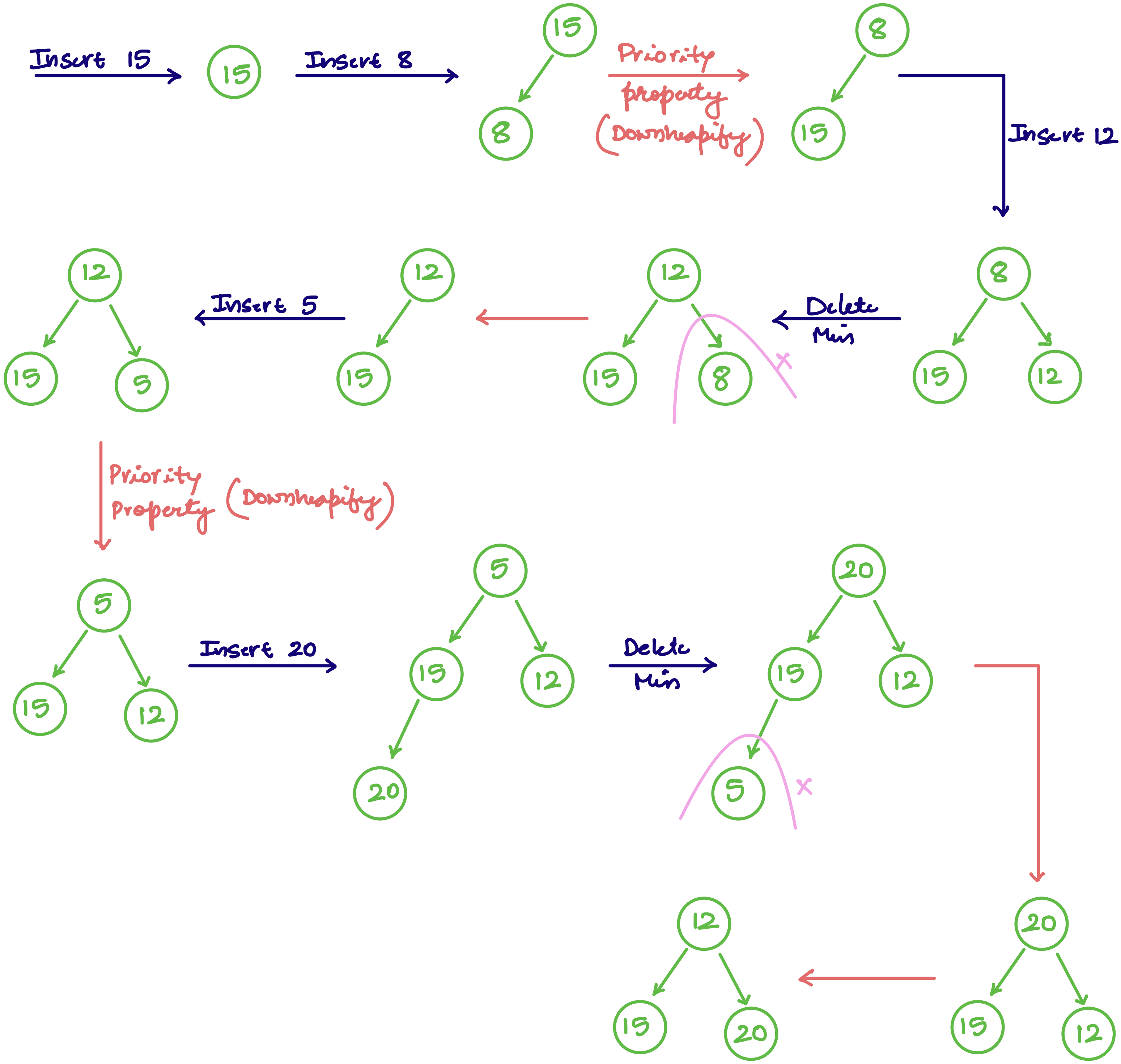
Ques 1:

a). min-heap

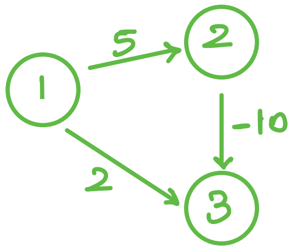


b). optimal substructure :

optimal solution to the overall problem can be constructed from optimal solution of its smaller subproblem.

c). Yes it will work. Assume adding a very large constant to every edge weight to make them all +ve. This approach doesnot change the resulting MST.

d).



Dijkstra gives result from $1 \to 3$ as 2 instead it will be $5 - 10 = -5$

e). $T(n) = 2T(\sqrt{n}) + \log n$

Let $n = 2^m \Rightarrow \log n = m$

$T(2^m) = 2T(2^{m/2}) + m$

Let $T(2^m) = S(m)$

$S(m) = 2S(m/2) + m$

Using Masters Theorem,

$a = 2 \quad b = 2 \quad k = 1 \quad p = 0$

$a = b^k$ and $p > -1$

$S(m) = \Theta(m^{\log_b^a} \log^{p+1} m) = \Theta(m^{\log_2^2} \log^1 m) = \Theta(m \log m)$

$T(2^m) = \Theta(m \log m)$

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

Ques 2 a).

i).

```
int main()
{
    TOH(3, "S", "D", "H") ;
    return 0 ;
}

void TOH(int n, string src, string dst, string helper)
{
    if(n == 0)
        return ;

    TOH(n-1, src, helper, dst) ;
    cout << "Move disc " << n << " from " << src << " to " << dst << endl ;
    TOH(n-1, helper, dst, src) ;
}
```

ii).

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

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

$$\vdots$$

$$T(1) = 1$$

$$\Downarrow$$

$$T(n) = 2T(\cancel{n-1}) + 1$$

$$2T(\cancel{n-1}) = 2^2 T(\cancel{n-2}) + 2$$

$$2^2 T(\cancel{n-2}) = 2^3 T(\cancel{n-3}) + 2^2$$

$$\vdots$$

$$2^{n-1} T(\cancel{n-(n-1)}) = 2^{n-1}$$

$$T(n) = 1 + 2 + 2^2 + \cdots\cdots + 2^{n-1}$$

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

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

iii). no of moves required = $2^n - 1$

if $n = 3$ then moves $= 7$

if $n = 4$ then moves $= 15$

Ques 2b).

Sort on bases of finish time

Activity	A1	A3	A2	A4	A6	A5	A7	A9	A8	A10
Start	1	3	2	4	8	7	9	11	9	12
Finish	3	4	5	7	9	10	11	12	13	14
	✓	✓	×	✓	✓	×	✓	✓	×	✓

A1, A3, A4, A6, A7, A9, A10

Ques 3

a).

	I_1	I_2	I_3	I_4	I_5
Profit	12	10	20	22	25
Weight	3	4	7	8	9

cap: 15

capacity →

Items ↓	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
I_1	0	0	0	12	12	12	12	22	22	25	32	34	37	37	42	44
I_2	0	0	0	0	10	10	10	20	22	25	25	30	32	35	35	42
I_3	0	0	0	0	0	0	0	20	22	25	25	25	25	25	25	42
I_4	0	0	0	0	0	0	0	0	22	25	25	25	25	25	25	25
I_5	0	0	0	0	0	0	0	0	0	25	25	25	25	25	25	25
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

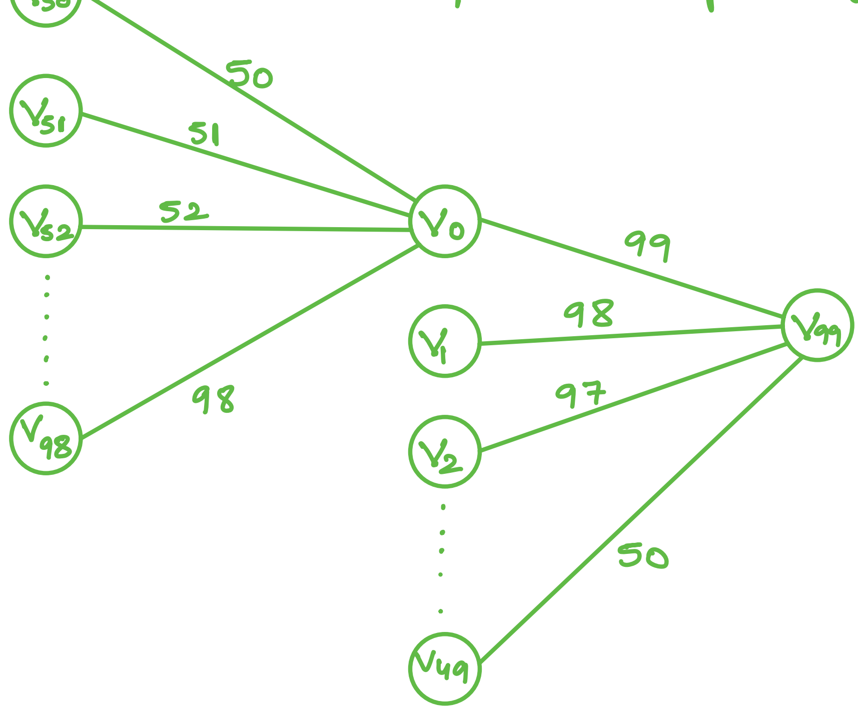
max profit = 44

Items chosen : 1, 2, 4

weight = 3 + 4 + 8 = 15

Ques 3b).

Maximum spanning tree



$$\text{Cost} = \underbrace{50 + 51 + 52 + \cdots + 98}_{} + \underbrace{99 + 98 + 97 + \cdots + 50}_{}$$

$$= 2\,(50 + 51 + 52 + \cdots + 98) + 99$$

$$= 7351$$

Ques 4:

a).

	B	A	B	C	A	A	B	_
A	5	5	4	3	3	2	1	0
A	5	5	4	3	3	2	1	0
B	4	4	4	3	2	2	1	0
C	3	3	3	3	2	2	1	0
D	2	2	2	2	2	2	1	0
A	2	2	2	2	2	2	1	0
B	1	1	1	1	1	1	1	0
_	0	0	0	0	0	0	0	0

length = 5

Elements = ABCAB

Ques 4b).

Logic: Divide the array in 2 parts and compare the maximum and minimum of the 2 parts to get the maximum and minimum of whole array

```
#include <iostream>

using namespace std;

struct Pair {
    int min;
    int max;
};

struct Pair fun(int arr[], int low, int high)
{
    struct Pair sp ;

    // If there is only one element
    if (low == high)
    {
        sp.max = arr[low];
        sp.min = arr[low];
        return sp;
    }

    // If there are two elements
    if (high == low + 1)
    {
        if (arr[low] > arr[high])
        {
            sp.max = arr[low];
            sp.min = arr[high];
        }
        else
        {
            sp.max = arr[high];
            sp.min = arr[low];
        }
        return sp;
    }

    // If there are more than 2 elements
    int mid = (low + high) / 2;
    struct Pair lp = fun(arr, low, mid);
    struct Pair rp = fun(arr, mid + 1, high);

    // Compare minimums of two parts
    if (lp.min < rp.min)
        sp.min = lp.min;
    else
        sp.min = rp.min;
```

```
    // Compare maximums of two parts
    if (lp.max > rp.max)
        sp.max = lp.max;
    else
        sp.max = rp.max;

    return sp;
}

int main()
{
    int arr[] = {100, 11, 35, 8, 55, 30};
    int n = sizeof(arr)/sizeof(int);

    struct Pair res = fun(arr, 0, n - 1);

    cout << "Minimum element is " << res.min << endl ;
    cout << "Maximum element is " << res.max << endl ;

    return 0;
}
```

$m = 2^n$

Recurrence Relation : $T(m) = 2T\left(\frac{m}{2}\right) + 1$

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

$T(2^n) = 2T(2^{n-1}) + 1$

Solving : $a = 2 \quad b = 2 \quad k = 0 \quad p = 0$

$a > b^k$

$2 > 2^0$

$T(m) = \Theta(m^{\log_b a}) = \Theta(m^{\log_2 2}) = \Theta(m) = \Theta(2^n)$

Ques 5a).

n = 10

relax every edge 9 times

<u>Edges</u>

A → C : -2
A → B : 4
C → D : 2
C → F : 1
S → A : 7
S → C : 6
S → F : 5
S → E : 6
E → F : -2
E → H : 3
B → G : -2
B → H : -4
H → G : 1
G → I : -1
I → H : 1
F → D : 3

<u>Cost</u>

		1st	2nd	3rd
S	→ 0			
A	→ $\cancel{\infty}$	7		
B	→ $\cancel{\infty}$		11	
C	→ $\cancel{\infty}$	$\cancel{6}$	5	
D	→ $\cancel{\infty}$	7		
E	→ $\cancel{\infty}$	6		
F	→ $\cancel{\infty}$	$\cancel{5}$ 4		
G	→ $\cancel{\infty}$	$\cancel{10}$	$\cancel{9}$ 8	
H	→ $\cancel{\infty}$	$\cancel{9}$	7	
I	→ $\cancel{\infty}$	$\cancel{9}$	7	

Relax 1st time

A → C: ∞, ∞ + (-2)

A → B: ∞, ∞ + 4 ✗

C → D: ∞, ∞ + 2 ✗

C → F: ∞, ∞ + 1 ✗

S → A: ∞, 0 + 7 ✓

S → C: ∞, 0 + 6 ✓

S → F: ∞, 0 + 5 ✓

S → E: ∞, 0 + 6 ✓

E → F: 5, 6 - 2 ✓

E → H: ∞, 6 + 3 ✓

B → G: ∞, ∞ - 2

B → H: 9, ∞ - 4

H → G: ∞, 9 + 1 ✓

G → I: ∞, 10 - 1 ✓

I → H: 9, 9 + 1

F → D: ∞, 4 + 3 ✓

Relax 2nd time

$A \to C$: 6, 7-2 ✓

$A \to B$: ∞, 7+4 ✓

$C \to D$: 7, 5+2 ×

$C \to F$: 4, 5+1 ×

$S \to A$: 7, 0+7 ×

$S \to C$: 5, 0+6 ×

$S \to F$: 4, 0+5 ×

$S \to E$: 6, 0+6 ×

$E \to F$: 4, 6-2 ×

$E \to H$: 9, 6+3 ×

$B \to G$: 10, 11-2 ✓

$B \to H$: 9, 11-4 ✓

$H \to G$: 9, 7+1 ✓

$G \to I$: 9, 8-1 ✓

$I \to H$: 7, 7+1 ×

$F \to D$: 7, 4+3 ×

Relax 3rd time

$A \to C$: 5, 7-2 ×

$A \to B$: 11, 7+4 ×

$C \to D$: 7, 5+2 ×

$C \to F$: 4, 5+1 ×

$S \to A$: 7, 0+7 ×

$S \to C$: 5, 0+6 ×

$S \to F$: 4, 0+5 ×

$S \to E$: 6, 0+6 ×

$E \to F$: 4, 6-2 ×

$E \to H$: 7, 6+3 ×

$B \to G$: 8, 11-2 ×

$B \to H$: 7, 11-4 ×

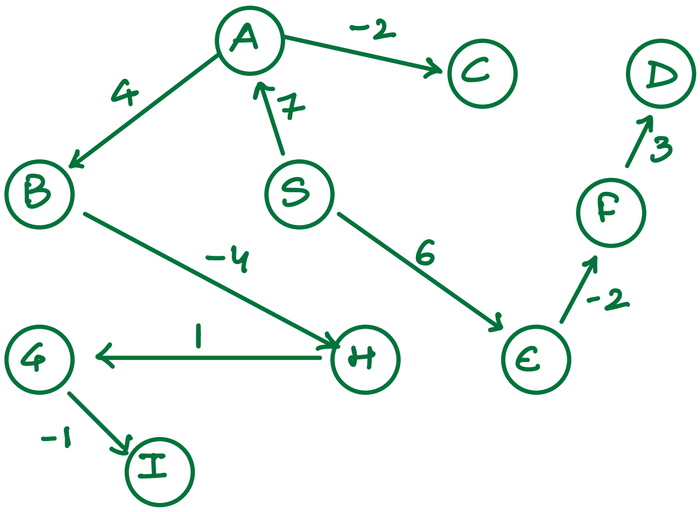
$H \to G$: 8, 7+1 ×

$G \to I$: 7, 8-1 ×

$I \to H$: 7, 7+1 ×

$F \to D$: 7, 4+3 ×

No changes after relaxing 3rd time. stop.



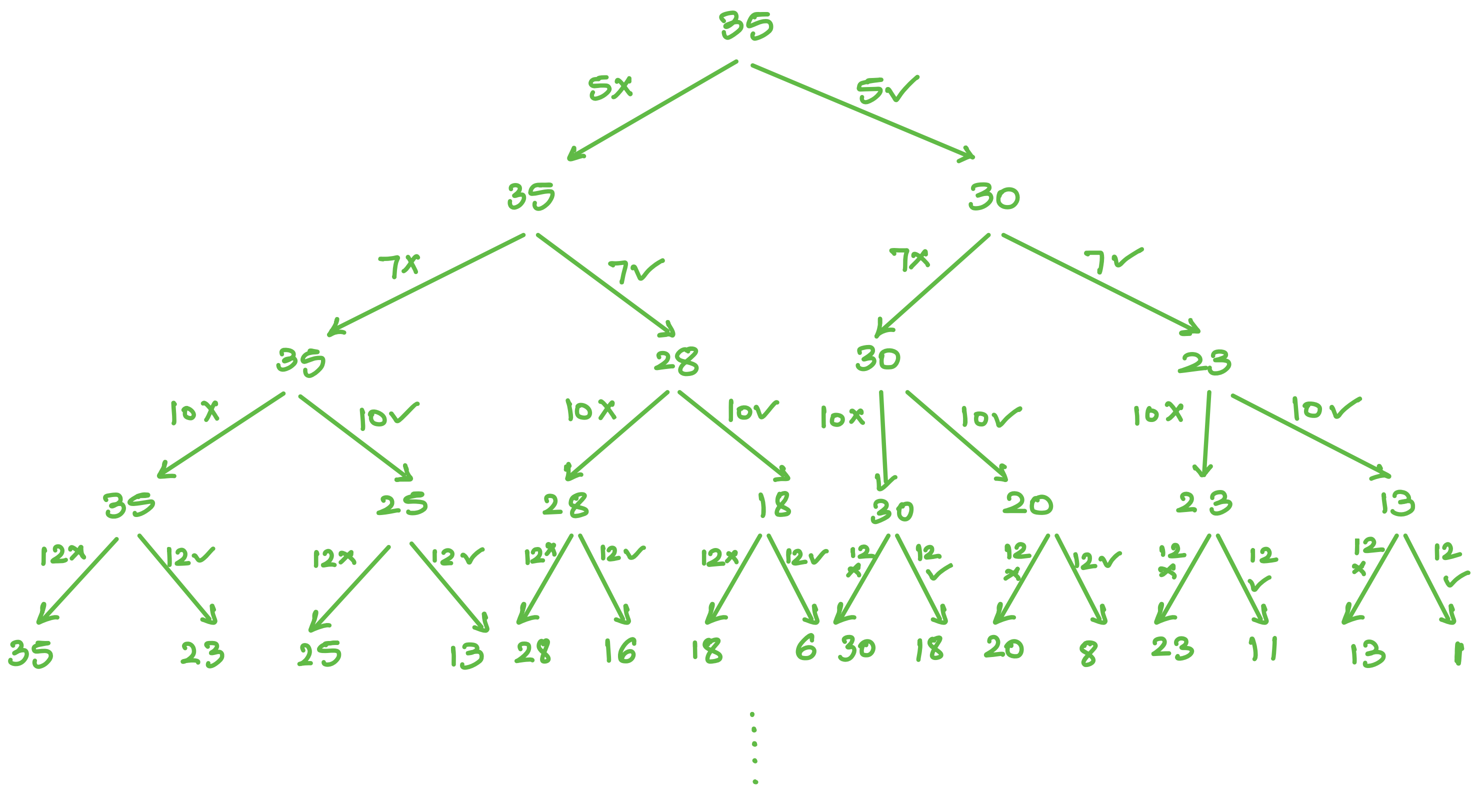
Ques 5b).

```
for (int i=1; i<=k-1; i++)
        heap.delete_max();
cout<< heap.get_max();
```

Time Complexity: $k \log n$

Ques 6a).

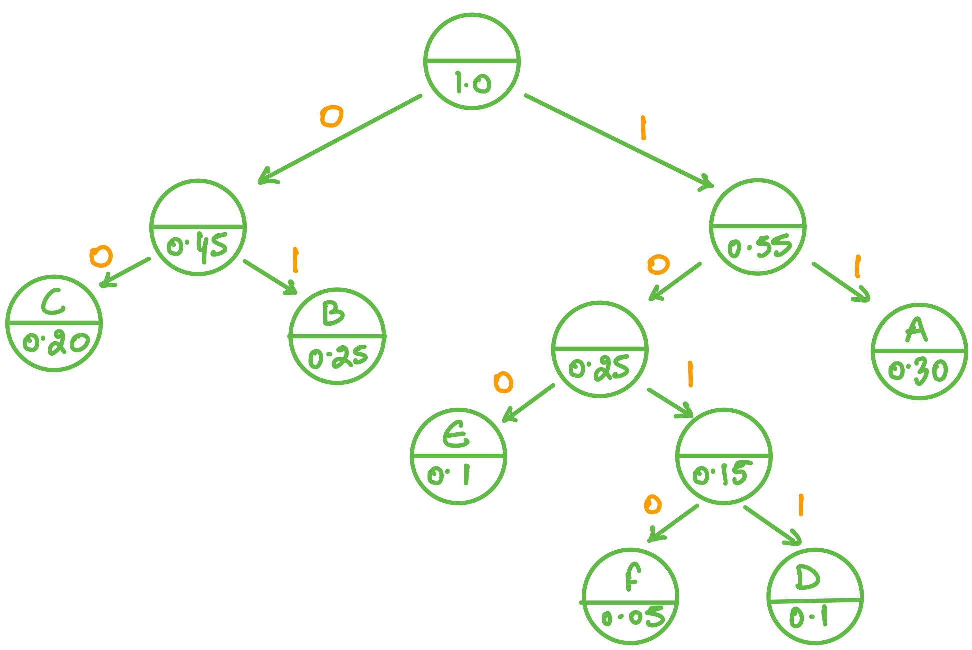
```
void subset(int *arr, int n, int idx, int target, string curr_set)
{
    if(target == 0)
    {
        cout << curr_set << endl ;
        return ;
    }

    if(idx == n || target < 0)
        return ;

    subset(arr, n, idx+1, target, curr_set) ; // exclude
    subset(arr, n, idx+1, target-arr[idx], curr_set + " " + to_string(arr[idx])) ; // include
}

int main()
{
    int arr[] = {5, 7, 10, 12, 15, 18, 20} ;
    int n = sizeof(arr) / sizeof(int) ;
    int m = 35 ;

    subset(arr, n, 0, m, "") ;

    return 0 ;
}
```



Subsets = 15 20
7 10 18
5 12 18
5 10 20

Ques 6b).

A: 0·30
B: 0·25
C: 0·20
D: 0·10
E: 0·10
F: 0·05



C: 00
B: 01
E: 100
F: 1010
D: 1011
A: 11