Peoples Empowerment Group

Isb&M College of Engineering, Pune

NAAC Accredited with "B++" Grade

LABORATORY MANUAL

Laboratory Practice – III

Final Year of Computer Engineering
(2019 Course)

A.Y. 2023- 2024

Semester – I

DEPARTMENT OF COMPUTER ENGINEERING

Program Outcomes (PO's):

POs are statements that describe what students are expected to know and be able to do upon graduating from the program. These relate to the skills, knowledge, analytical ability attitude and behavior that students acquire through the program.

- **PO1: Engineering Knowledge:**
  Graduates will be able to apply the Knowledge of the mathematics, science and engineering fundamentals for the solution of engineering problems related to IT.
- **PO2: Problem Analysis:**
  Graduates will be able to carry out identification and formulation of the problem statement by requirement engineering and literature survey.
- **PO3: Design/Development of Solutions:**
  Graduates will be able to design a system, its components and/or processes to meet the required needs with consideration for public safety and social considerations.
- **PO4: Conduct Investigations of Complex Problems:**
  Graduates will be able to investigate the problems, categorize the problem according to their complexity using modern computational concepts and tools.
- **PO5: Modern Tool Usage:**
  Graduates will be able to use the techniques, skills, modern IT engineering tools necessary for engineering practice.
- **PO6: The Engineer and Society:**
  Graduates will be able to apply reasoning and knowledge to assess global and societal issues
- **PO7: Environment and Sustainability:**
  Graduates will be able to recognize the implications of engineering IT solution with respect to society and environment.
- **PO8: Ethics:**
  Graduates will be able to understand the professional and ethical responsibility.
- **PO9: Individual and Team Work:**
  Graduates will be able to function effectively as an individual member, team member or leader in multi -disciplinary teams.
- **PO10: Communication:**
  Graduates will be able to communicate effectively and make effective documentations and presentations.

- **PO11: Project Management and Finance:**
  Graduates will be able to apply and demonstrate engineering and management principles in project management as a member or leader.
- **PO12: Life-long Learning:**
  Graduates will be able to recognize the need for continuous learning and to engage in life-long learning.

Course Objectives and Course Outcomes (COs)

Course Objectives:

1. Learn effect of data preprocessing on the performance of machine learning algorithms
2. Develop in depth understanding for implementation of the regression models.
3. Implement and evaluate supervised and unsupervised machine learning algorithms.
4. Analyze performance of an algorithm.
5. Learn how to implement algorithms that follow algorithm design strategies namely divide and conquer, greedy, dynamic programming, backtracking, branch and bound.
6. Understand and explore the working of Blockchain technology and its applications.

Course Outcomes:

On completion of the course, students will be able to–

- **CO1:** Apply preprocessing techniques on datasets.
- **CO2:** Implement and evaluate linear regression and random forest regression models.
- **CO3:** Implement and evaluate linear regression and random forest regression models.
- **CO4:** Analyze performance of an algorithm
- **CO5:** Implement an algorithm that follows one of the following algorithm design strategies: divide and conquer, greedy, dynamic programming, backtracking, branch and bound
- **CO6:** Interpret the basic concepts in Blockchain technology and its applications.

Peoples Empowerment Group

ISB & M College of Engineering, Pune

NAAC Accredited with "B++" Grade

CERTIFICATE

This is to certify that Mr. /Ms. _____

of class BE Computer, Roll No._____Examination Seat No./PRN No. _____

has completed all the practical work in the Lab Practice-III [410246] satisfactorily, as prescribed by Savitribai Phule Pune University, Pune in the academic year 2023-24 (Term-I).

Place:

Date:

Course In-charge	**HOD**	**Principal**
Department of Comp. Engg.	**Department of Comp. Engg.**	**ISB&M COE, Pune**

INDEX

Sr. No	Title of Experiment	Date of performance	Date of Submission	Marks Obtained (10)	Signature of Faculty
1	Write a program non-recursive and recursive program to calculate Fibonacci numbers and analyze their time and space complexity.				
2	Write a program to implement Huffman Encoding using a greedy strategy.				
3	Write a program to solve a fractional Knapsack problem using a greedy method.				
4	Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy.				
5	Design n-Queens matrix having first Queen placed. Use backtracking to place remaining Queens to generate the final n-queen‘s matrix.				
6	Mini-Project				

Name & Signature of Course In-charge

Name of the Student:_____

Roll No: _____

CLASS: - B.E [Computer] **Course: - Lab Practice - III**

Experiment No. 1

**** Fibonacci series using recursive and non recursive method.****

Marks: /10

Date of / / 2023 **Sign With**

Assignment No. 1

Problem Statement:

Write a program non-recursive and recursive program to calculate Fibonacci numbers and analyze their time and space complexity.

Objective:

To study:

1. Students should be able to perform non-recursive and recursive programs to calculate Fibonacci numbers and analyze their time and space complexity.

Prerequisite:

1. Basic of Python or Java Programming
2. Concept of Recursive and Nonrecursive functions
3. Execution flow of calculate Fibonacci numbers
4. Basic of Time and Space complexity

Theory:

Introduction to Fibonacci numbers

- The Fibonacci series, named after Italian mathematician Leonardo Pisano Bogollo, later known as Fibonacci, is a series (sum) formed by Fibonacci numbers denoted as Fn. The numbers in Fibonacci sequence are given as: 0, 1, 1, 2, 3, 5, 8, 13, 21, 38, . ..
- In a Fibonacci series, every term is the sum of the preceding two terms, starting from 0 and 1 as first and second terms. In some old references, the term '0' might be omitted.

What is the Fibonacci Series?

- The Fibonacci series is the sequence of numbers (also called Fibonacci numbers), where every number is the sum of the preceding two numbers, such that the first two terms are '0' and '1'.
- In some older versions of the series, the term '0' might be omitted. A Fibonacci series can thus be given as, 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, . . . It can be thus be observed that every term can be calculated by adding the two terms before it.

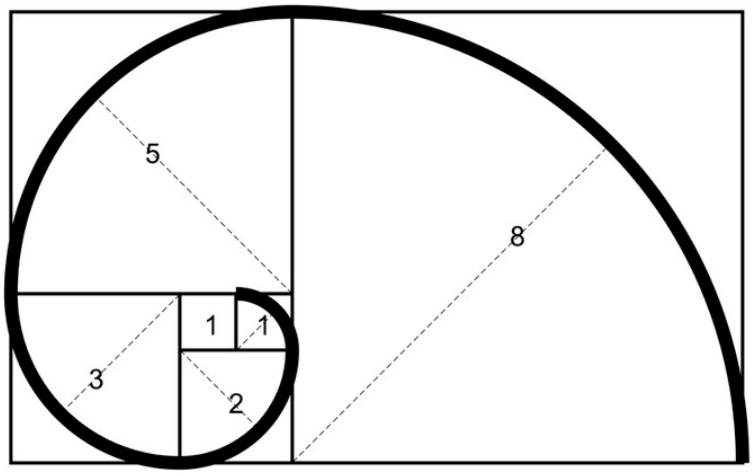
- Given the first term, F0 and second term, F1 as '0' and '1', the third term here can be given as, F2 = 0 + 1 = 1

Similarly,

F3 = 1 + 1 = 2

F4 = 2 + 1 = 3

Given a number n, print n-th Fibonacci Number.



Fibonacci sequence Formula

The Fibonacci sequence of numbers “Fn” is defined using the recursive relation with the seed values.

F0=0 and F1=1: Fn = Fn-1+Fn-2

Here, the sequence is defined using two different parts, such as kick-off and recursive relation. The kick-off part is F0=0 and F1=1.

The recursive relation part is Fn = Fn-1+Fn-2.

It is noted that the sequence starts with 0 rather than 1. So, F5 should be the 6th term of the sequence.

Examples:

Input : n = 2

Output: 1

Input: n = 9

Output : 34

The list of Fibonacci numbers are calculated as follows:

F_n	Fibonacci Number	F_n	Fibonacci Number
0	0	6	8
1	1	7	13
2	1	8	21
3	2	9	34
4	3	And so on…	And so on….
5	5		

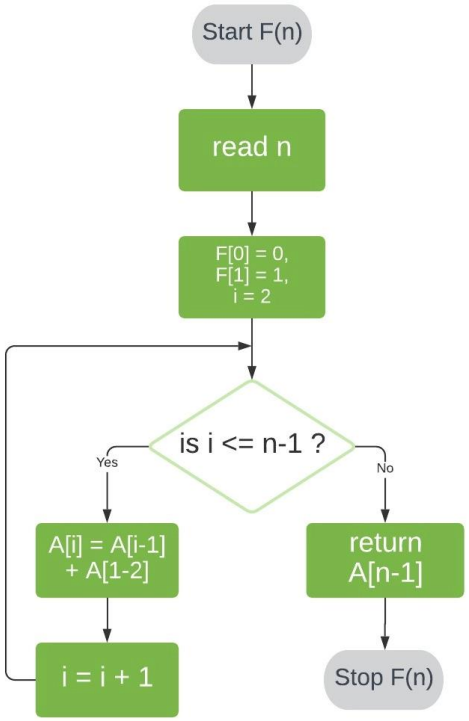
Method 1 (Use Non-recursion):

A simple method that is a direct recursive implementation of mathematical recurrence relation is given above.

First, we'll store 0 and 1 in F[0] and F[1], respectively.

Next, we'll iterate through array positions 2 to n-1. At each position i, we store the sum of the two preceding array values in F[i].

Finally, we return the value of F[n-1], giving us the number at position n in the sequence. Here's a visual representation of this process:



Time and Space Complexity of Space Optimized Method

- The time complexity of the Fibonacci series is **T(N) i.e, linear**. We have to find the sum of two terms and it is repeated n times depending on the value of n.
- The space complexity of the Fibonacci series using dynamic programming is **O(1)**.

Time Complexity and Space Complexity of Dynamic Programming

- The time complexity of the above code is **T(N) i.e, linear**. We have to find the sum of two terms and it is repeated n times depending on the value of n.

  The space complexity of the above code is **O(N)**.

Method 2 (Use Recursion):

Let's start by defining F(n) as the function that returns the value of Fn.

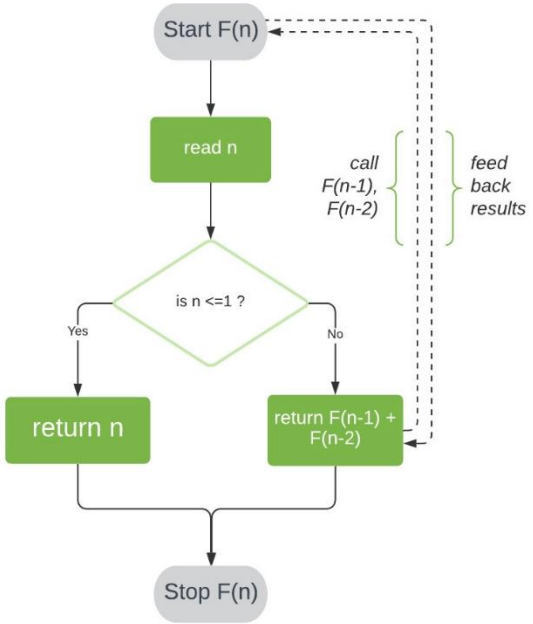
To evaluate *F*(*n*) for *n* > 1, we can reduce our problem into two smaller problems of the same kind: *F*(*n*-1) and *F*(*n*-2). We can further reduce *F*(*n*-1) and *F*(*n*-2) to *F*((*n*-1)-1) and *F*((*n*-1)-2); and *F*((*n*-2)-1) and *F*((*n*-2)-2), respectively.

If we repeat this reduction, we'll eventually reach our known base cases and, thereby, obtain a solution to *F*(*n*).

Employing this logic, our algorithm for *F*(*n*) will have two steps:

1. Check if $n \leq 1$. If so, return *n*.
2. Check if $n > 1$. If so, call our function *F* with inputs *n*-1 and *n*-2, and return the sum of the two results.

Here's a visual representation of this algorithm:



Time and Space Complexity

●The time complexity of the above code is T(2^N) i.e, exponential.

●The Space complexity of the above code is O(N) for a recursive series.

Applications of Fibonacci Series:

The Fibonacci series finds application in different fields in our day-to-day lives. The different patterns found in a varied number of fields from nature, to music, and to the human body follow the Fibonacci series. Some of the applications of the series are given as,

- It is used in the grouping of numbers and used to study different other special mathematical sequences.
- It finds application in Coding (computer algorithms, distributed systems, etc). For example, Fibonacci series are important in the computational run-time analysis of Euclid's algorithm, used for determining the GCF of two integers.
- It is applied in numerous fields of science like quantum mechanics, cryptography, etc.
- In finance market trading, Fibonacci retracement levels are widely used in technical analysis.

Conclusion: In this way we have explored Concept of Fibonacci series using recursive and non recursive method and also learn time and space complexity

Viva Questions:

1. What is the Fibonacci sequence of numbers?
2. How do the Fibonacci work?
3. What is the Golden Ratio?
4. What is the Fibonacci Search technique?
5. What is the real application for Fibonacci series?

Name of the Student:_____

Roll No: _____

CLASS: - B.E [Computer] **Course: - Lab Practice - III**

Assignment No. 2

**** Implement Huffman Encoding using greedy method ****

Marks: /10

Date of / / 2023 **Sign With**

Assignment No. 2

Problem Statement:

Write a program to implement Huffman Encoding using a greedy strategy.

Objectives:

Students should be able to understand and solve Huffman Encoding using greedy method

Prerequisite:

1. Basic of Python or Java Programming
2. Concept of Greedy method
3. Huffman Encoding concept

Theory:

What is a Greedy Method?

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result.

The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach. This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice to produce the global best result.

Advantages of Greedy Approach

- The algorithm is easier to describe.
- This algorithm can perform better than other algorithms (but, not in all cases).

Drawback of Greedy Approach

- As mentioned earlier, the greedy algorithm doesn't always produce the optimal solution. This is the major disadvantage of the algorithm
- For example, suppose we want to find the longest path in the graph below from root to leaf.

Greedy Algorithm

1. To begin with, the solution set (containing answers) is empty.
2. At each step, an item is added to the solution set until a solution is reached.
3. If the solution set is feasible, the current item is kept.
4. Else, the item is rejected and never considered again.

Huffman Encoding

- Huffman Coding is a technique of compressing data to reduce its size without losing any of the details. It was first developed by David Huffman.
- Huffman Coding is generally useful to compress the data in which there are frequently occurring characters.
- Huffman Coding is a famous Greedy Algorithm
- It is used for the lossless compression of data.
- It uses variable length encoding.
- It assigns variable length code to all the characters.
- The code length of a character depends on how frequently it occurs in the given text.
- The character which occurs most frequently gets the smallest code.
- The character which occurs least frequently gets the largest code.
- It is also known as Huffman Encoding.

Prefix Rule-

●Huffman Coding implements a rule known as a prefix rule.

●This is to prevent the ambiguities while decoding.

●It ensures that the code assigned to any character is not a prefix of the code assigned to any other character.

Major Steps in Huffman Coding-

There are two major steps in Huffman Coding-

1. Building a Huffman Tree from the input characters.
2. Assigning code to the characters by traversing the Huffman Tree.

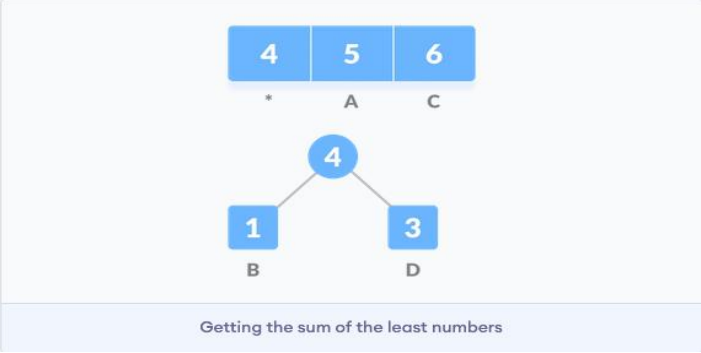
How does Huffman Coding work?

Suppose the string below is to be sent over a network.

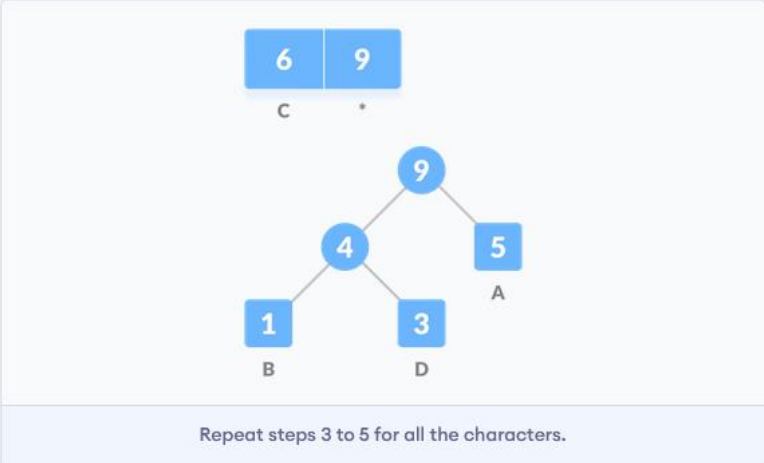
- Each character occupies 8 bits. There are a total of 15 characters in the above string. Thus, a total of 8 * 15 = 120 bits are required to send this string.
- Using the Huffman Coding technique, we can compress the string to a smaller size.
- Huffman coding first creates a tree using the frequencies of the character and then generates code for each character.
- Once the data is encoded, it has to be decoded. Decoding is done using the same tree.
- Huffman Coding prevents any ambiguity in the decoding process using the concept of prefix code ie. a code associated with a character should not be present in the prefix of any other code. The tree created above helps in maintaining the property.
- Huffman coding is done with the help of the following steps.
    1. Calculate the frequency of each character in the string.

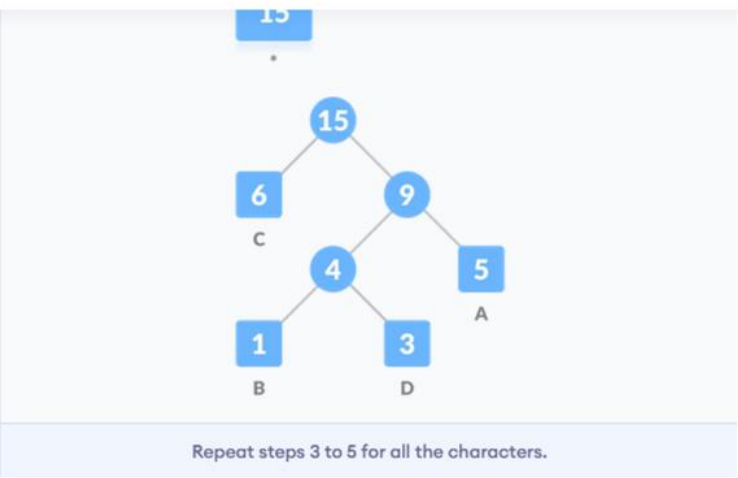
    2. Sort the characters in increasing order of the frequency. These are stored in a priority queue Q.

    3. Make each unique character as a leaf node.
    4. Create an empty node z. Assign the minimum frequency to the left child of z and assign the second minimum frequency to the right child of z. Set the value of the z as the sum of the above two minimum frequencies.

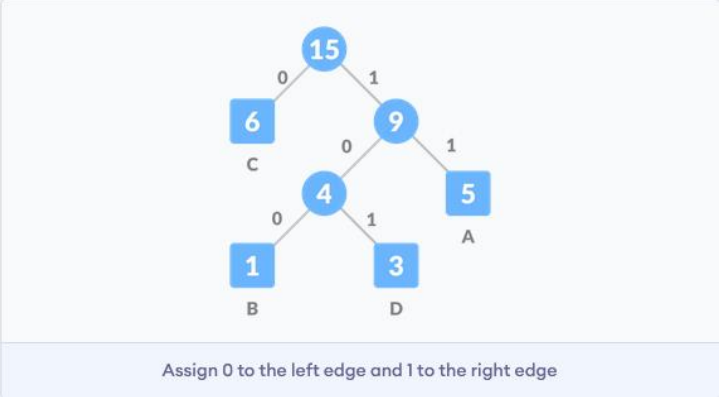


5. Remove these two minimum frequencies from Q and add the sum into the list of frequencies (*denote the internal nodes in the figure above).
6. Insert node z into the tree.
7. Repeat steps 3 to 5 for all the characters.





8. For each non-leaf node, assign 0 to the left edge and 1 to the right edge



For sending the above string over a network, we have to send the tree as well as the above compressed-code. The total size is given by the table below.

Character	Frequency	Code	Size
A	5	11	5*2 = 10
B	1	100	1*3 = 3
C	6	0	6*1 = 6
D	3	101	3*3 = 9
4 * 8 = 32 bits	15 bits		28 bits

Without encoding, the total size of the string was 120 bits. After encoding the size is reduced to- 32+ 15 + 28 = 75.

Time Complexity-

The time complexity analysis of Huffman Coding is as follows-

- extractMin( ) is called 2 x (n-1) times if there are n nodes.
- As extractMin( ) calls minHeapify( ), it takes O(logn) time.

Thus, Overall time complexity of Huffman Coding becomes **O(nlogn)**.

Conclusion: In this way we have explored Concept of Huffman Encoding using greedy method

A. Write short answer of following questions:

1. What is Huffman Encoding?
2. How many bits may be required for encoding the message ‘mississippi’?
3. Which tree is used in Huffman encoding?Give one Example
4. Why Huffman coding is lossless compression?

Name of the Student:_____

Roll No: _____

CLASS: - B.E [Computer] **Course: - Lab Practice - III**

Assignment No. 3

**** To Implement a Program Fractional Knapsack using greedy method ****

Marks: /10

Date of / /2023 **Sign With**

Assignment No. 3

Problem Statement:

Write a program to solve a fractional Knapsack problem using a greedy method.

Objectives:

Students should be able to understand and solve fractional Knapsack problems using a greedy method.

Prerequisite:

1. Basic of Python or Java Programming
2. Concept of Greedy method
3. Fractional Knapsack problem

Theory:

What is a Greedy Method?

A greedy algorithm is an approach for solving a problem by selecting the best option available at the moment. It doesn't worry whether the current best result will bring the overall optimal result. The algorithm never reverses the earlier decision even if the choice is wrong. It works in a top-down approach. This algorithm may not produce the best result for all the problems. It's because it always goes for the local best choice to produce the global best result.

Advantages of Greedy Approach

- The algorithm is easier to describe.
- This algorithm can perform better than other algorithms (but, not in all cases).

Drawback of Greedy Approach

- As mentioned earlier, the greedy algorithm doesn't always produce the optimal solution. This is the major disadvantage of the algorithm.
- For example, suppose we want to find the longest path in the graph below from root to leaf.

Greedy Algorithm

1. To begin with, the solution set (containing answers) is empty.
2. At each step, an item is added to the solution set until a solution is reached.
3. If the solution set is feasible, the current item is kept.
4. Else, the item is rejected and never considered again.

Knapsack Problem

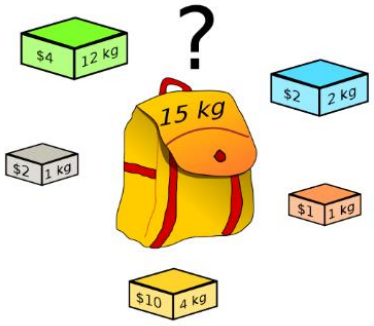
You are given the following-

- A knapsack (kind of shoulder bag) with limited weight capacity.
- Few items each having some weight and value.

The problem states-

Which items should be placed into the knapsack such that-

- The value or profit obtained by putting the items into the knapsack is maximum. and the weight limit of the knapsack does not exceed.



Knapsack Problem

Knapsack Problem Variants

Knapsack problem has the following two variants-

1. Fractional Knapsack Problem
2. 0/1 Knapsack Problem

Fractional Knapsack Problem-

In Fractional Knapsack Problem,

- As the name suggests, items are divisible here.
- We can even put the fraction of any item into the knapsack if taking the complete item is not possible.

Fractional Knapsack Problem using greedy method:

Step 1: For each item, compute its value /weight ratio.

Step 2: Arrange all the item in decreasing order for their value /weight ratio.

Step 3: start putting the item into the knapsack beginning from item with the highest ratio.

Time Complexity-

●The main time taking step is the sorting of all items in decreasing order of their value / weight ratio.

●If the items are already arranged in the required order, then while loop takes O(n) time.

●The average time complexity of Quick Sort is O(nlogn).

●Therefore, total time taken including the sort is O(nlogn).

Conclusion: In this way we have explored Concept of Fractional Knapsack using greedy method

Viva Questions:

1. What is Greedy Approach?
2. Explain concept of fractional knapsack
3. Difference between Fractional and 0/1 Knapsack
4. Solve one example based on Fractional knapsack(Other than Manual).

Name of the Student:_____

Roll No: _____

Class : - B.E [Computer] **Course: - Lab Practice - III**

Assignment No. 4

**** Implement 0/1 Knapsack problem using dynamic programming ****

Marks: /10

Date of / / 2023 **Sign With**

Assignment No. 4

Problem Statement:

Write a program to solve a 0-1 Knapsack problem using dynamic programming or branch and bound strategy.

Objectives:

Students should be able to understand and solve 0-1 Knapsack problem using dynamic programming

Prerequisite:

1. Basic of Python or Java Programming
2. Concept of Dynamic Programming
3. 0/1 Knapsack problem

Theory:

What is Dynamic Programming?

- Dynamic Programming is also used in optimization problems. Like divide-and-conquer method, Dynamic Programming solves problems by combining the solutions of subproblems.
- Dynamic Programming algorithm solves each sub-problem just once and then saves its answer in a table, thereby avoiding the work of re-computing the answer every time.
- Two main properties of a problem suggest that the given problem can be solved using Dynamic Programming. These properties are overlapping sub-problems and optimal substructure.
- Dynamic Programming also combines solutions to sub-problems. It is mainly used where the solution of one sub-problem is needed repeatedly. The computed solutions are stored in a table, so that these don't have to be re-computed. Hence, this technique is needed where overlapping sub-problem exists.
- For example, Binary Search does not have overlapping sub-problem. Whereas recursive program of Fibonacci numbers have many overlapping sub-problems.

Steps of Dynamic Programming Approach

Dynamic Programming algorithm is designed using the following four steps –

- Characterize the structure of an optimal solution.
- Recursively define the value of an optimal solution.
- Compute the value of an optimal solution, typically in a bottom-up fashion.
- Construct an optimal solution from the computed information.

Applications of Dynamic Programming Approach

- Matrix Chain Multiplication
- Longest Common Subsequence
- Travelling Salesman Problem

Knapsack Problem

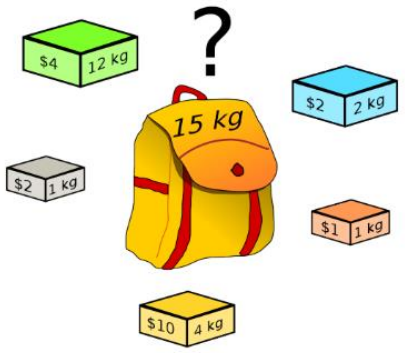
You are given the following-

- A knapsack (kind of shoulder bag) with limited weight capacity.
- Few items each having some weight and value.

The problem states-

Which items should be placed into the knapsack such that-

- The value or profit obtained by putting the items into the knapsack is maximum.
- And the weight limit of the knapsack does not exceed.



Knapsack Problem

0/1 knapsack problem is solved using dynamic programming in the following steps-

Step-01:

- Draw a table say 'T' with (n+1) number of rows and (w+1) number of columns.
- Fill all the boxes of 0^{th} row and 0^{th} column with zeroes as shown-

	0	1	2	3	.....	W
0	0	0	0	0	.....	0
1	0					
2	0					
	.....					
n	0					

T-Table

Step-02:

Start filling the table row wise top to bottom from left to right. Use the following formula-
$T(i , j) = \max \{ T(i-1 , j) , value_i + T(i-1 , j – weight_i) \}$

Here, T(i , j) = maximum value of the selected items if we can take items 1 to i and have weight restrictions of j.

- This step leads to completely filling the table.
- Then, value of the last box represents the maximum possible value that can be put into the knapsack.

Step-03:

- To identify the items that must be put into the knapsack to obtain that maximum profit,
- Consider the last column of the table.
- Start scanning the entries from bottom to top.
- On encountering an entry whose value is not same as the value stored in the entry immediately above it, mark the row label of that entry.
- After all the entries are scanned, the marked labels represent the items that must be put into the knapsack

Problem-.
For the given set of items and knapsack capacity = 5 kg, find the optimal solution for the 0/1 knapsack problem making use of a dynamic programming approach.

Item	Weight	Value
1	2	3
2	3	4
3	4	5
4	5	6

n = 4

w = 5 kg

(w1, w2, w3, w4) = (2, 3, 4, 5)

(b1, b2, b3, b4) = (3, 4, 5, 6)

Solution-

Given

- Knapsack capacity (w) = 5 kg
- Number of items (n) = 4

	0	1	2	3	4	5
0	0	0	0	0	0	0
1	0	0	3	3	3	3
2	0	0	3	4	4	7
3	0	0	3	4	5	7
4	0	0	3	4	5	7

T-Table

- The last entry represents the maximum possible value that can be put into the knapsack.
- So, maximum possible value that can be put into the knapsack = 7.

Time Complexity-

- Each entry of the table requires constant time θ(1) for its computation.
- It takes θ(nw) time to fill (n+1)(w+1) table entries.
- It takes θ(n) time for tracing the solution since tracing process traces the n rows.
- Thus, overall θ(nw) time is taken to solve 0/1 knapsack problem using dynamic programming

Conclusion: In this way we have explored Concept of 0/1 Knapsack using Dynamic approach

Viva Questions:

1. What is Dynamic Approach?
2. Explain concept of 0/1 knapsack?
3. Difference between Dynamic and Branch and Bound Approach.Which is best?
4. Solve one example based on 0/1 knapsack(Other than Manual)

Name of the Student:_____

Roll No: _____

Class : - B.E [Computer] **Course: - Lab Practice - III**

Assignment No. 5

**** N-Queen Problem using Backtracking ****

Marks: /10

Date of | | **Sign With** | |

Assignment No. 5

Problem Statement:

Design n-Queens matrix having first Queen placed. Use backtracking to place remaining Queens to generate the final n-queen's matrix.

Objectives:

- Students should be able to understand and solve n-Queen Problem.
- Understand basics concept of Backtracking.

Prerequisite:

1. Basic of Python or Java Programming
2. Concept of backtracking method
3. N-Queen Problem

THEORY:

Introduction to Backtracking

- Many problems are difficult to solve algorithmically. Backtracking makes it possible to solve at least some large instances of difficult combinatorial problems.

Suppose we have to make a series of decisions among various choices, where

- We don't have enough information to know what to choose
- Each decision leads to a new set of choices.
- Some sequence of choices (more than one choices) may be a solution to your problem.

What is backtracking?

Backtracking is finding the solution of a problem whereby the solution depends on the previous steps taken. For example, in a maze problem, the solution depends on all the steps you take one-by-one. If any of those steps is wrong, then it will not lead us to the solution. In a maze problem, we first choose a path and continue moving along it. But once we understand that the particular path is incorrect, then we just come back and change it.

In backtracking, we first take a step and then we see if this step taken is correct or not i.e., whether it will give a correct answer or not. And if it doesn't, then we just come back and change our first step. In general, this is accomplished by recursion. Thus, in backtracking, we first start with a partial sub-solution of the problem (which may or may not lead us to the solution) and then check if we can proceed further with this sub-solution or not. If not, then we just come back and change it.

Thus, the general steps of backtracking are:

- start with a sub-solution
- check if this sub-solution will lead to the solution or not
- If not, then come back and change the sub-solution and continue again

Applications of Backtracking:

- N Queens Problem
- Sum of subsets problem
- Graph coloring
- Hamiltonian cycles.

N queens on NxN chessboard

One of the most common examples of the backtracking is to arrange N queens on an NxN chessboard such that no queen can strike down any other queen. A queen can attack horizontally, vertically, or diagonally. The solution to this problem is also attempted in a similar way. We first place the first queen anywhere arbitrarily and then place the next queen in any of the safe places. We continue this process until the number of unplaced queens becomes zero (a solution is found) or no safe place is left. If no safe place is left, then we change the position of the previously placed queen.

N-Queens Problem:

A classic combinational problem is to place n queens on a n*n chess board so that no two attack, i.,e no two queens are on the same row, column or diagonal.

What is the N Queen Problem?

N Queen problem is the classical Example of backtracking. N-Queen problem is defined as, “given N x N chess board, arrange N queens in such a way that no two queens attack each other by being in the same row, column or diagonal”.

- For N = 1, this is a trivial case. For N = 2 and N = 3, a solution is not possible. So we start with N = 4 and we will generalize it for N queens.

If we take n=4then the problem is called the 4 queens problem.

If we take n=8 then the problem is called the 8 queens problem.

Algorithm

1) Start in the leftmost column

2) If all queens are place return true

3) Try all rows in the current column.

Do following for every tried row.

a) If the queen can be placed safely in this row then mark this [row, column] as part of the solution and recursively check if placing queen here leads to a solution.

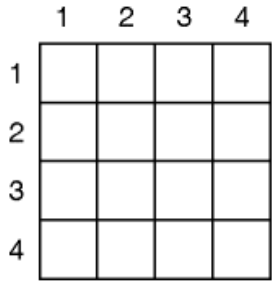
b) If placing the queen in [row, column] leads to a solution then return true.

c) If placing queen doesn't lead to a solution then unmark this [row, column] (Backtrack) and go to step (a) to try other rows.

4) If all rows have been tried and nothing worked,return false to trigger backtracking.

4-Queen Problem

Problem 1 : Given 4 x 4 chessboard, arrange four queens in a way, such that no two queens attack each other. That is, no two queens are placed in the same row, column, or diagonal.



4 x 4 Chessboard

- We have to arrange four queens, Q1, Q2, Q3 and Q4 in 4 x 4 chess board. We will put with queen in ith row. Let us start with position (1, 1). Q1 is the only queen, so there is no issue. partial solution is <1>.
- We cannot place Q2 at positions (2, 1) or (2, 2). Position (2, 3) is acceptable. the partial solution is <1,3>.
- Next, Q3 cannot be placed in position (3, 1) as Q1 attacks her. And it cannot be placed at (3, 2), (3, 3) or (3, 4) as Q2 attacks her. There is no way to put Q3 in the third row. Hence, the algorithm backtracks and goes back to the previous solution and readjusts the position of queen Q2. Q2 is moved from positions (2, 3) to (2, 4). Partial solution is <1, 4>.
- Now, Q3 can be placed at position (3, 2). Partial solution is <1, 4, 3>.

- Queen Q4 cannot be placed anywhere in row four. So again, backtrack to the previous solution and readjust the position of Q3. Q3 cannot be placed on (3, 3) or(3, 4). So the algorithm backtracks even further.
- All possible choices for Q2 are already explored, hence the algorithm goes back to partial solution <1> and moves the queen Q1 from (1, 1) to (1, 2). And this process continues until a solution is found.

All possible solutions for 4-queen are shown in fig (a) & fig. (b)

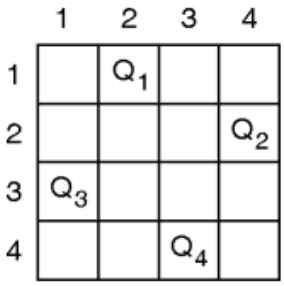


Fig. (a): Solution – 1

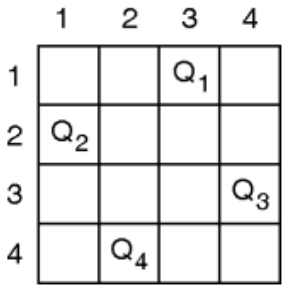
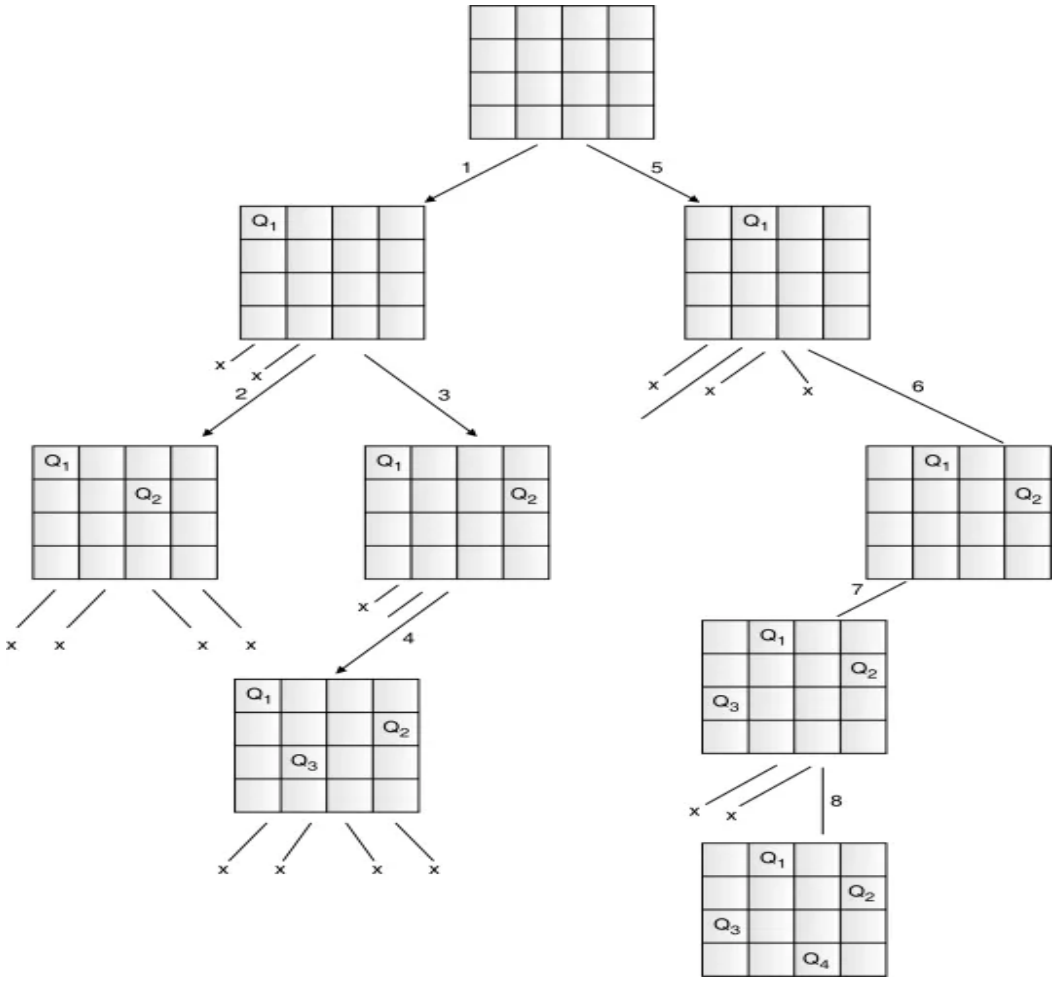


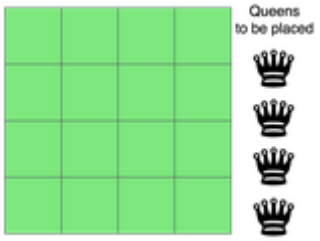
Fig. (b): Solution – 2

Fig. (d) describes the backtracking sequence for the 4-queen problem.

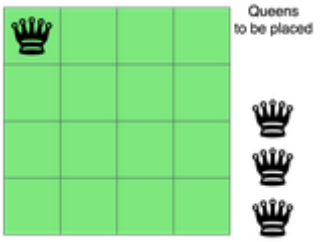


The solution of the 4-queen problem can be seen as four tuples (x1, x2, x3, x4), where xi represents the column number of queen Qi. Two possible solutions for the 4-queen problem are (2, 4, 1, 3) and (3, 1, 4, 2).

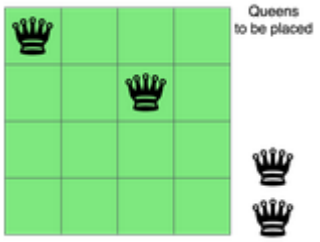
Explanation :

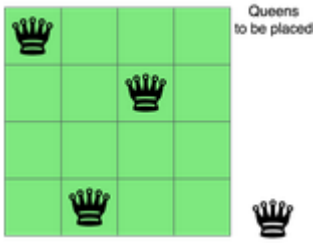


The above picture shows an NxN chessboard and we have to place N queens on it. So, we will start by placing the first queen.



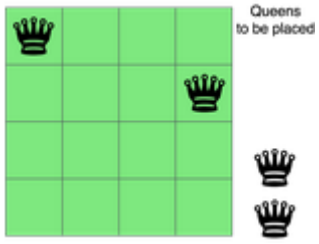
Now, the second step is to place the second queen in a safe position and then the third queen.



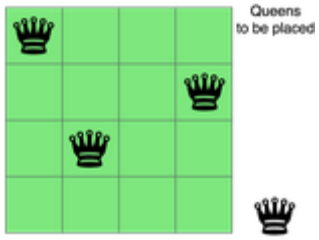


Now, you can see that there is no safe place where we can put the last queen. So, we will just change the position of the previous queen. And this is backtracking.

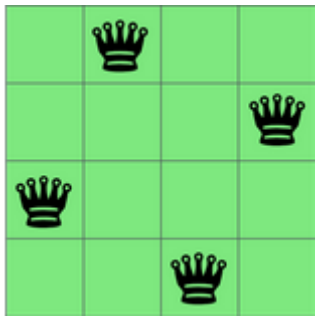
Also, there is no other position where we can place the third queen so we will go back one more step and change the position of the second queen.



And now we will place the third queen again in a safe position until we find a solution.



We will continue this process and finally, we will get the solution as shown below.



We need to check if a cell (i, j) is under attack or not. For that, we will pass these two in our function along with the chessboard and its size - IS-ATTACK(i, j, board, N).

Conclusion: In this way we have explored Concept of Backtracking method and solve n-Queen problem using backtracking method.

Viva Questions:

1. What is backtracking? Give the general Procedure.
2. Give the problem statement of the n-queens problem. Explain the solution
3. Write an algorithm for N-queens problem using backtracking?
4. Why it is applicable to N=4 and N=8 only?