WELCOME

Presentation -1

TODAY'S TOPICS:

Dynamic Programming

- Dynamic Programming VS Divide-Conquer method
- Assembly-line scheduling
- Matrix-chain multiplication

TODAY'S TOPICS:

Greedy Method

- Dynamic method VS Greedy method
- An activity Selection Problem
- Elements of Greedy Strategy
- Huffman Codes

Members:

- Bithi
- Prema
- Jency

What is Dynamic Programming?

Dynamic programming is a technique of solving complex problem by breaking the problems into a collection of sub-problems, solving each sub-problem just once and then storing the solutions for future purposes so that we do not need to compute the result again.
The sub-problems are optimized to optimize the overall solution is known as optimal substructure property. The main use of dynamic programming is to solve optimization problems.

What is Optimization Problem?

Optimization problems mean that when we are trying to find out the minimum or the maximum solution of a problem.
The dynamic programming guarantees to find the optimal solution of a problem if the solution exists.

Example: Here we have Fibonacci series- 0,1,1,2,3,5,8,13,21,34,55,89,….. The numbers in the above series are not randomly calculated.
The Formula: F(n) = F(n-1) +F(n-2) with the base values F(0)=0 and F(1)=1. To calculate the other numbers, we follow above relationship. F(2) is sum of F(0) and F(1) which is equal to 1.

5 Steps of Dynamic programming:

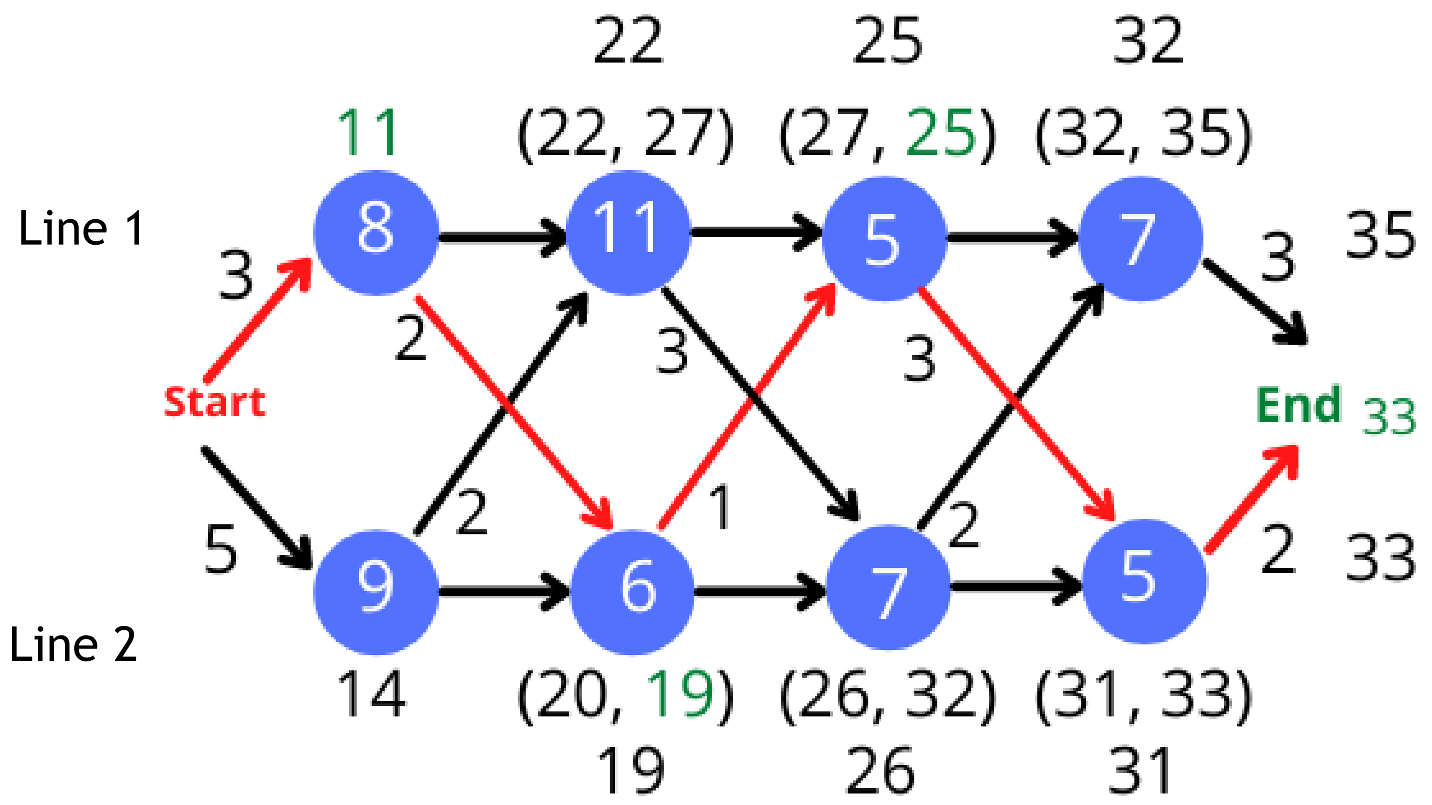
1. It breaks down the complex problem into similar sub-problems.
2. It finds the optimal Solution to these sub-problems.
3. It Stores the results of sub-problems. The storing process is called memorization.
4. It reuses the results so that no sub-problem is calculated more then once.
5. Finally calculate the result of complex Problem.

Features:	Dynamic Programming	Divide and Conquer
Recursive	Dynamic Programming Algorithm is non-recursive	Divide and Conquer is mostly recursive
Efficiency	More efficient	Less efficient than Divide and conquer
Data processing	Each processed state are saved to be reused for future need	All the data has to be computed even if it was processed
Time Consumption	Consumes less time in general	Consumes more time for execution
Solution type	Uses results of sub-problems to find optimal solutions of the main problem	Combines solutions of sub-problems to obtain the solution of main problem
Sub-problem dependency	Interdependent on each other	Independent of each other
Example	Dynamic program algorithm for Fibonacci series	Divide and conquer algorithm for Binary Search, merge sort

Assembly Line Scheduling:

We know that in automobile industry, automobiles are produced using assembly lines. There are multiple lines that are working together to produce products. Then a finished auto exits at the end of the line . The problem is that which line we should choose next from any station that will give best time utilization for company for one auto.

The main Goal of Assembly line scheduling is to give best route or can say the fastest from all assembly line.



	1	2	3	4	
F_{1j}	11	22	25	32	
F_{2j}	14	19	26	31	33

	2	3	4	
L_{1j}	1	2	1	
L_{2j}	1	1	2	2

Matrix Chain Multiplication:

Matrix Chain Multiplication is an optimization problem concerning the most efficient way to multiply a given sequence of matrices.

The problem is not actually to perform the multiplication, but merely to decide the sequence of the matrix multiplication involved.

Some opinion of matrix multiplication:

There are many options because matrix multiplication is associative. No matter how the product is parenthesized, the result obtained will remain the same.

For Example:
For Four matrices A, B, C and D there are five possible:

((AB)C)D = (A(BC))D = (AB)(CD)= A((BC)D) = A(B(CD))

And it doesn't affect the product.

The order in which the terms are parenthesized affects the number of Simple arithmetic operations needed to compute the product, that is, the computation complexity.

For example:
If A is 10X30 matrix, B is 30X5 matrix, and C is a 5X60 matrix then,
Computing (AB)C needs
=(AB) dimension + (AB)C dimension
=(10X30X5) + (10X5X60) =1500+3000 = 4500 operations.

While,
computing A(BC) needs
=(BC) dimension + A(BC) dimension
=(30X5X60) + (10X30X60) = 9000+18000 = 27000 operations.

Clearly the first method is more efficient.

Longest Common Subsequence

A subsequence of a given sequence is just the given sequence with some elements left out.

Longest means that the subsequence should be the biggest one.

Common means that some of the characters are common between the two strings.

Subsequence means that some of the characters are taken from the string that is written in increasing order to form a subsequence.

Example: Let, string X = abcd and Y = bd. The longest common Subsequence of these 2 strings is bd.

The longest subsequence is common to all the given sequence is referred to as longest common Subsequence.

The reason for using the LCS is to restrict the element of subsequence from occupying the consecutive position within the original subsequence.

Optimal Binary Search Tree

In binary search tree, the conditions are:
 left_subtree(key) < parent (key) <= right_subtree(key)

In order to search a key value it has search cost. **Optimal binary search tree** is a way to reduce the cost of search in a binary search tree.

We know the key values of each node in the tree, and we also know the frequencies of each node in terms of searching means how much time is required to search a node. The frequency and key value determine overall cost of searching node. The time required to search a node in BST is more than the balanced binary search tree as a balanced search tree contains a lesser number of levels than the BST.
There is one way that can reduce the cost of a binary search tree is known as an optimal binary search tree.

Let the nodes be: 10, 20, 30 for the BST

Find: 20

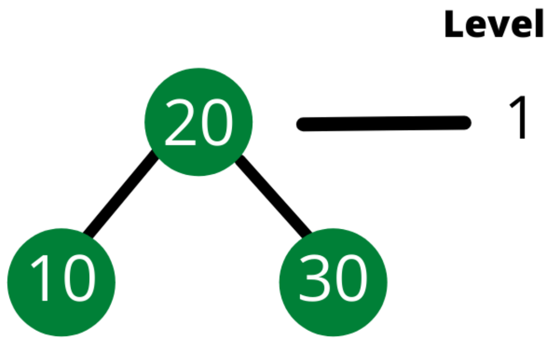


Figure:a

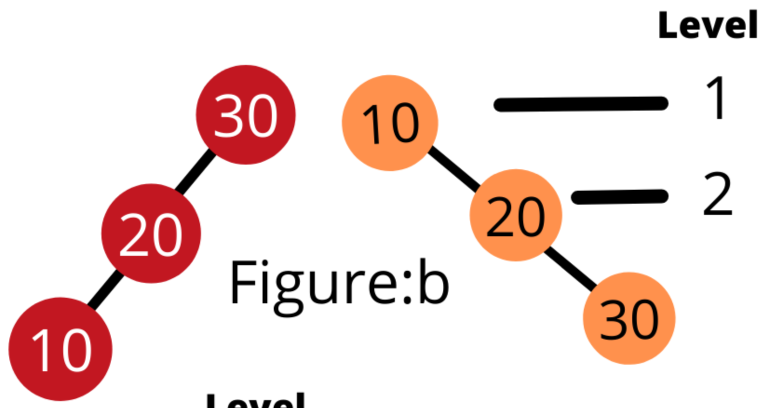


Figure:b

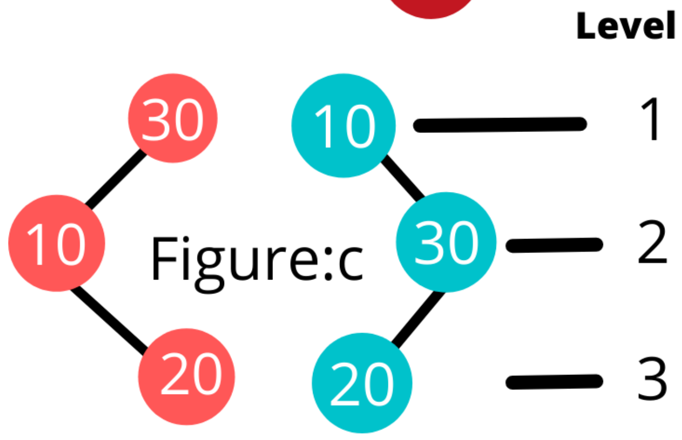


Figure:c

$$\frac{{}^{2n}C_n}{n+1} = \frac{{}^{2X3}C_3}{3+1}$$

$$= \frac{{}^{6}C_3}{4} = 5$$

Greedy Algorithm:
The greedy method is one of the strategies like Divide and conquer used to solve optimization problems. An optimization problem is a problem that demands either maximum or minimum results.

The Greedy method is the simplest and Straight forward approach.
It is not an algorithm but it is a technique. The main function of this Approach is that the decision is taken on the basic of the currently available information.

Characteristics of a greedy method:

i) To construct the solution in an optimal way. This algorithm creates two sets where one set contains all the chosen items and another set contains the rejected items.
ii) A greedy algorithm makes good local choices in the hope that the solution should be either possible solution or optimal one.

The components:
i) Candidate set: A solution that is created from the set is known as a candidate set.

ii) Selection function: This function is used to choose the candidate or subset which can be added in the solution.

Feature	Dynamic Programming	Greedy Method
1. Definition:	We choose at each step but the choice may depend on the solution to sub-problem.	We make whatever choice seems best at the moment in each step.
2. Efficiency	More efficient	Less efficient
3. Time consumption	Slower than Greedy method	Generally Faster then Dynamic programming
4. Approach	Bottom-up approach	Top- down approach
5. Decisions	Generates sequence of decisions for determining exact optimal solution.	Generated single decision sequence and hopes it might be the optimal solution. (Not garenteed)
6. Example:	0/1 Knapsack problem	Fractional knapsack

Application of Greedy Algorithm:

i) It is used in finding the shortest path.
ii) It is used to find the minimum spanning tree using the prim's algorithm or the Kruskal's algorithm.
iii) It is used in a job sequencing with a deadline.
iv) This algorithm is also used to solve the fractional knapsack problem.

An activity selection problem:
It is a mathematical optimization problem concerning the selection of non-conflicting activities to perform within a given time frame, given a set of activities each marked by a start time (s_i) and finish time (f_i)

Steps:
1. Sort the activities as per finishing time in ascending order.
2. Select the first activity.
3. Selection of the new activity if it's starting time is greater than or equal to previously selected activity's finish time.
4. Repeat step-3 till all activities are checked.

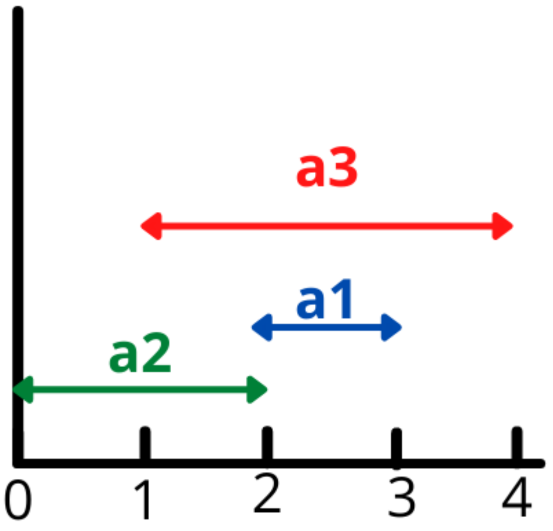
	a1	a2	a3
si	2	0	1
fi	3	2	4

Step-1: Sort by ascending order of finishing time

	a2	a1	a3
si	0	2	1
fi	2	3	4

Step-2: Selected

	a2	a1
si	0	2
fi	2	3



Huffman Coding:

It is a compression technique used for reducing the size of the data or message.

Why use Huffman Coding:

(i) Data can be encoded efficiently using Huffman Codes.
(ii) It is a widely used and beneficial technique for compressing data.
(iii) Huffman's greedy algorithm uses a table of the frequencies of occurrences of each character to build up an optimal way of representing each character as a binary string.

In order to store some data in a file if the data is very large in size then we can compress the file. To transfer the file we can send the compressed file in the network to **reduce the cost of transmission**.

Huffman Coding is of 2 types:
1. Fixed Size encoding
2. Variable Size encoding

Suppose we have a message of length 20 characters:

BCCABBDDAECCBBAEDDCC

When we normally want to send this message we sent it in it’s ASCII value we have to use 8 bits for each character.

So, A = $(65)_{10}$ = $(0100\ 0001)_2$

B = $(66)_{10}$ = $(0100\ 0010)_2$

And so on..

For total 20 we will have to spent 8X20 bits = 160 bits

Fixed Size Encoding:

Message: BCCABBDDAECCBBAEDDCC

Character	Count	Code
A	3	000
B	5	001
C	6	010
D	4	011
E	2	100
total	20	

As we use 1 bit for 2 types of data representation 0,1

2 bits for 4 types of data representation 00, 01, 10, 11

We have only 5 characters, So we can use 3 bits. $2^n = 2^3 = 8$ combinations. We will use only 5 combinations.

We will sent [message + encoded table]

Message has 20 characters each 3 bits now size of message=20 X3=60 bits
Encoded table has
[5 characters in 8 bits i.e. ASCII] + [3 bits code for those 5 characters]
5x8 + 3X5 =40 +15 = 55 bits

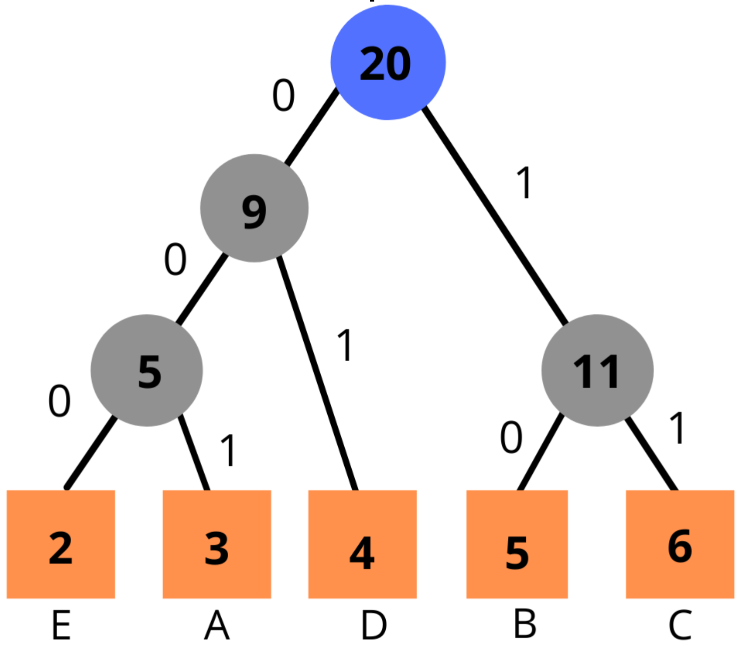
Total encoded file: message + table =60 +55 =115 bits

Variable Size Encoding:

Message: BCCABBDDAECCBBAEDDCC

Character	Count	Code	Count X bit no
A	3	001	3 X 3 = 9
B	5	10	5 X 2 = 10
C	6	11	6 X 2 =12
D	4	01	4 X 2 = 8
E	2	000	2 x 3 =6
total	20	12 bits	45 bits

Read from top to bottom



We will sent [message + encoded table]

Message has 20 characters each 3 bits now size of message= [total of count X bit] = 45 bits
Encoded table has
[5 characters in 8 bits i.e. ASCII] + [sum of bits of Variable sized codes] = 5x8 + 12 =40+12=52
Total encoded file: message + table =45+52 =97 bits

The End

Thanks for your Patience