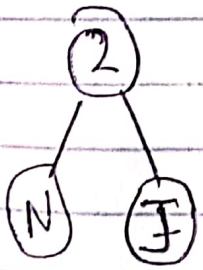
Question (1)

(a)(1) Muhammad Iftikhar Jan
MUHAMMAD IFTIKHAR JAN

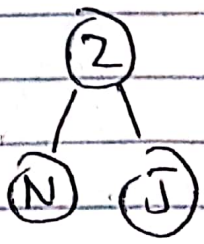
	Letter	Freequency
1	M	3
2	U	1
3	H	2
4	A	4
5	D	1
6	I	2
7	F	1
8	T	1
9	K	1
10	R	1
11	J	1
12	N	1

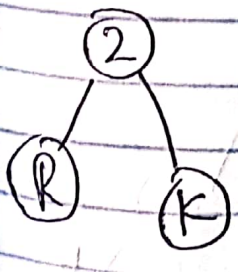
→ frequency table

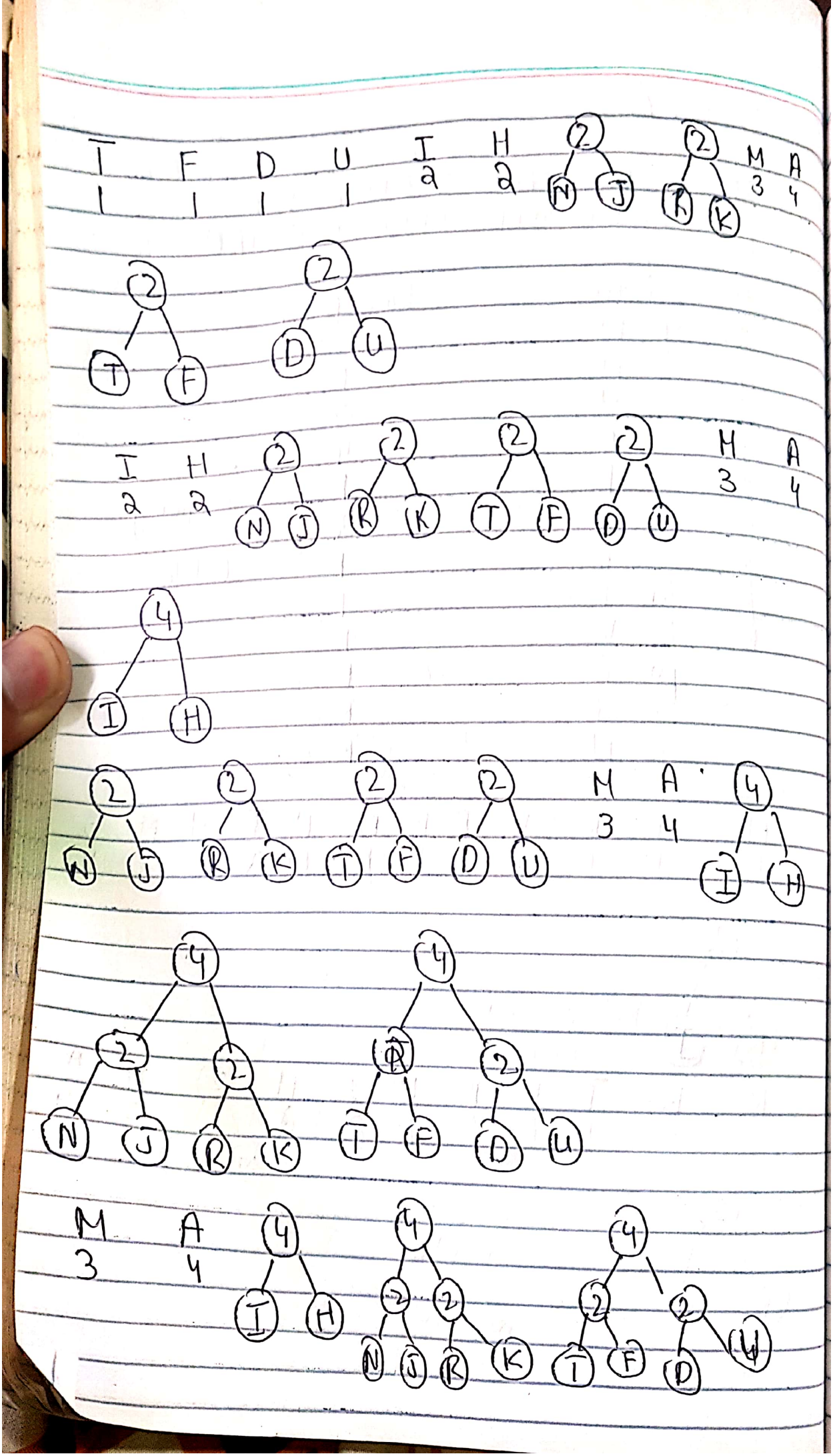
N J R K T F D U I H M A
1 1 1 1 1 1 1 1 2 2 3 4

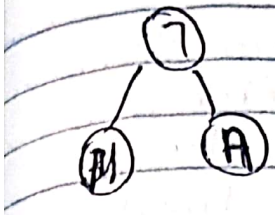


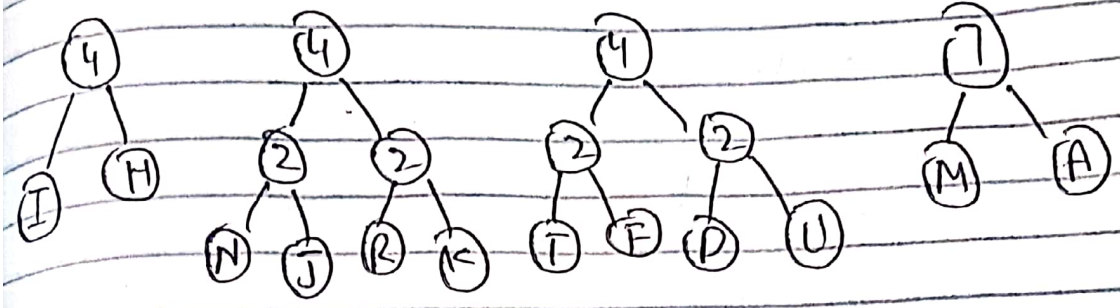
R K T F D U I H 2 M A
1 1 1 1 1 1 2 2 3 4

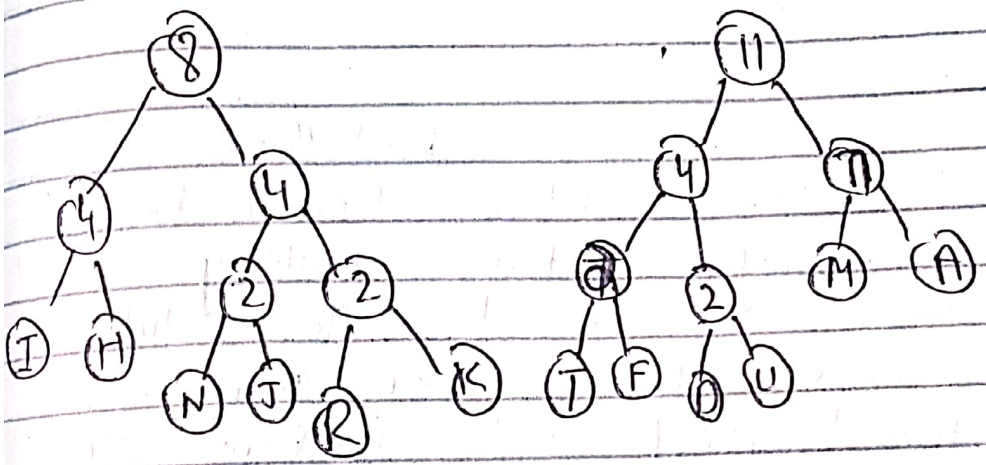


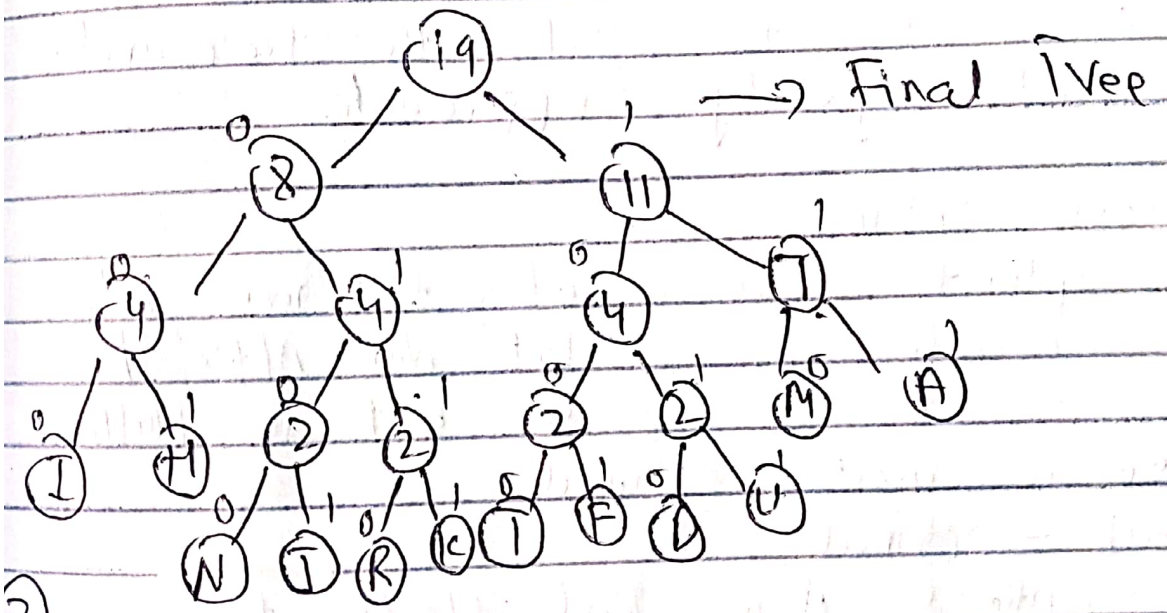










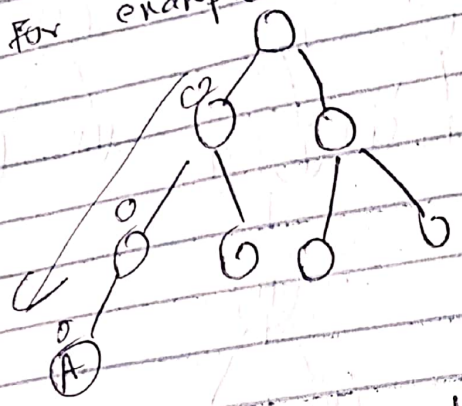


(2)

I = 000	H = 001	J = 0101
F = 1001	A = 111	N = 0100
T = 1000	R = 0110	
I = 0006	M = 110	
K = 0111	U = 1011	
	D = 1010	

(3) When we receive the stream of bits of codes like 000 then we have constructed tree then follow the path get the character/ value.

For example



(4) I used our solution Lower Frequency buid Heap? High priority before making Tree and I buid the Huffmen Code tree based on ~~priori~~ prioritized list and perform the traversal to get the tree and used greedy Approach.

(5) ?

(i) Most optimal

Most optimal because before Huffmen code every character is 8 bit Now After this reduce to 3 to 4 bit. Huffmen tree is most optimal

(ii) Sub - optimal

Sub - optimal it is Red - black tree because it take less time to the tree

(iii) Least - optimal Huffman

↳ because it reduce 8 bit to 3 or 4 bit.

Question # 2

Worst Case probabaly happen when the data is alreedy sorted for partition happend beginning or ending the running time is $O(n^2)$

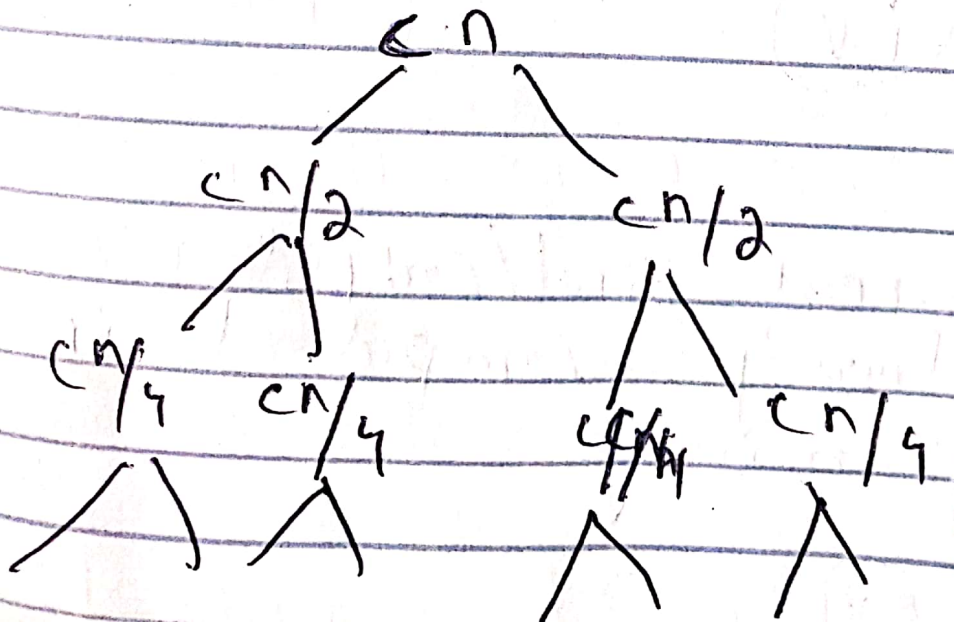
We avoid this using selet middle elemeet as pivot

Recurance Relation

~~$T(n) = T(n-2) + \ldots N-1$~~

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

Recursion Tree



Solve

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

$\Theta(n)$ = partition time

$T(1) = \Theta(1)$

(2)

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

$T(n) = T(n-2) + (n+1) \longrightarrow (A)$

$T(n-2) = T(n-4) + (n-1) \longrightarrow$ Put (A)

$T(n) = T(n-4) + (n-1) + n + 1$

$T(n-4) = T(n-6) + (n-3)$

$= T(n-6) + (n-3) + (n-1) + n + 1$

$T(n-6) = T(n-8) + (n-5)$

$T(n) = T(n-8) + (n-5) + (n-3) + (n-1) + n + 1$

$T(n-k) = T(n-8) + \sum_{i=0}^{k-3} (n-i) + 1$

$n - k = 1$

$n = k + 1$

$T(n-n) = T(1) + \sum_{i=0}^{n-1} (n-i) + 1$

$1 + \sum_{i=0}^{n-3} (n-i) + 1$ So total Running time is $\Theta(n^2)$

Question #03

for example

coin : {1, 5, 6, 4

w 2 10

j →

i ↓	0	1	2	3	4	5	6	7	8	9	10
1	0	1	2	3	4	5	1				
5	0	1	2	1	1	1					
6	0	1	2	3	4						
4	0	1	2	1	1						

Using Dynamic strategy

```
a[i][0]=0

for(i=0  i<=coins.lenght ; i++)
{
    for(j=0 ; j<=w ; j++)
    {
        if((coin[i]>j)
        {
            a[i][j]=a[i-1][j]
        }
        else
            a[i][j]=min(a[i-1][j],
                        1+a[i][j-coin[i]
```

$O(n^3)$ take this time

10
9
2
5

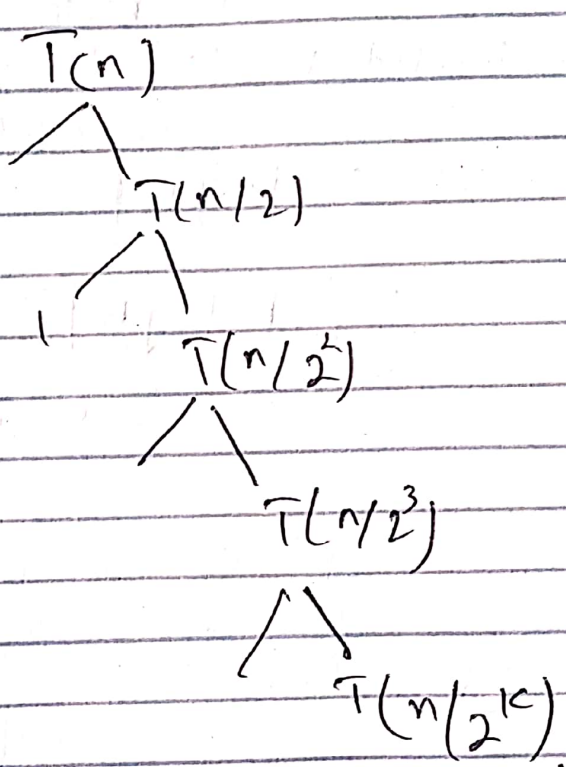
Question # 4

f(n) ——— T(n)

1— if (n mod 2 == 1)
return n;
— else
f(n/2) —— return f(n/2)

Recurrance Relation

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



When n/ value is odd then mod is 1

$\frac{n}{2^k} = 3$ (odd)

$n = 3 \cdot 2^k$

$\log n = 3k \log 2$ = O(log n)

$k = \log n$

```
G(n)
    for k ← 1 to n do:
        Print F(k)
```

```
F(n)
    if (n mod 2 == 1)
        return n
    else
        return f(n/2)
```

```
G(n)
    for k ← 1 to n do k+1
        Print f(k)
```

$f(n) \longrightarrow O(\log n)$

$g(n) \geq \Omega(n \log n)$

Question #5

```
a = 0
for( i=1 ; i <n ; i++)  ——— n+n

    for (j=1 ; j<i*i ; j++)   n + n+1
        for ( k; k<j ; k++)  n*n
            {
            cin >> q          n
            if(a%2==0) k
                break;
            }
```

Worst Case Analys

in the worst case This code will run

$O(n^3)$

and in the best case This code will run

$O(n^2)$

because break take you out of the third loop not other that why in the best case $O(n^2)$ if condition is true

Question # 6

We can Handle Np complet problems as Approximation instead of finding the optimal Solution. and we try to Searching of Solution of most of facto from optimal once. because we cannot solved the Np complet problem in polynomial time

Question #07

(ii)

False,
$\Theta(V^2)$ take memory not the time time depend upon How implomented either Adjancy list or matrix. If adjancy Matrix then $\Theta(1)$ will take time

(iii)

False
$\Theta(V+E)$ take time

(iv)

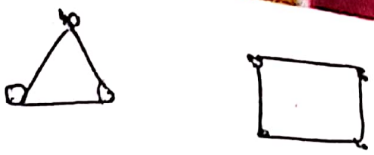
True:
because we cannot find/compute which is already computed in Dynamic programming.

(v)

False:
When the Array is already sorted then This worst case then partition will start end or beginng.

(vi)

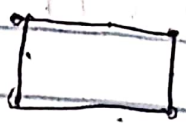
True:
If we adjancy list then it will be $O(V+E)$



(vii)

False

In connected graph we have n vertice and n edge



(ix)

Yes,

using Counting sort

(i)

false:

lenght is dect to root

(viii)

Yes

because in one Array we can computed median first 10 elemnt small and 10 largest there need first Sort the values.