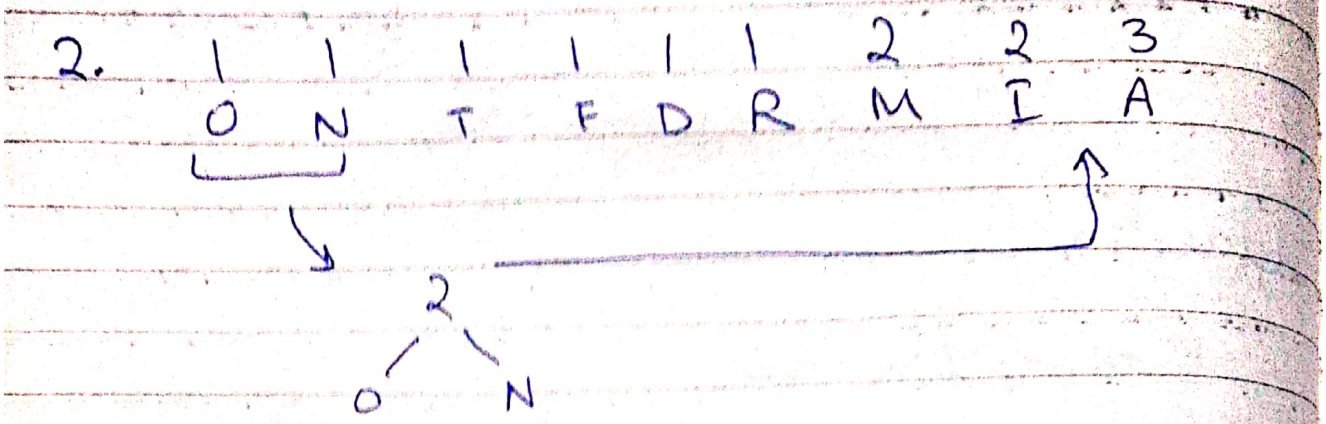
Momina Atif Dar
P18-0030
Algorithms – CS302
22nd June, 2020

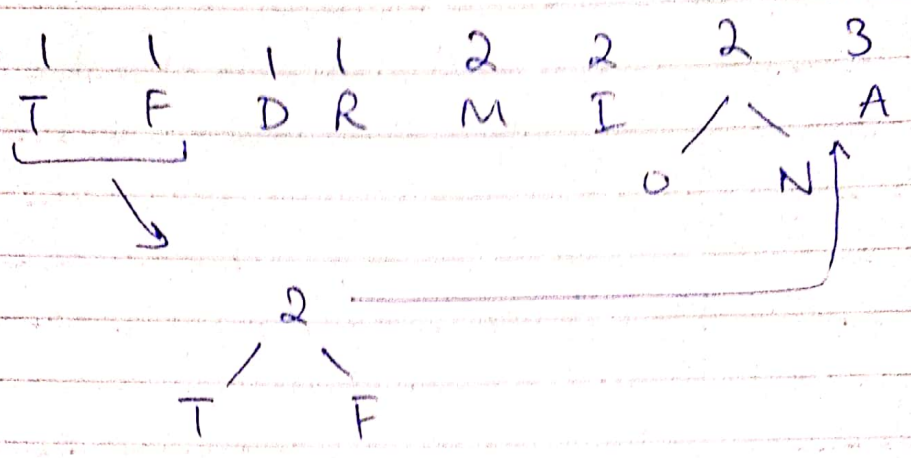
Question 1:

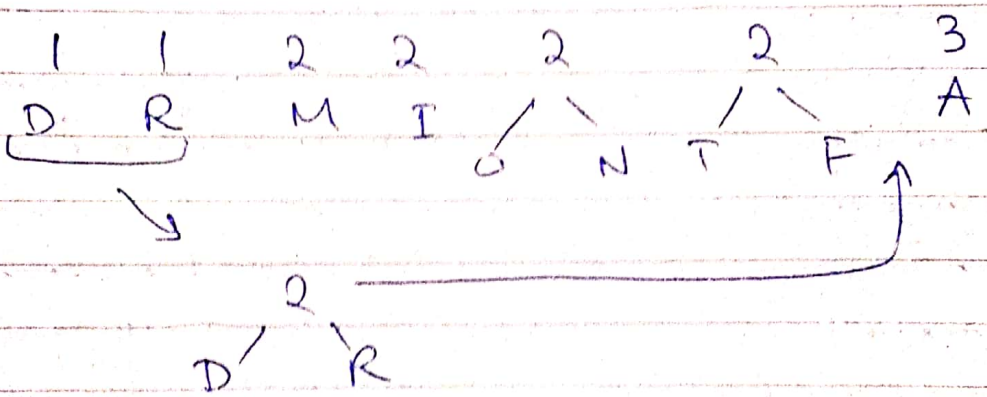
1.

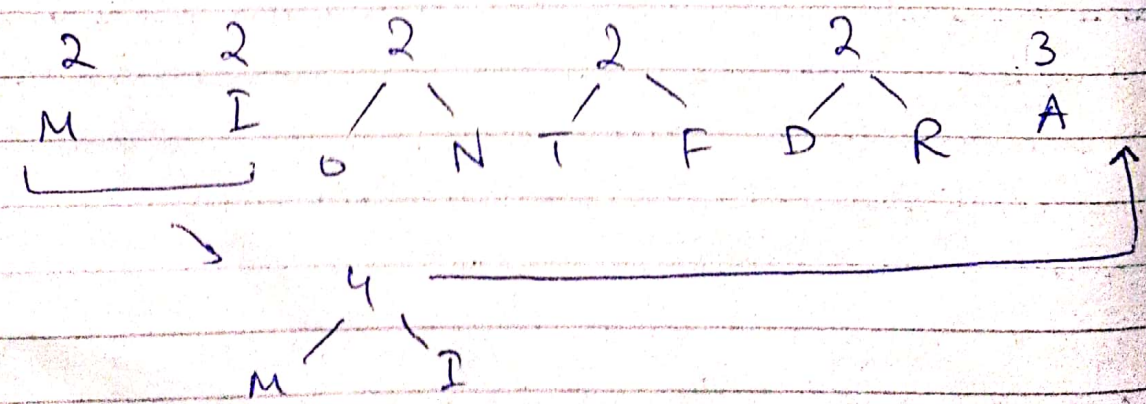
M	2
O	1
I	2
A	3
N	1
T	1
F	1
D	1
R	1

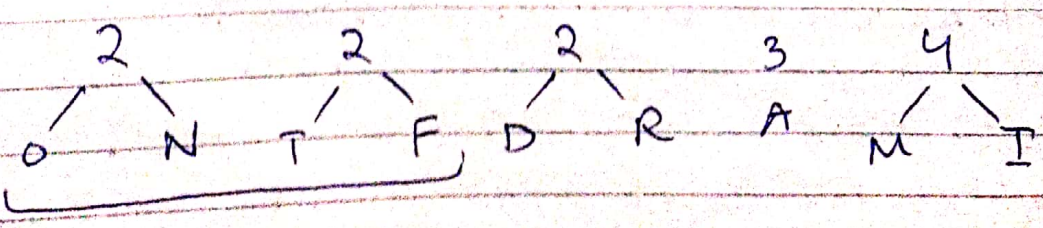
2.

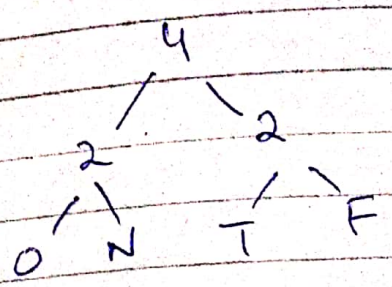


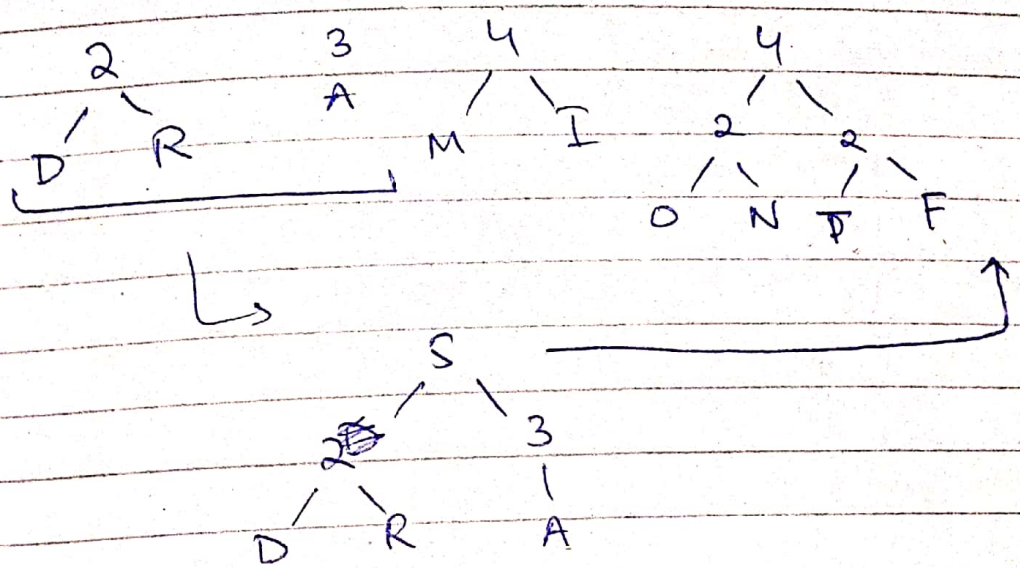


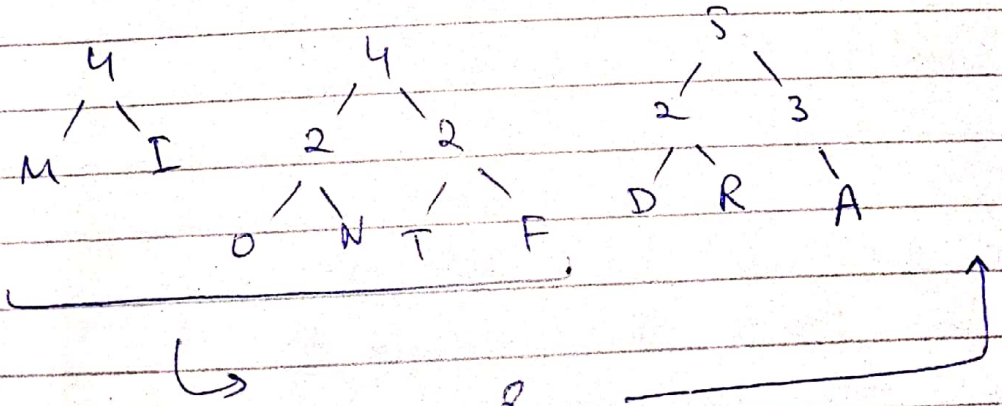


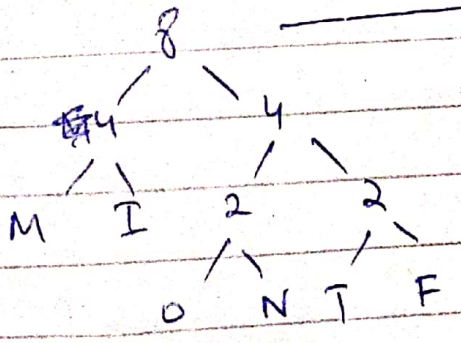


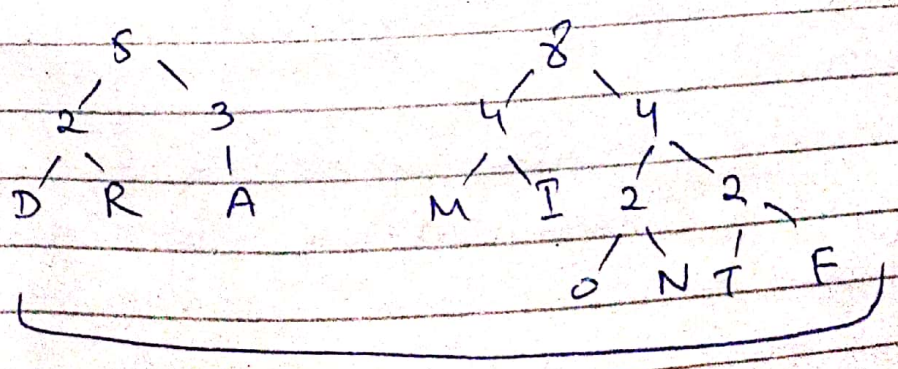


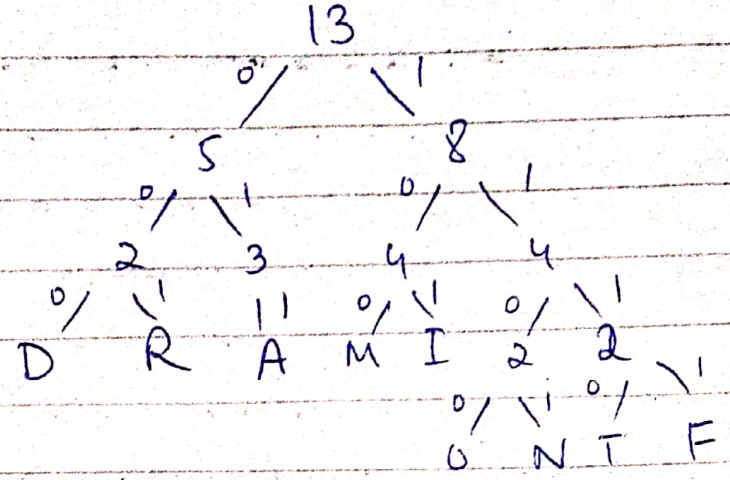












Character	Code
M	100
O	1100
I	101
N	1101
A	011
T	1110
F	1111
D	000
R	001

3. Reciever has got the tree as above so they can trace the codes. As all the codes are uniques and not prefix of any other code so decrypting becomes easy.

4. Greedy technique because
   i) Greedy choice property is applied
   ii) optimal substructure.
   We want shorter length of codes so that is the greed. We arrange mostly occured to close to root and least occured at leaf nodes.

As there are other solutions as well. For example a CBT like structure where every character is of same length no matter if it occurs the least or the most. And this is the optimal solution.

5. (i) The tree that I just made is the most optimal as it has short codes for most occuring characters and long codes for leeast occuring. This way we are freeing space of memory.

(ii) The tree which has short as well as long codes but doesn't save much space as that of most optimal structure.

(iii) CBT-like tree will be the least optimal as all the characters will have same length of code no matter what is their frequency of occurence. This takes up so much memory.

Q2 1, n-2

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

$\frac{n}{2}$ levels

n → cost : n

↓

n-2 → cost : n

↓

n-4

↓

⋮

1

$$T(n) = \frac{n}{2} \times n$$

$$T(n) = O(n^2) \quad \text{(worst case)}$$

~~because divide~~

Question #3

We will go for Greedy Algorithm because we need least number of coins exchange and this is greedy choice property. Because there are more than one solution to all the problems so 'optimal substructure' property also exists.

For example $n = 15000$

Possible sols: { 5000, 5000, 5000 }

{ 5000, 5000, 1000, 1000, 1000, 1000, 1000 }

but 1st one is optimal because least no. of exchanges.

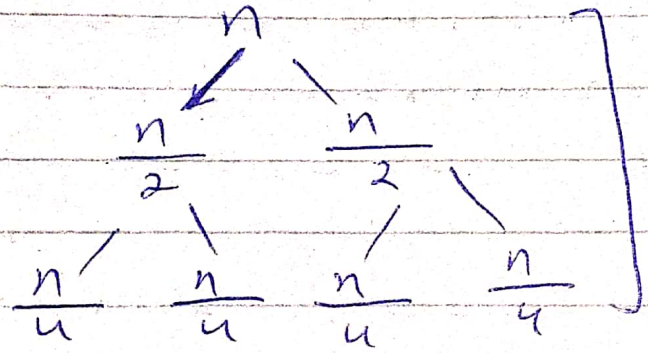
Time complexity: $O(n \log_2 n)$

Space complexity: $\Theta(1)$

↓

because no space is being taken.

Q4 $F(n)$ worst case running time



$\log_2 n$ levels

worst case running time $= O(\log_2 n)$

$G(n)$ for $k = 1$ to n → n - times

print($F(k)$); → running time of F

total running time of $G_1(n)$ =

$$n \times \log_2 n$$

$$= O(n \log_2 n)$$

Qs Worst case

$i = 1$	$i = 2$	$i = 3$
$j = (1)^2$	$j = 4$	$j = 9$
$k = j * j$	$k = 4 \times 4$	$k = 9 \times 9$
$= 1$	$= 16$	$= 81$

or $k = j * i^2$

$k = 1 * i^2$ $\quad k = 4 * i^2$ $\quad k = 9 * i^2$

So,

$$i^2 + 4i^2 + 9i^2 + \dots + 1$$

$$i^2 (1 + 4 + 9 + \dots)$$

$$i^2 \left(\frac{2n^3 + 3n^2 + n}{6} \right)$$

$$i^2 (n^3)$$

i is running n times so

$$n^2 (n^3) = O(n^5)$$

Best Case: (We assume on the first pass of k loop the loop breaks)

$i = 1$	$i = 2$	$i = n$
$j = 1$	$j = 4$	$j = n^2$ or i^2
$k = 1$	$k = 4$	$k = n^2$ or i^2

$$i^2 \rightarrow \frac{2n^3 + 3n^2 + n}{6} = \Omega(n^3)$$

Q6 There are different strategies to cope with NP-complete Problems.

1. Brute-Force
2. Approximation Algorithm
3. Heuristics
4. General Search Methods

In Brute Force we use our thinking abilities or take help of computers.

In Approximation strategy the algorithm runs pollynomially and produces close to optimal solution.

In Heuristics we cannot guaranty if a solution is good or not.

In General Search Method we use general methods to solve the problems.

Q1 (i) True Because in BFS a node is traversed only once. So there can be no other paths from s to v.

(ii) False. Searching takes O(1) time in adjacency matrix.

(iii) ~~False~~ True - We just have to go on every vertex and count no. of nodes attached to adjacency list. Counting is $\Theta(1)$ so total time would be $O(V)$.

(iv) False. Because even though DP is faster than recursion because it doesn't compute overlapping subproblems again and again. ~~But there is~~ But same happens in memoization as well. It saves the solution of overlapping subproblems like in DP. Thus DP is faster than recursion but same like memoization.

(v) True. As quick sort is the most efficient sorting algorithm so doesn't matter if array is sorted or not, it will ~~have~~ have $O(n \log n)$ complexity (if we assume pivot is not at end points).

(vi) True. Because BFS takes $\Theta(V+E)$ time to traverse all edges and vertices.

(vii) False. Any one vertex can join all other vertices as well so edges will be greater than vertices in that case.

(viii) True. First sort the array of 'n' numbers (I did it in merge-sort-median assignment) using for loops. in O(n) time. Then take first ten numbers and find median by formula $\frac{n+1}{2}$ and then take last ten numbers and find median.

(ix) False. Only counting sort could give us O(n). but that doesn't apply here because of negative element.
