1) What is Alogorithm ?
It is a step by step procedure to solve a computational program/problem & it should be unambiguous
Example 2 wheeler bike (instruction manual)

2) Program :- set of instructions
Types of instruction or statements
i) Declaration statement
ii) Assignment or initialization
iii) conditional
iv) Return
v) Repetative
vi) continue & break
vii) Functions
viii) Prototype declaration

3) Difference b/w Alogrith & program

Algorithm	Program
• It is a designed type	• It's a implementation type
• Study abt domain knowledge	• Study about programmer
• It can be written in any languages	• It can be written in programming language
• Hardware & o/s is independant	• H/W & o/s are dependand
• Analyse	• Testing
• Algorithm is also called as priori analysis	• Postorio analysis

4) Charcteristics of algorithm
i/p, o/p, definiteness, finiteness, effectiveness

5) Analysis of algorithm.
- Time
- space
- network
- CPU
- register
- power

→ knap sack problem

knapsack is also called as bag or container capacity - M, no of objects = n,
weights $w_1, w_2, w_3 \ldots w_n$
profits $p_1, p_2, p_3 \ldots p_n$
fractions $= x_1, x_2, x_3 \ldots x_n$
to be added to the knap sack

Objective :- place the objects into the knap-sack so that max profit is obtained & weight of the object should not exceed the capacity of knapsack

Maximise: $\sum_{i=1}^{n} p_i \times x_i$ (optimization func)

constraints: $\sum_{i=1}^{n} w_i x_i \leq M$

1. Obtain the optimal solution for the knap sack using greedy method. given
$M = 40 \quad N = 3$
$(w_1\ w_2\ w_3) = (20, 25, 10)$
$(P_1, P_2, P_3) = (30, 40, 35)$

$P_1 / w_1 = 30/20 = 1.5$
$P_2 / w_2 = 40/25 = 1.6$
$P_3 / w_3 = 35/10 = 3.5$

$P_i / w_i \quad 3.5 > 1.6 > 1.5$

p_i	35	40	30
w_i	10	25	20

	Object	w_i	p_i	$x = 1$ or rc/w_i	$x * p_i$	$rc = rc - w_i x$
initial						40
step 1	1	10	35	$40/10 = 4 > 1$	$1 \times 35 = 35$	$rc = 40 - 10 \times 1 = 30$
step 2	2	25	40	$30/25 = 1.2 > 1$	$1 \times 40 = 40$	$30 - 25 \times 1$ 5
step 3	3	30	30	$5/20$ $0.25 < 1$	0.25×30 7.5	$5 - 20 \times 0.25$ 0

Profit $= 35 + 40 + 7.5 = 82.5$
Fraction $= (1, 1, 0.25)$
weight $= (10, 25, 20)$

2. $m=15$

$n=7$

$(P_1, P_2, P_3, P_4, P_5, P_6, P_7) = (10, 5, 15, 7, 6, 18, 3)$

$(w_1, w_2, w_3, w_4, w_5, w_6, w_7) = (2, 3, 5, 7, 1, 4, 1)$

P_i/w_i

$P_1/w_1 = 10/2 = 5$

$P_2/w_2 = 5/3 = 1.6$

$P_3/w_3 = 15/5 = 3$

$P_4/w_4 = 7/7 = 1$

$P_5/w_5 = 6/1 = 6$

$P_6/w_6 = 18/4 = 4.5$

$P_7/w_7 = 3/1 = 3$

P_i/w_i	6 >	5 >	4.5 >	3 >	3 >	1.6 >	1
P_i	6	10	18	15	3	5	7
w_i	1	2	4	5	1	3	7

	Object	w_i	p_i	$x=1$ r rc/w_i	profit $x * p_i$	$rc - w_i x$ $rc=15$
Step 1	1	1	6	$15/1 = 15$	$1 \times 6 = 6$	$15 - 1 \times 1 = 14$
Step 2	2	2	10	$14/2 = 7 > 1$	$1 \times 10 = 10$	$14 - 2 \times 1 = 12$
Step 3	3	4	8	$12/4 = 3 > 1$	$1 \times 18 = 18$	$12 - 4 \times 1 = 8$
Step 4	4	5	15	$8/5$ $1.6 > 1$	$1 \times 15 = 15$	$8 - 5 \times 1 = 3$

Obj.	Bu1					
p5	5	1	3	$3/1 = 3 > 1$	$1 \times 3 = 3$	$rc = 3-1 \times 1 = 2$
p6	6	3	5	$2/3 = 0.66 < 1$	$0.66 \times 5 = \cancel{3}3.33$	$rc \rightarrow 3 \times 0.66 = 0$
p7	7	7	7	0	0	0

Profit $= 6 + 10 + 8 + 15 + 3 + 3.3 = 55.3$

Fraction $(1, 1, 1, 1, 1, 0.66, 0)$.

3. $m = 20$

$w_1, w_2, w_3 \quad (18, 15, 10)$

$P_1, P_2, P_3 \quad (30, 21, 18)$

Find P_i/w_i

$w_1 \quad P_1/w_1 = 30/18 = 1.6$

$P_2/w_2 = 21/15 = 1.46$

$P_3/w_3 = 18/10 = 1.8$

$P_i/w_i \quad 1.8 > 1.6 > 1.46$

P_i	18	30	~~30~~ 21
w_i	10	18	15

Object	w_i	P_i	$x = 1$ or rc/w_i	Profit $x \times P_i$	$rc = rc - w_i * x$
initially					$rc = 20$
1	10	18	$20/10 = 2 > 1$	$1 \times 18 = 18$	$20 - 10 * 1$
2	15	21	$10/18 = 0.5 > 1$	$0.5 \times 30 = 16.5$	$10 - 18 * 0.5$
3	15	21	$\frac{0.1}{15} = \frac{0.0066}{21}$	$0.0066 \times 21 = 0.1386$	$rc = 0.15 \times -w_i$, 0.005, 21×10^{-3}

Profit $= 18 + 16.5 + 0.1386$

$= 34.6$

Fraction $= 2, 0.55, 0.0066$

weight $= 10, 18, 15$

4. $m = 20$

$w_1 \ w_2 \ w_3 \ (25, 24, 15)$

$P_1 \ P_2 \ P_3 \ (18, 15, 10)$

Find P_i/w_i

$P_1/w_1 = 18/25 = 0.72$

$P_2/w_2 = 15/24 = 0.625$

$P_3/w_3 = 10/15 = 0.666$

$P_i/w_i = 0.72 > 0.666 > 0.625$

$P_i = 18 \quad 10 \quad 15$

$w_i = 25 \quad 15 \quad 24$

Object	w_i	P_i	$x = 1$ or rc/w_i	Profit $x \times P_i$	$rc = rc - w_i \times x$
initial					$rc = 20$
1	25	18	$\frac{20}{25}$ 0.8	0.8×18 14.4	$20 - 25 \times 0.8 = 0$

Profit = 14.4

$w_i = 25, 15, 24$

$frac = 0.8, 0, 0$

URBAN EDGE

→ Job sequencing in deadline.

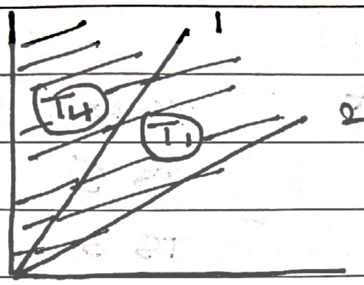
n = set of jobs
d = deadline for ith job
p = profit for ith job
Find soln of jobs such that all the choosen jobs should be completed within their deadlines & the profit earned should be maximum with following constrains.
Only one machine is available
Only one job must be processed for a given time. A job is said to be completed within deadline and task
Feasible soln :- set of jobs should be completed within deadlines.
Optimal soln :- It should result in max profit

→ Problems

1. Obtain the optimal soln for job sequencing problem with deadlines with n=4

i	1	2	3	4
P	100	10	15	27
d	2	1	2	1

Task	~~Profit~~ Deadline	Profit	TSA
T1	2	100	T1
T4	1	27	T4
T3	2	15	not pos
T2	1	10	not pos

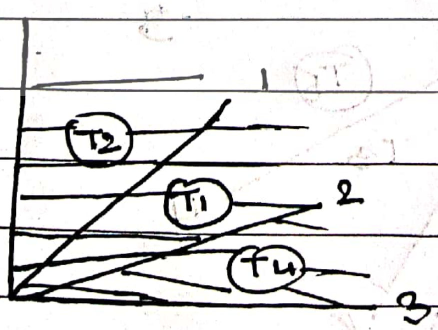


Profit = T1 + T4
100 + 27
127

2.

i	1	2	3	4	5
P	20	15	10	5	1
d	2	2	1	3	3

Task	Deadline	Profit	TSA
T1	2	20	T2
T2	2	15	T1
T3	1	10	not pos
T4	3	5	T3
T5	3	1	not pos

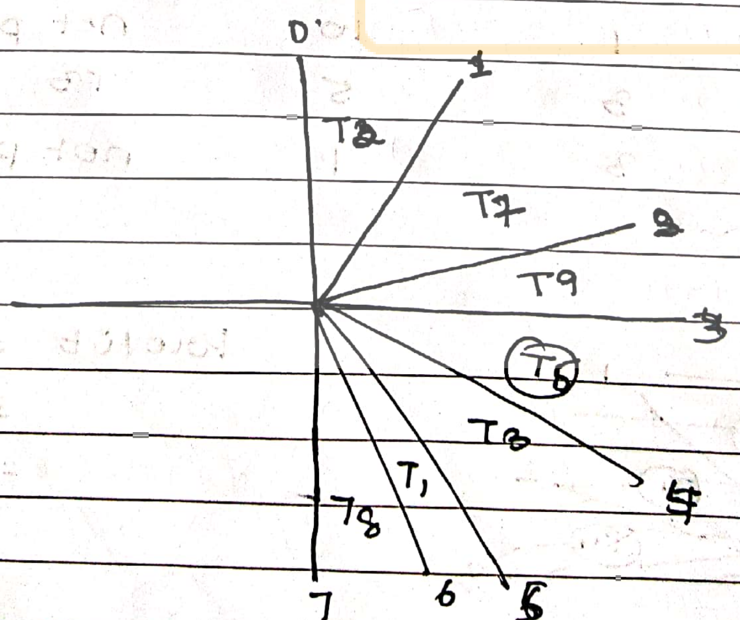


Profit = T2 + T1 + T3
= 20 + 15 + 5
= 40

3) Find the soln to job sequencing problem given below

i	1	2	3	4	5	6	7	8	9
di	7	2	5	3	4	5	2	7	3
Pi	15	20	30	18	18	10	23	16	25

Task	deadline	Profit	TSA
T_1	5	30	TS 5
T2	3	25	TS 3
T3	2	23	TS 1
T4	2	20	NP
T5	3	18	
T6	4	18	
T7	7	16	
T8	7	15	
T9	5	10	



→ Kruskal's Algorithm

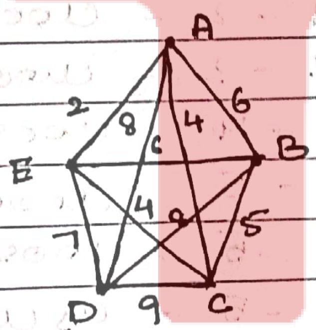
To find minimal spanning tree of a graph with n nodes.

Step 1: Choose the edge of least weight

Step 2: Among these edges choose the remaining edge with least weight & should not form a cycle. (conflict resolution criteria). If there are more edges eligible for selection you can choose any edge among them

Step 3: Repeat step 2 until n-1 edges have been choosen

1. Find the MSP using Kruskal Alg.

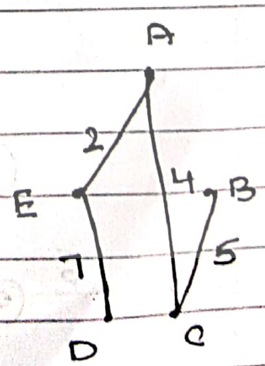


nodes = 5

n − 1 = 4

total edges = 10

AE	2	choose
AC	4	choose
EC	4	cycle
BC	5	choose
AB	6	cycle
BE	6	cycle
DE	7	choose
AD	8	cycle
DC	9	cycle

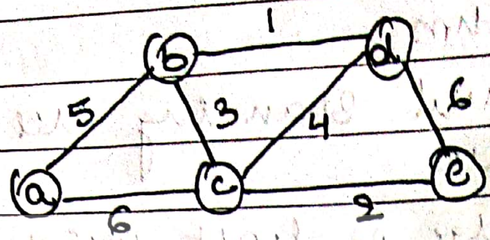


MST = AE + AC + BC + ED

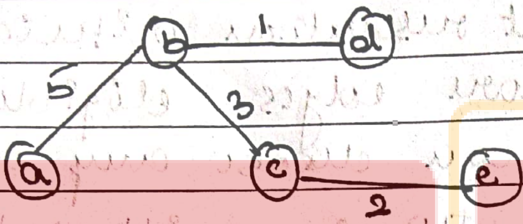
2 + 4 + 5 + 7

18

2)

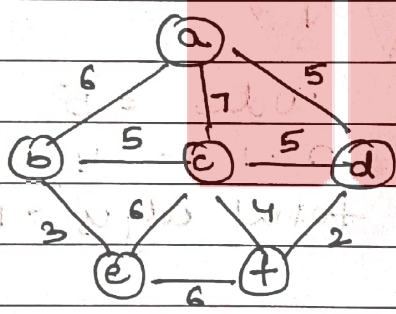


nodes . 5
n - 1   4
total edges 7.

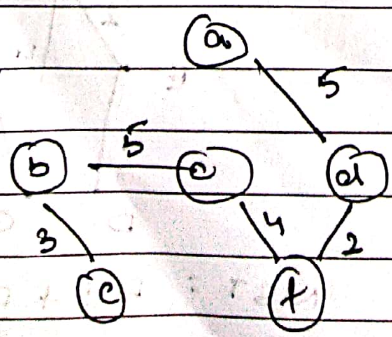


MST = 5 + 3 + 2 + 1
= 11

3)



fd - 2 - choose
bc - 3 - choose
cf - 4 - choose
bc - 5 - choose
ad - 5 - choose
cd - 5 - cycle
ab - 6 - cycle
ef - 6 - cycle
ce - 6 - cycle
ac - 7 - cycle



bc + bc + cf + fd + ad
3 + 5 + 4 + 2 + 5
19

→ Prim's algorithm

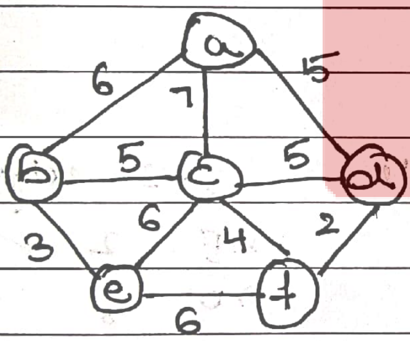
Step 1: Select any node to be the first node of 'T'

Step 2: Consider the edge which connects & connects the node in T, nodes should not over right. Pick anyone with min weight to avoid cycle, its called as conflict resolution criteria. If more than one edge is available choose any.

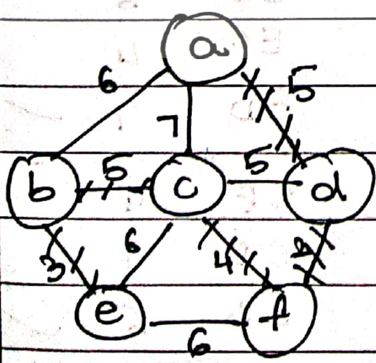
Step 3: Repeat step 2 until T contains every node in the graph.

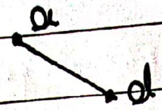
→ Problems

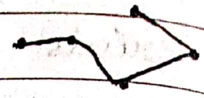
1. Find minimum spanning tree by Prim's Alg.

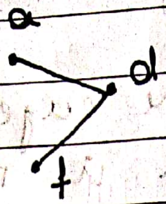


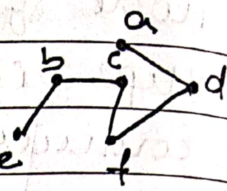
Set node	Pos edges in asc order
a	ad - 5 ✓
	ac - 7
	ab - 6
a, d	dc - 5
	df - 2 ✓
a, d, f	fc - 4 ✓
	fe - 6
a, d, f, c	ca - 7
	cb - 6 ✓
	ce - 6
a, d, f, c, b	be - 3
	ba = 6
a, d, f, c, b, e	

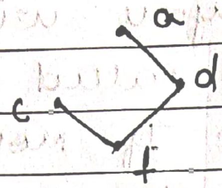


Step 1 

Step 4 

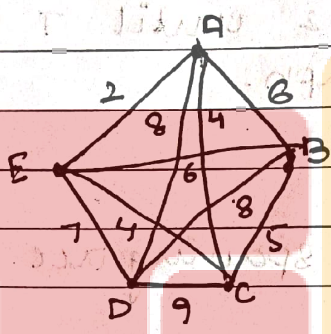
Step 2 

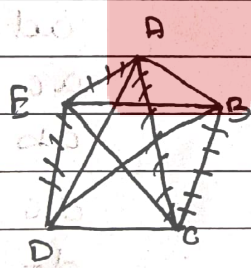
Step 5 

Step 3 

ad + df + fc + cb + b

5 + 2 + 4 + 5 + 3 =

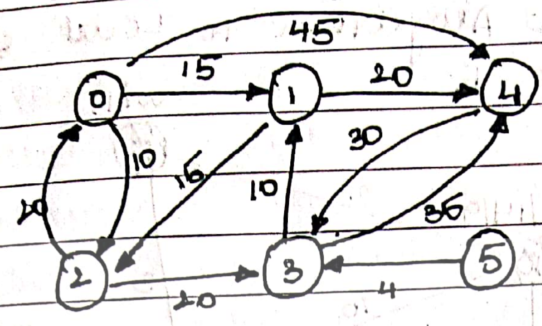
2) 



Set node	pos edges	
A	AE	2
	AD	8
	AC	4
	AB	6
A, E	EB	6
	EC	4
	ED	7
A, E, C	CE	4
	CD	9
	CB	5
A, E, C, B	BA	6
	BE	6
	BD	8
A, E, C, B, D	DB	8
	DA	8
	DC	9

Dijkstra's Algorithm

1)



	0	1	2	3	4	5
0	0	15	10	∞	45	∞
1	∞	0	15	∞	20	∞
2	20	∞	0	20	∞	∞
3	∞	10	∞	0	35	∞
4	∞	∞	∞	30	0	∞
5	∞	∞	∞	4	∞	0

source vertex: 5

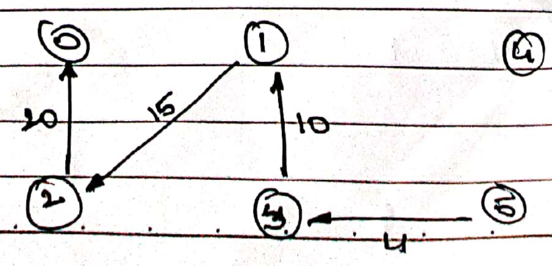
Distance from vertex i to i is 0 a[i][i] = 0

If there is no path a[i][j] = ∞

d[0] = ∞ d[1] = ∞ d[2] = ∞ d[3] = 4 d[4] = ∞

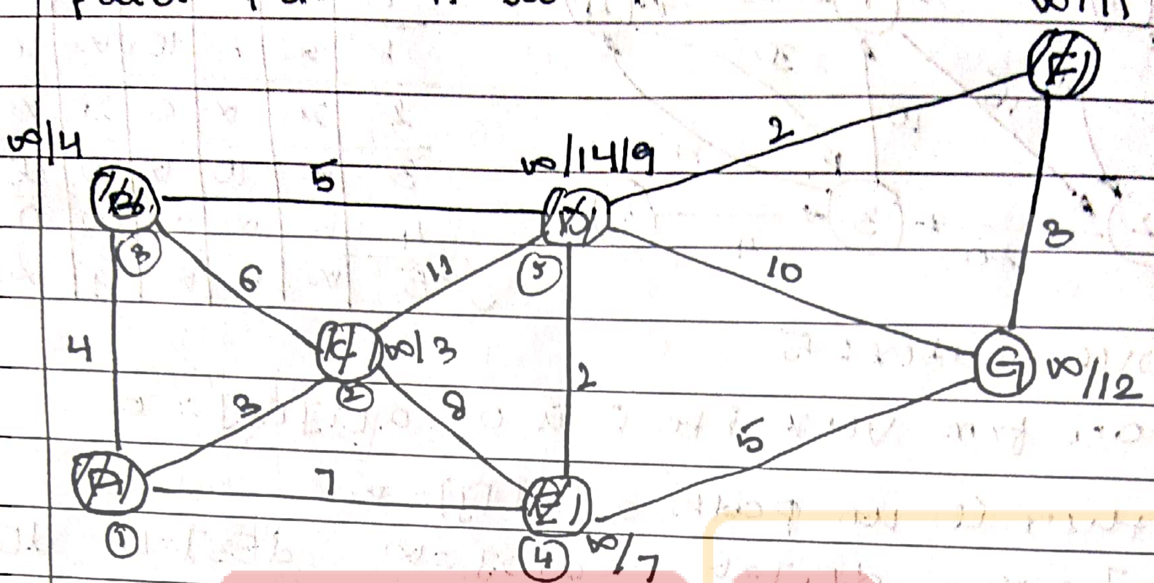
d[5] = 0

S	remaining	d[v] = min(d[v], d[u] + cost[u][v])	u	d[u]
5	0,1,2,3,4	- - - - - - - - - - - -	3	4
5, 3	0,1,2,4	d[0] min(∞, 4 + [3][0]) = ∞ d[1] = min(∞, 4+10) = 14 d[2] min(∞, 4+∞) = ∞ d[4] = min(∞, 4+35) = 39	1	14
5,3,1	0,2,4	d[0] = min(∞, 14+∞) = ∞ d[2] = min(∞, 14+15) = 29 d[4] = min(39, 14+20) = 34	2	29
5,3,1,2	0,4	d[0] = min(∞, 29+20) = 49 d[4] = min(34, 29+∞) 34	4	34
5,3,1,2,4	0	d[0] = min(49, 34+∞) = 49	0	49
5,3,1,2,0,4				



→ Dijkstraw using Elimination method

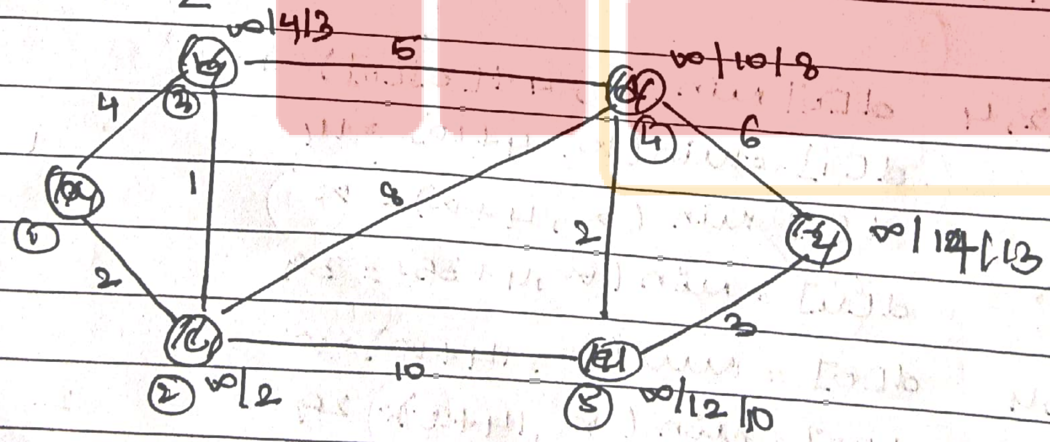
1) use Dijkstraw Alogrithm to find shortest path from A to F



A → C → B → E → D → F

Total A → F = 11

2) A to Z



A → C → B → D → E → Z

Total A → Z = 13

→ Huffman Tree
sequence of bits is codewords.

Huffman Alg :- step 1 : Initialize n, node tree & label them with character of the alphabet, record freq of each character in the root to indicate tree's weight.

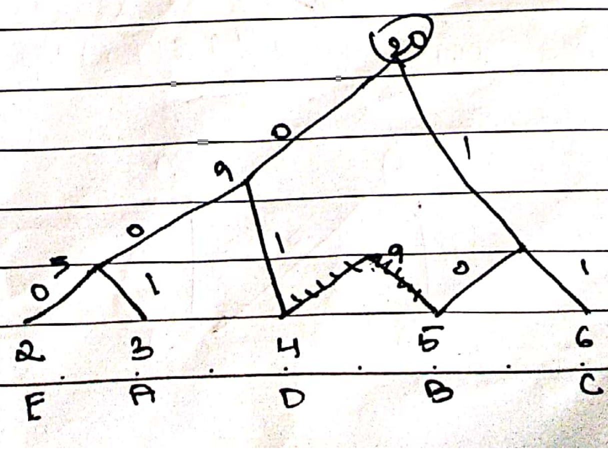
step 2: Repeat following operation until the 2nd tree is obtained. Find 2 trees in smallest weight. Make them left & right subtree of a new tree and record sum of their weights in the root of new tree as its weight.

1. consider 5 diff character alphabets in the following occurance probility & construct Huffman tree.

BCCABBDDAECCBBAEDDCC

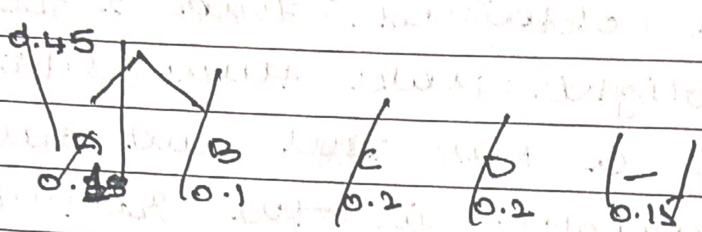
char	count	code	
E	2	000	3×2 = 6
A	3	001	3×3 = 9
D	4	01	4×2 = 8
B	5	10	5×2 = 10
C	6	11	6×2 = 12
5×8 40	20 bits	12 codes	45 bits

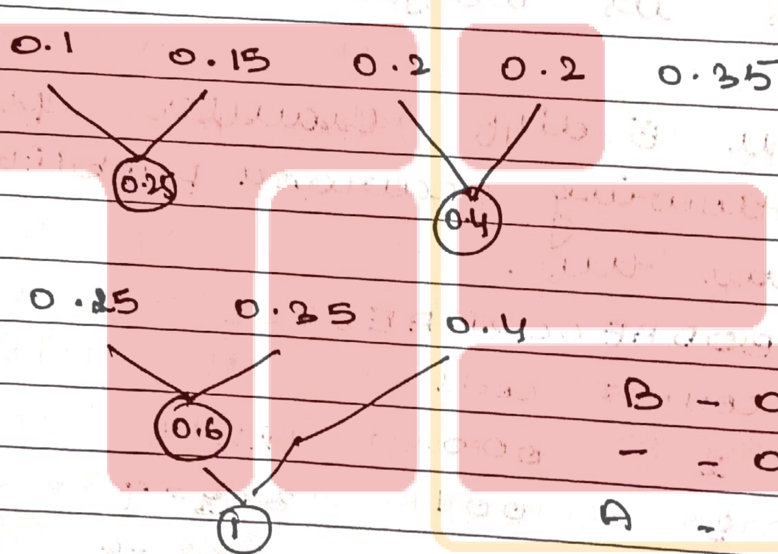
Msg 45 bits
count = 20
char + code = 52



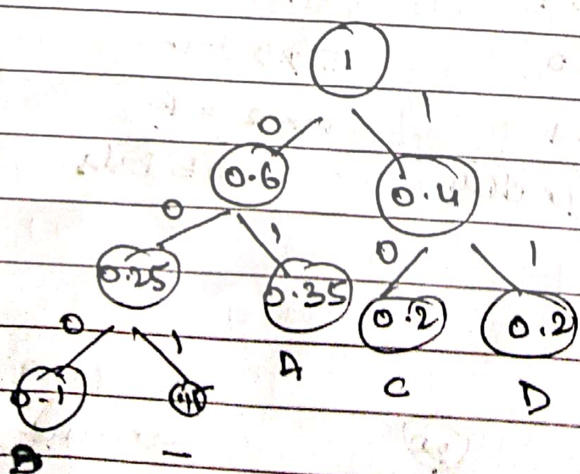
2. Consider 5 characters of alphabet (A, B, C, D, _)

char	A	B	C	D	_
Prob	0.35	0.1	0.2	0.2	0.15



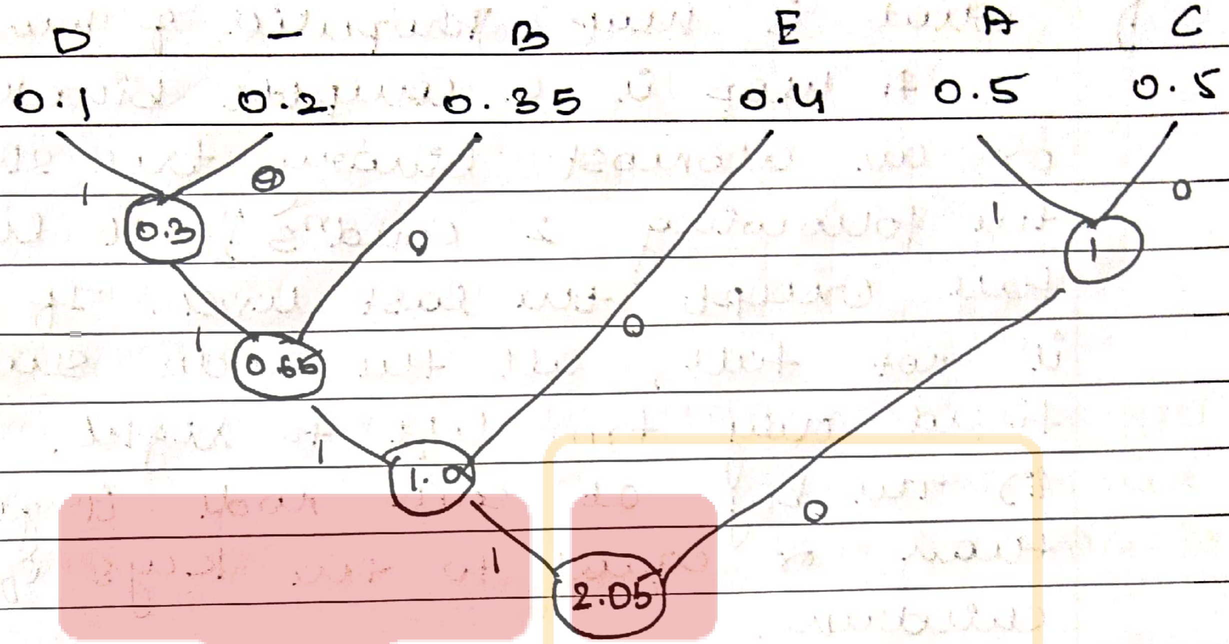


B - 000
_ - 001
A - 01
C - 10
D - 11



3.

A	B	C	D	E	_
0.5	0.35	0.5	0.1	0.4	0.2



D – 1111
_ 1110
B 1110
E 110
A 01
C 00

Heaps & Heap sort

what is heap? Properties of heap.

A heap is a complete binary tree or an almost binary tree satisfying the following 2 cond^ns i) all levels are full except the last level. If last level is not full, all the nodes should be filled only from left to right.

ii) The key at each node is greater than or equal to the keys of its children.

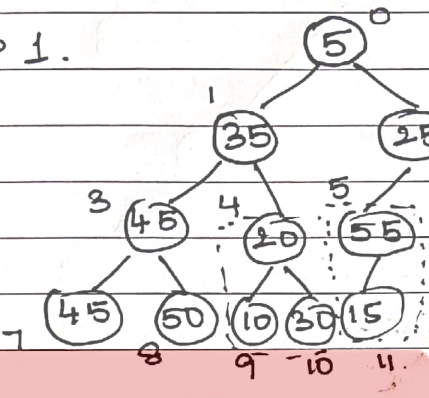
Properties of Heap

- The heap must be a complete binary tree or almost complete binary tree. Its height is always $\log_2 n$.
- The root of the tree always contains highest element.
- Each subtree must be a heap. i.e children of a node must be less than the parent.
- In a heap all the nodes are numbered level from left to right
- Given any node "i", position of left child is given by $2i+1$ & position of right child is given by $2i+2$.
- Given any node 'i' the parent position is given by ~~i-1/2~~ $\frac{i-1}{2}$

Problems

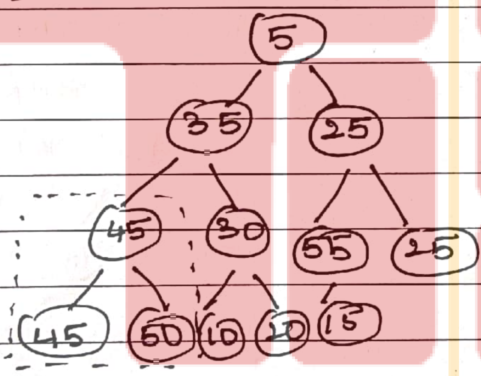
1. Create a bottom up heap construction for the elements 5, 35, 25, 45, 20, 55, 25, 45, 50, 10, 30, 15

Step 1.



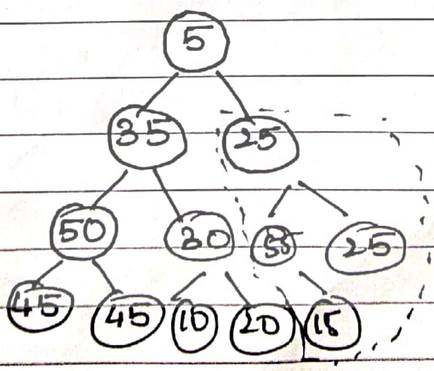
// here 55 is greater than 15 hence it remains as it is

Step 2



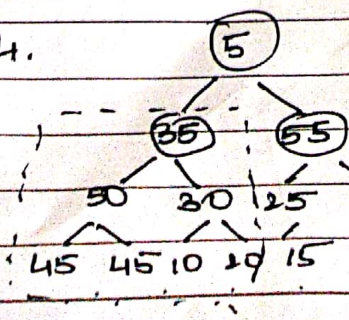
// here 50 is greater than 45 hence it is gonna swap.

Step 3

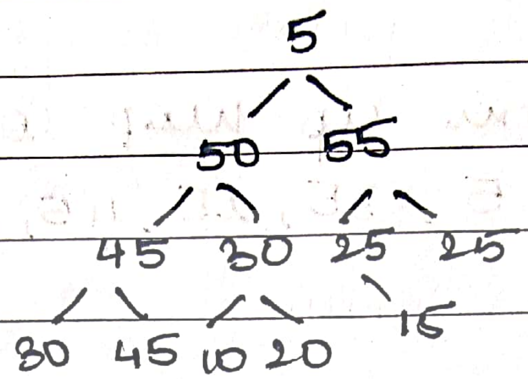


// Here 55 is greater hence 25 is gonna swap with 55

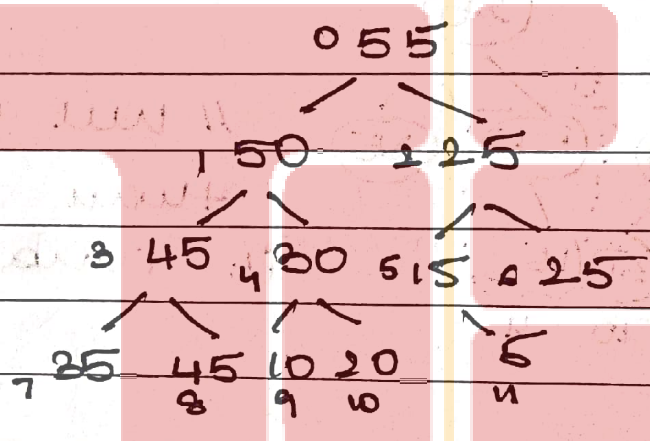
Step4.



Step 5

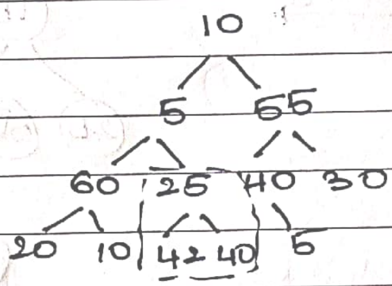

5
50 55
45 30 25 25
30 45 10 20 15

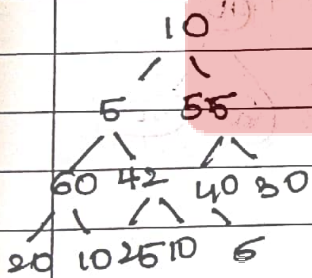
Step 6


0 55
1 50 2 25
3 45 4 30 5 15 6 25
7 35 8 45 9 10 10 20 11 5

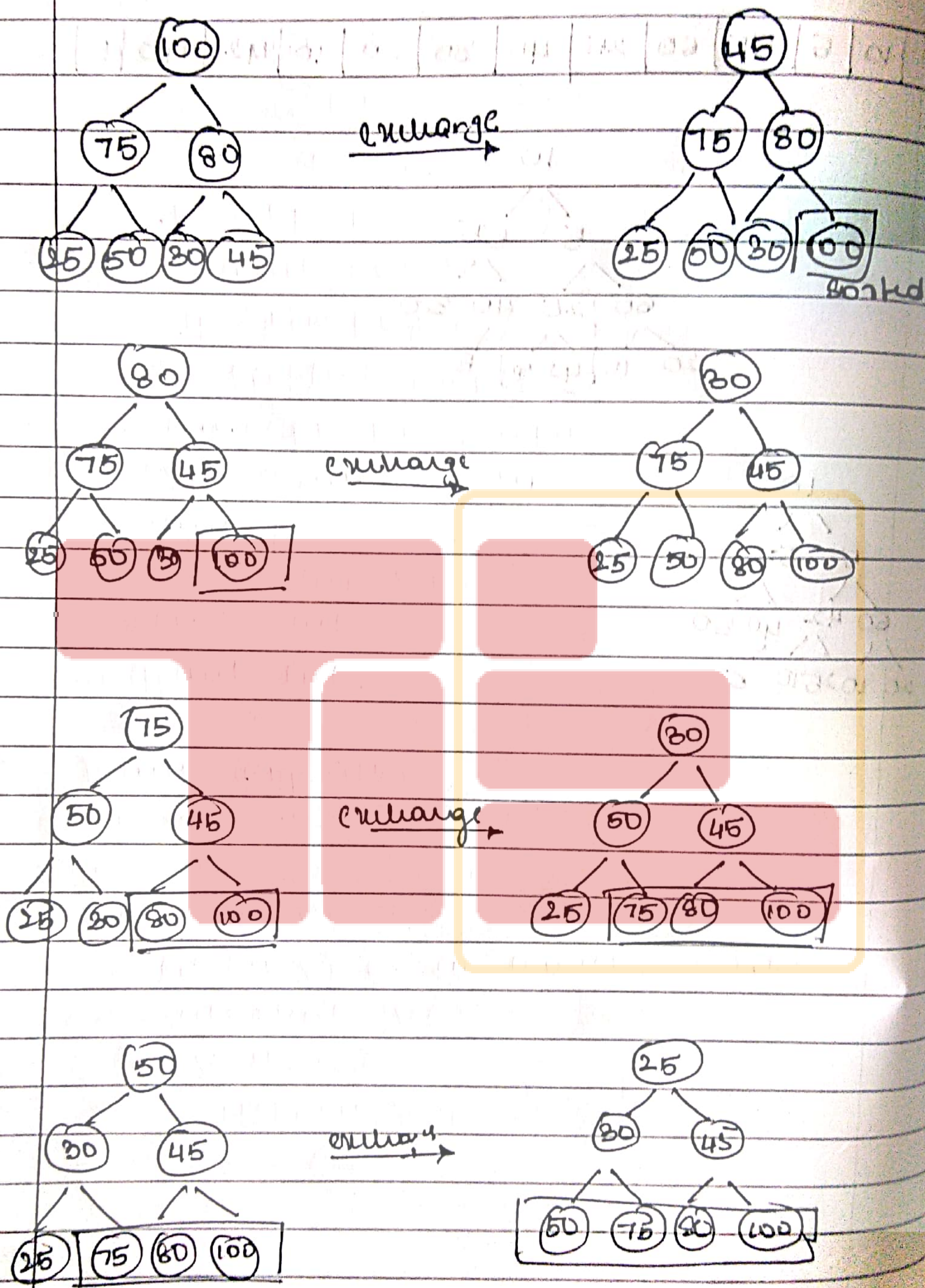
2. 10, 5, 55, 60, 25, 40, 30, 20, 10, 42, 40, 5

10	5	55	60	25	40	30	20	10	42	40	5

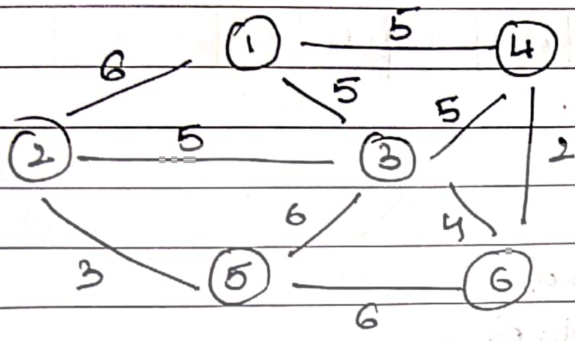




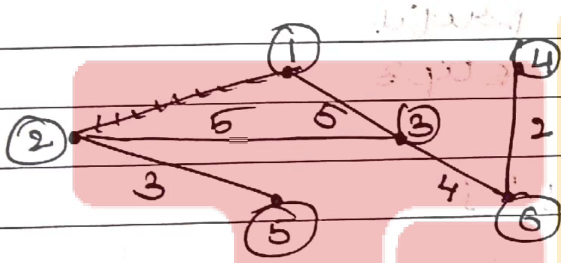
→ Tracing of heap sort



→ Kruskal continued



4 6 choose
3 6 choose
3 1 choose
2 3 choose
2 5 choose
1 4 cycle
3 4 cycle
2 3 cycle
5 6 cycle
2 1 cycle



Total $3+5+5+4+2 = 19$

→ Algorithms.

i) Knapsack

// purpose : To find the solution vector that shows the fraction of the object selected

i/p : m, n, w Initialization

o/p : x

for i ← 1 to n
  x[i] ← 0
end for.
rc ← m
for i ← 1 to n
if i ← 1 to n
  if (w[i] > rc) break //ith object can be selected
  x[i] = 1 // select the ith object
  rc = rc = w[i] // find the remaining capacity of knapsack
end for
  if (i <= n) x[i] = rc/wi

2) Job sequencing with Deadline

Initialize

Profit = 0

for i = 1 to n

do

k = d[i]

while (k > 0)

If (list[k] = 0)

list[k] = job[i]

add p[i] to profit

go back to step 3

end if

Decrement by 1;

end while

print list.

3) Prims Algorithm

$T = \Phi$

$U = \{1\}$

while ($U \neq V$)

let (u, v) be the lowest cost edge

such that $v \in U$ & $v \in V$

$\in V - U$;

$T = T \cup \{(u, v)\}$

$V = V \cup \{v\}$

4) Kruskal Algorithm

Input : A weight connected graph G(V, E)

Output : ET, the set of edges composing a MST of G sort E in non decreasing order of the edge weight $w(e_{i1}) \leq \ldots \leq w(e_{i|E|})$

ET ← ϕ ; encounter ← 0 // initialise the set of tree edges & its size

k ← 0 // Initialize the number of processed edges

while encounter < |V| - 1 do

    k ← k+1

    if ET ∪ {eik} is acyclic

    ET ← ET ∪ {eik}; encounter ← encounter+1;

return ET.

5) Dijkstra's algorithm

Input : n - no of vertex

    w - cost of weight

    source - source vertex

    destination - dest vertex

Output : d - shortest dist b/w source & other vertex

    ϕ - shortest path

    s - no of nodes visited.

for i = 0 to n-1

    d[i] = cost [source, i]

    ϕ [i] = source

    s[i] = 0

end for

s[source] = 1

```
for i = 1 to n - 1
do
Find u, d[u], where d[u] is min
add u to r
if (u, destination) break
for every v ∈ vs
do
  if (d[u] + w[u, v] < d[v])
        d[v] = d[u] + w[u, v]
        ϕ[v] = u
      end if
    end for
end for
```

6) Heap sort algorithm

```
for i = (n-1)/2 down to 0
item = a[p]
c = 2p + 1
while (c <= n - 1) do
if (c+1 <= n-1 and a[c] < a[c+1])
  c = c+1
end if
if (item < a[c])
  a[p] = a[c]
  p = c
  c = 2p+1
else
    break
end if
end while
a[p] = item
end for
```