PAGE NO.:
DATE / /

NAME:- AFNAN ATTAR PRN:- F1911 2003 CLASS: BE COMP2
SUBJECT:- DAA Experiment No.:- DAA (LP III) 01

Q1) What is Fibonacci series?

Ans 1. Fibonacci series is a series formed by fibonacci numbers denoted as Fn.
2. The numbers in the sequence are given as:
0, 1, 1, 2, 3, 5, 8, 13, 21, ...
3. In Fibonacci series every term is the sum of preceding two terms, starting from 0 and 1 as first and second terms.
4. Hence N^{th} term in fibbonacci series is given as

$$F_N = F_{N-1} + F_{N-2}$$

Q2) What is recursion?

Ans 1. The process in which a function calls itself directly or indirectly is called recursion and the corresponding function is called a recursive function.
2. A recursive function solves a particular problem by calling a copy of itself and solving smaller subproblem of original problems.
3. We must always provide a certain base in order to terminate the recursion process.

Q3) Analyse time and space complexity of fibonacci series?

Ans I) Recursive Approach:-
1. In this approach we go on finding fibonacci term until we hit 1 or 0 for which the base case is defined.

2.

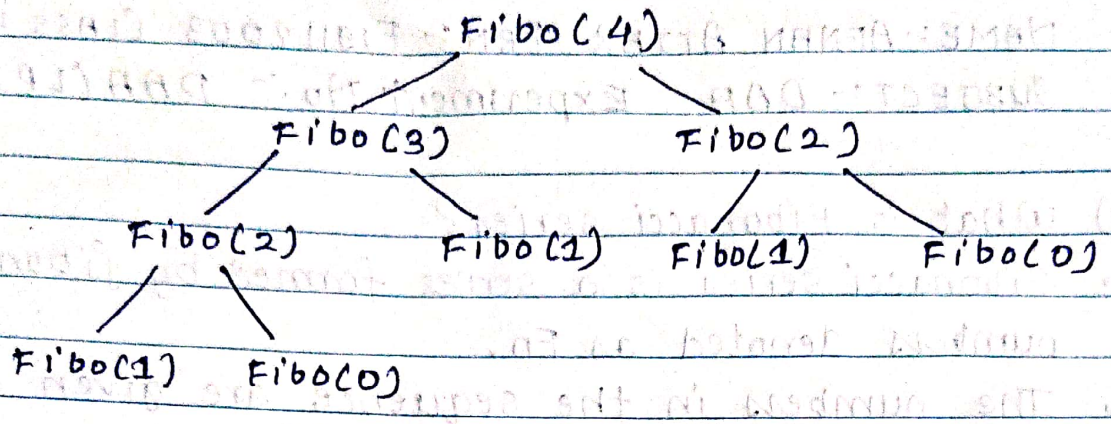


Fig: Recursive tree

As we can see we end up calculating redundant values hence the complexity of this approach:)

Time : $O(2^N)$ Space : $O(N)$

II. Dynamic Programming :-

1. By simply remembering or storing values of FN-1 and FN-2 we can add these two values and go on repeating this until we find the required term;
2. Time complexity : $O(N-2) \approx O(N)$

Space complexity : $O(1)$

PAGE NO.:
DATE / /

NAME: AFNAN ATTAR PRN:- F1911200D3 CLASS NO.:-
B.E COMP II SUBJECT:- DAA EXPERIMENT:- 02

Q1) Explain greedy approach?

Ans 1. A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment.

2. We do not worry if current best result will bring the overall best result.

3. If an optimal solution to the problem can be found by choosing best choice at each step without reconsidering the previous steps once choosen, then greedy approach can be applied.

Q2) Explain huffman coding with examples using greedy approach?

Ans 1. Huffman coding is a lossless data compression algorithm.

2. The idea is to assign variable length codes to input characters, lengths of the assigned codes are based on the frequency of corresponding characters.

3. The most frequent character gets the smallest code and the least frequent gets the largest code.

4. Example:-

i) Consider this string is to be send through a network:- B C A A D D D C C A C A C A C

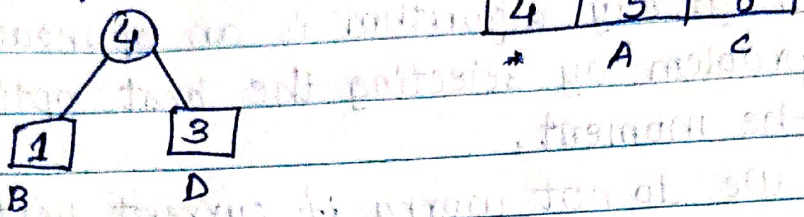
ii) We calculate frequency of each character:-

1	6	5	3
B	C	A	D

iii) Sort the frequency:

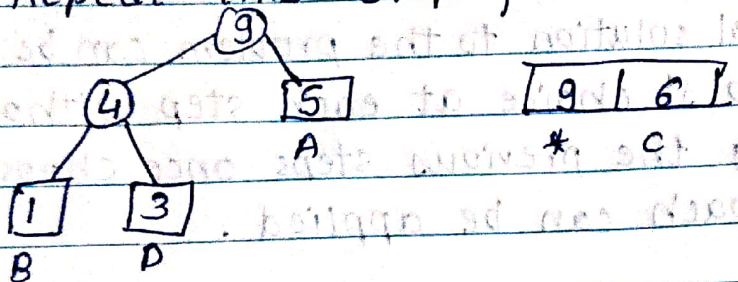
1	3	5	6
B	D	A	C

iv) Create an empty node and assign minimum frequency to the left of node and second minimum frequency to the right and set the value of node as the summation of both frequencies.

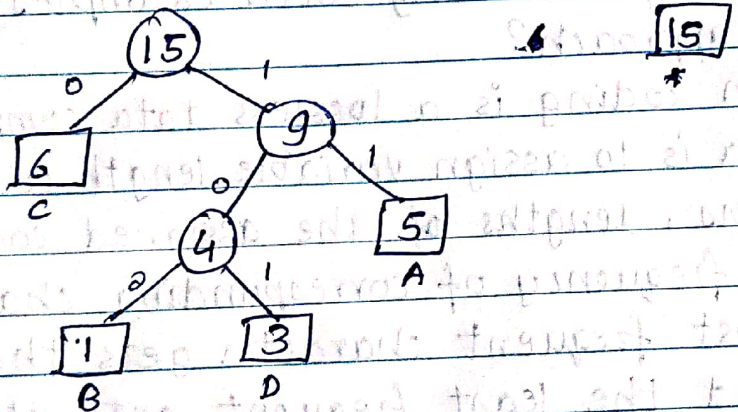


v) Repeat this step for remaining frequencies :-

1.



2.



We have successfully constructed Huffman tree.

Q3) Analyse time complexity of huffman coding?

Ans 1. The time complexity for encoding each unique character based on its frequency is O(nlogn).

2. Extracting minimum frequency from priority queue takes place 2*(n-1) times and its complexity is O(logn). Thus overall complexity is O(nlogn).

PAGE NO.:
DATE / /

NAME: AFNAN ATTAR PRN:- F19112003 CLASS:- BE COMPI
SUBJECT:- DAA EXPERIMENT NO.:- 03

Q1) Write realistic applications of this experiment in brief.

Ans 1. Knapsack problems appear in a wide variety of fields such as finding least wasteful way to cut raw materials, selection of investment and portfolios, selection of assets for asset-backed securitization, and generating keys for the Merkle Hellman and other knapsack cryptosystems.

2. One application is to display ads in between overs of a cricket match, where within a given time limit one has to show maximum number of ads.

3. An early application is in construction and scoring of tests in which the test-takers have a choice as to which questions they answer.

Q2) Explain knapsack with example using greedy apprach?

Ans

Item	1	2	3	4	5
Weight	5	10	15	22	25
Value	30	40	45	77	90

1. Consider the above table and let us assume we have a knapsack of 30kgs.

2. When we use a greedy approach we will simply pick the item having ~~height~~ highest value to weight ratio. Let us sort the table accordingly

Item	1	2	5	4	3
Weight	5	10	25	22	15
Value	30	40	90	77	45

PAGE NO:
DATE / /

8. Let us choose the first item,
Knapsack = 30 - 5 = 25 kgs
Value = 30

4. Second item,
Knapsack = 25 - 10 = 15 kgs
Value = 30 + 40 = 70

5. Now third item is of value 90 for 25 kgs
Since our knapsack is full we will take only 15 kgs of it.
∴ Price of 15 kgs = $\frac{90 \times 15}{25} = 54$

∴ Knapsack = 15 - 15 = 0
value = 70 + 54 = 124

6. We have solved our knapsack problem.

Q.3) Analyze time complexity of Knapsack.

Ans I) Dynamic Programming :-
Here the idea is to store the results of sub-problems so that they do not have to be recomputed when needed later.
Time - Complexity :- O(W * N)
N is no. of items and W is capacity of knapsack

II) Greedy Algorithm :-
Here we sort the item by their values and then fill our knapsack. This does not always result in a optimal solution.
Time - Complexity :- O(nlogn)

III) Branch and Bound Algorithm :-
Here we perform two main operations : branching that is dividing our problem and bounding i.e pruning off paths
Time - Complexity :- O(2n)