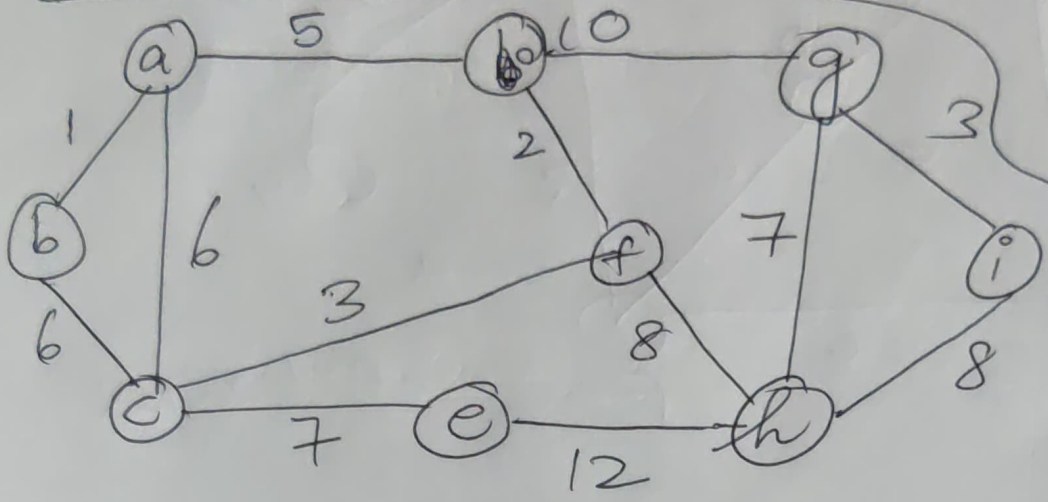
MODULE-3
GREEDY METHOD

PRIM'S ALGORITHM → Visit all nodes, that's all ~~lowest~~ shortest path



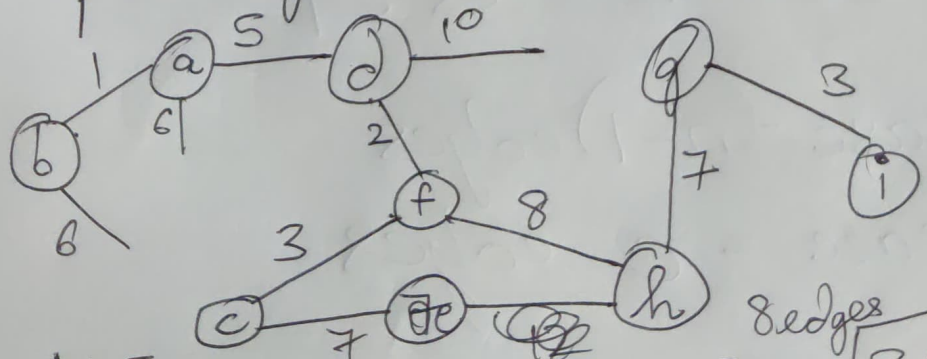
To find Minimum Cost Spanning Trees

For (V vertices, V-1 edges.)

Minimum cost

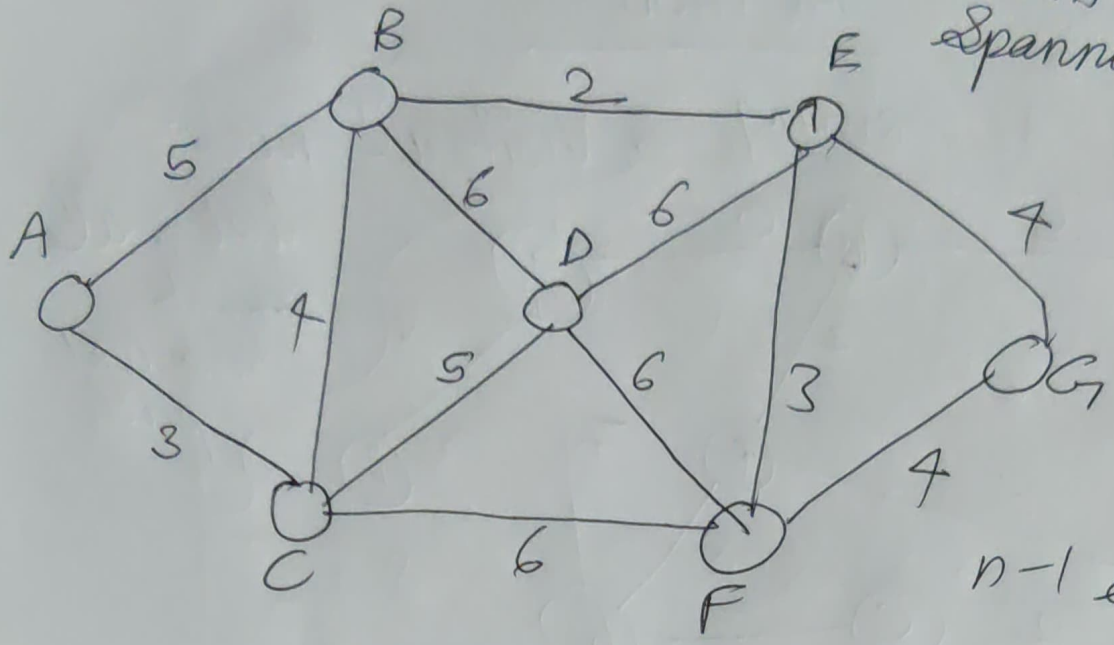
(NO LOOP)

* Start from any vertex
* Choose the lowest edge but remember previously not travelled ~~edge~~ edges.

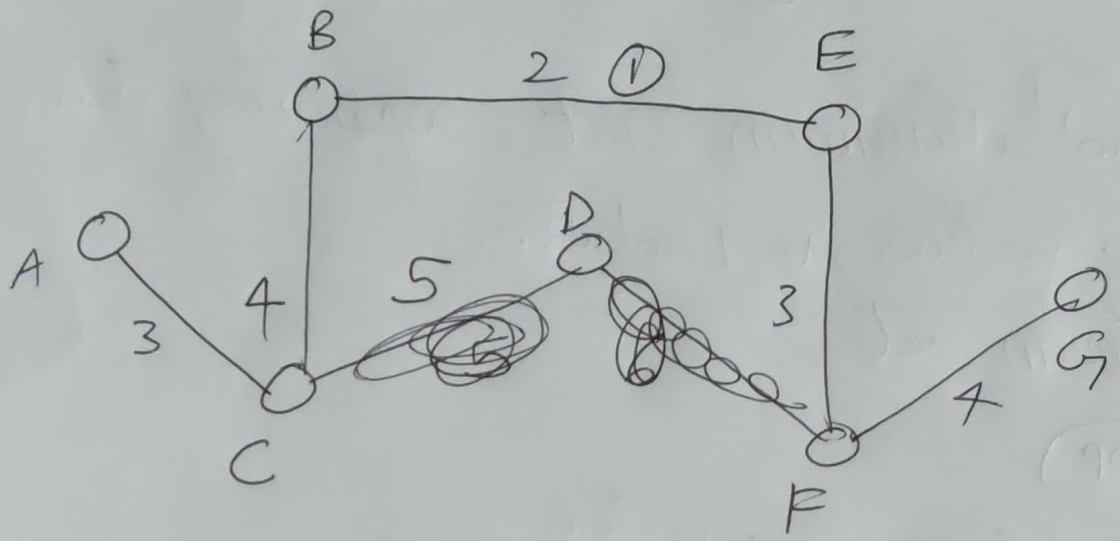


1+5+2+3+7+8+7+3 = 36.

KRUSKAL'S ALGORITHM ⇒ Minimum cost Spanning Tree



n-1 edges



Arrange in increasing order

3+5+6+2+3+4 = 21

Best case = (n-1) edges

Worst case = e edges

DJIKSTRA'S ALGORITHM → used in Google maps, DNA mapping

↳ In multiple nodes, ↳ directed & undirected

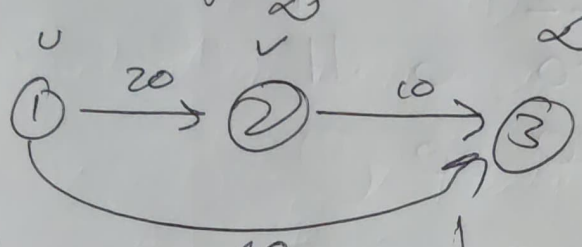
~~the~~ single source shortest path
(minimization algorithm)

if $d(u)+c(u,v) < d(v)$

$d(v) = d(u) + c(u,v)$

initial → initial to final

u will always be the same constant node until v is found, then replace u to check again



Step 1:-

① = u

② = ∞

③ = ∞

$d[u] + c[u,v] < d(v)$

$d(①) + c[①,②] < d[②]$

$0 + 20 < \infty$

$20 < \infty$ ✓

$d[v] = d(u) + c(u,v)$

$d[v] = 20$

Step 2:-

① = u

② = 20

③ = ∞

$d[u] + c(u,v) < d(v)$

$d[①] + c(①,③) < d(③)$

$0 + 40 < \infty$

$40 < \infty$

$d[v] < \infty$

Step 3:-

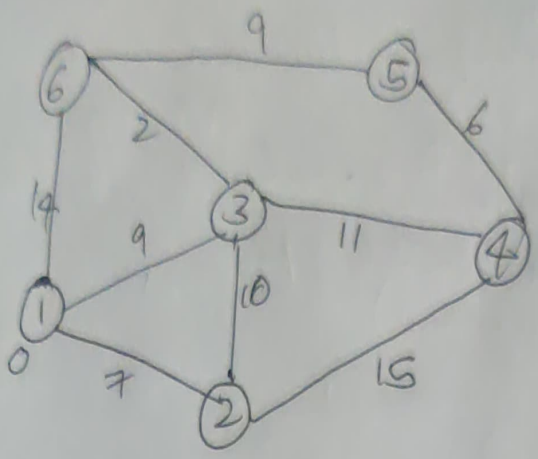
~~d(①) = 20~~ ~~② = 20 = d~~

~~③ = 40 = d(v)~~

$20 + 10 < 40$

$30 < 40$

(30)



Source 1	Destination 2	3	4	5	6	
Links	∞	∞	∞	∞	∞	
1, 2, 3, 6, 4, 5	(7)	9	~~4~~ ∞	∞	14	→ Direct
	(7)	(9)	22	∞	14	
	(7)	(9)	20	∞	(11)	Minimum
	(7)	(9)	20	20	(11)	

Min

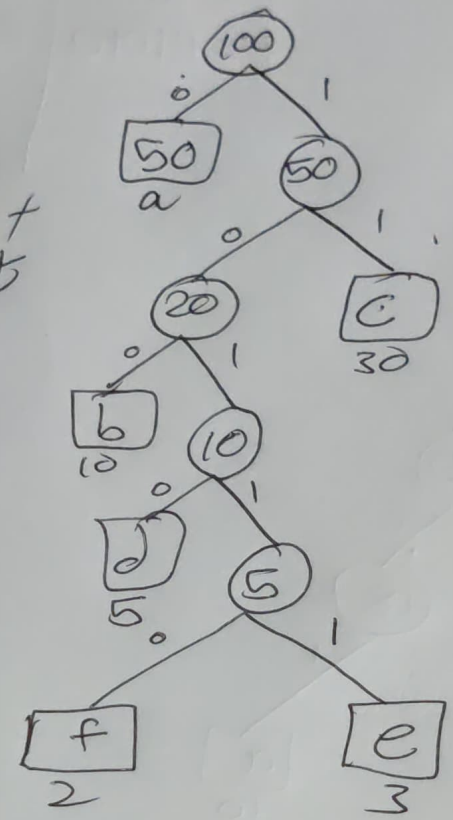
Links est.

HUFFMAN CODING

a = 50
b = 10
c = 30
d = 5
e = 3
f = 2

~~Tree axis~~

Step 1:- Lowest 2, cumulative + = highest



Step 2:- table

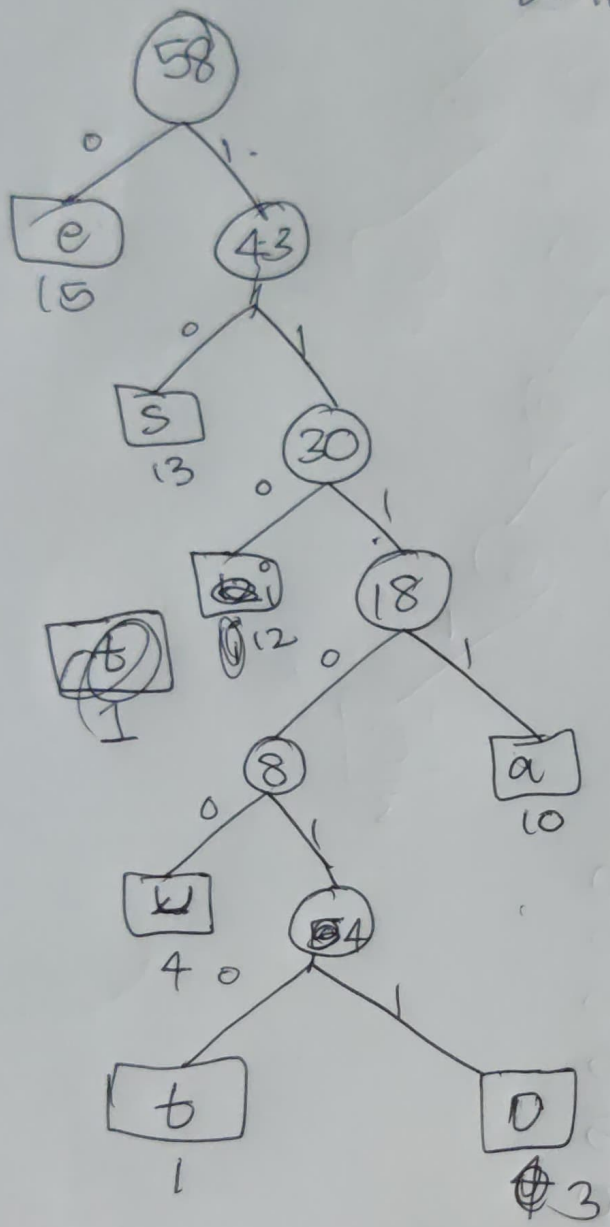
a = 0
b = 100
c = 11
d = ~~H010~~ 1010
e = 101011
f = 101010

50 × 1 = 50
10 × 3 = 30
30 × 2 = 60
5 × 4 = 20
3 × 5 = 15
2 × 5 = 10

185 bits

a = 10
e = 15
i = 12
o = 13
u = 4
s = 13
t = 1

a = 1111	4 x 10	40
e = 0	1 x 15	15
i = 110	3 x 12	36
o = 111011	6 x 3	18
u = 11100	5 x 4	20
s = 10	2 x 13	26
t = 111010	6 x 1	6
		161



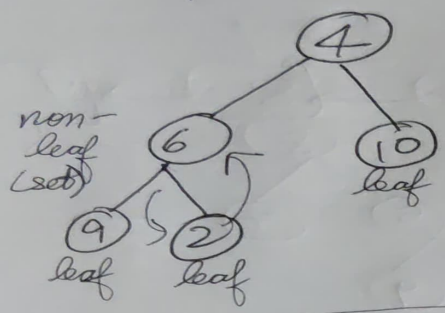
HEAP SORT

HEAP SORT (Inplace, Unstable, O(n log n))

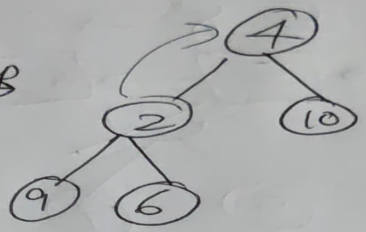
4 6 10 9 2 → 2 4 6 9 10

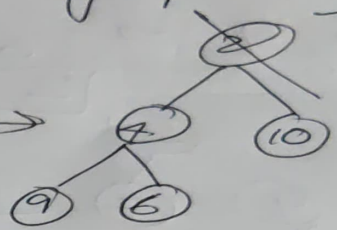
Heap tree

every parent < child

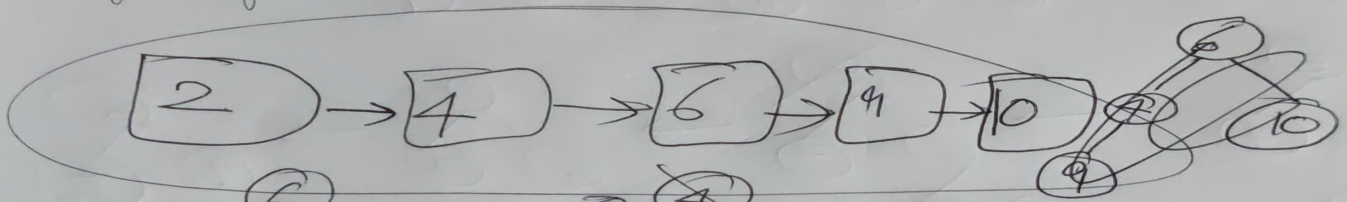


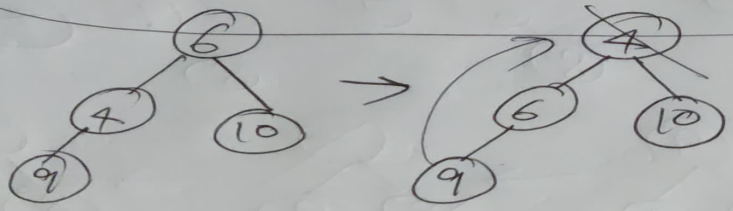
Min heap

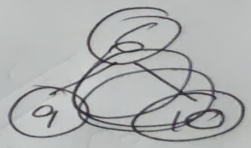


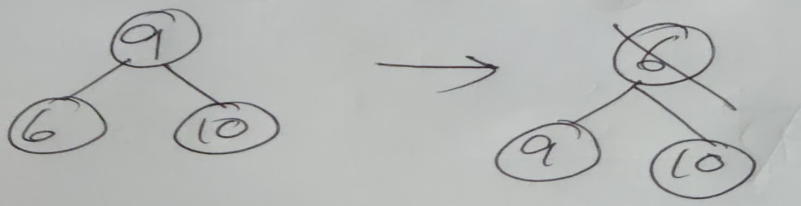


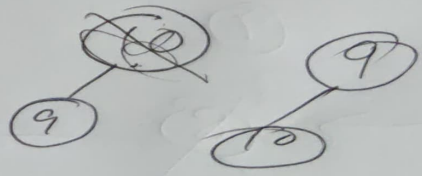
Min. heap





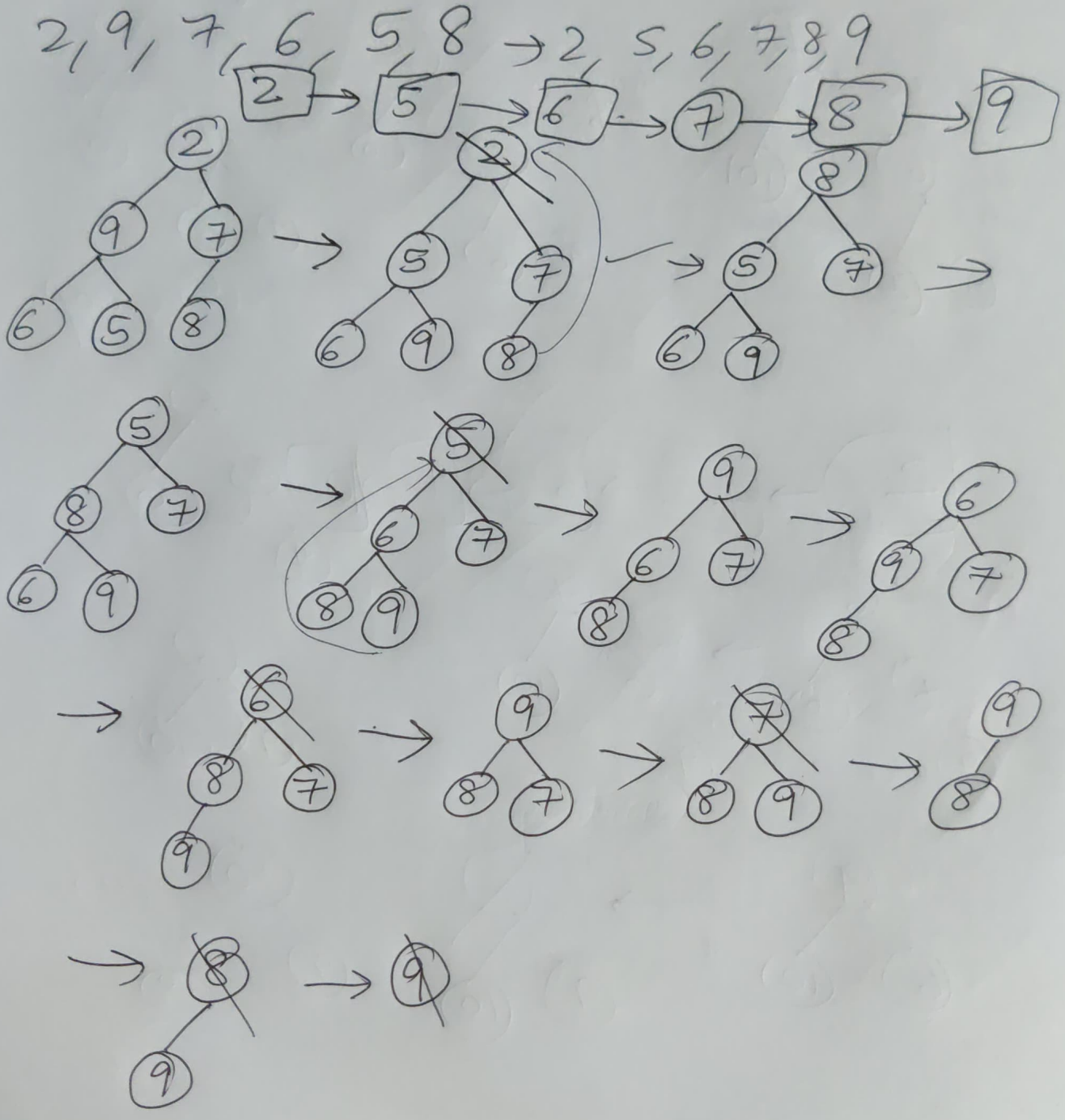






Heap tree steps:-

i) Build a minimum heap tree
ii) Cancel the top-most element
iii) Take the last node (leaf to first) ~~(right leaf)~~
iv) Repeat processes (i → iii)



COIN CHANGE PROBLEM

Given a value V, if we want to make change for V Rs, we have to find the minimum number of coins needed to make the change.

Eg:- coins = {5, 10, 20, 25}
value = 40

Solution	Denominations	Total coins
1	25+10+5	3
2	20+20	2
3	20+10+10	3
4	10+10+10+10	4
5	10+10+5+5+5	6

etc.

The optimal solution is 20+20, ∴ total coins = 2

Eg:- coins = {5, 10, 20, 25}
value = 70

Solution	Denominations	Total coins
1	25+20+20+5	4
2	25+25+20	3
3	20+20+20+10	4
4	25+25+10+10 and so on	4

∴ 3 coins

KNAPSACK GREEDY

$n=4, m=10$

$p=(40, 42, 25, 12)$

$w=(4, 7, 5, 3)$

	O_1	O_2	O_3	O_4
w	4	7	5	3
p	40	42	25	12
$\frac{p_i}{w_i}$	10	6	5	4

O_1, O_2, O_3, O_4

```
for (i=1 to n)
{
  if (m>0 and wi≤m)
  {
    m=m-wi;
    p=p+pi
  }
  else break;
}
if (m>0)
{
  p=p+pi(m/wi)
}
```

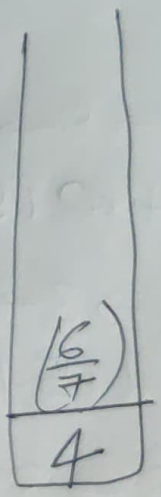
I:- $m=6$

$p=40$

II:- $p=40+42\left(\frac{6}{7}\right)$

$=40+36$

$=76$



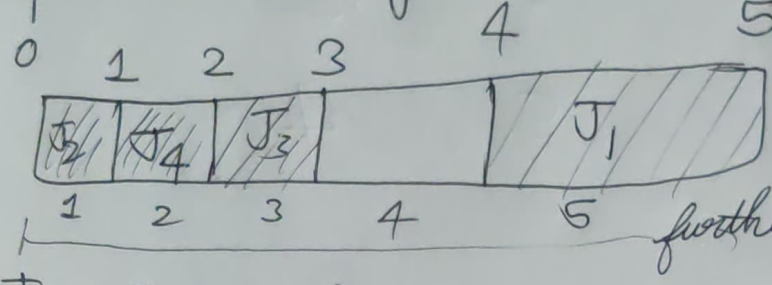
JOB SEQUENCING WITH DEDLINES (14)

	J_1	J_2	J_3	J_4	J_5	J_6
D:	5	3	3	2	4	2
P:	15	10	12	20	8	5

(deadline)

Before deadline, what is the profit?

Purpose → sequencing → maximise the profit



I :- $J_4 \rightarrow 2 \rightarrow 20$

$\Rightarrow 20$

II :- $J_1 \rightarrow 5 \rightarrow 15$

$\Rightarrow 20 + 15$

III :- $J_3 \rightarrow 3 \rightarrow 12$

$20 + 15 + 12$

IV :- $J_2 \rightarrow 3 \rightarrow 10$

$20 + 15 + 12 + 10$

V :- $J_4 \rightarrow 4 \rightarrow 8$

$20 + 15 + 12 + 10 + 8$

$= 65$

~~if occupied also~~

if present again

before ~~anything~~ 9

9 से पहले

The greedy method satisfies some constraints and that either maximizes or minimizes a given function

Control abstraction

```
Algorithm Greedy(a,n)
{
    solution = 0;
    for (i=1 to n)
    {
        x = Select(a);
        if Feasible(solution, x)
        {
            solution = Union(solution, x);
        }
    }
    return solution;
}
```

DIJKSTRA'S ALGORITHM

```
S[] = initialize S[], d[], u, v, i;
while (i<=n)
{
    u = Extract_Min(S, d);
    S[u] = 1;
    for (v=1 to n)
    {
        if (d[u] + c[u][v] < d[v] && S[v]==0)
        {
            d[v] = d[u] + a[u][v];
        }
    }
```

Kruskal's (edge min edge)

```
if (cost[i][j] ==0)
cost[i][j]=999;
while (ne<n)
 {
  for(i=1 to n)
   {
    for(j=1 to n)
     {
      if(cost[i][j]<a min)
       {
        min=cost[i][j];
        a=i;
        b=j;
       }
     }
   }
  ne++;
  mincost+=min;
  cost[a][b]=cost[b][a]=999;
```

Prim's

```
if (cost[i][j]==0)
cost[i][j]=999; visited[1]=1;
 while (ne<n)
  {
   for(i,j)
    {
     if(visited[i]==1)
      {
       min=cost[i][j
       a=i=v;
       b=j=v;
      }
    }
  }
```

```
if(visited[v]==0 || visited[v]==0)
  {
     ne++;
     mincost += min;
     visited[b]=1;
  }
  cost[a][b]=cost[b][a]=999;
 }
}
```