16 ✓	20	11 ✓
5	{1,2,3,6,4}	5						

5 is destination, 1 is source

Algorithm shortestpath(v, cost, dist, n)

{

for i:=1 to n do

{

s[i] := false ; dist[i] := cost[v,i];

}

s[v] := true dist[v] = 0

for num := 2 to n-1 do

s[u] := true

for (each w adjacent to a and s[w] := false) do

if (distance[w] > dist[u] + cost[u,w]) then

dist[w] := dist[u] + cost[u,w] ; // update adjacent verticies

}.

Evaluator: Team 7

Faculty Name: Mrs. S. Shoba Rani
Activity Name: TAPPS
Class: B.Tech II Year
Subject: Design and Analysis of Algorithms
Topic: Finding Shortest Path Using Kruskal’s algorithm