Shri Gajanan Maharaj Shikshan Prasarak Mandal's
SHARADCHANDRA PAWAR COLLEGE OF ENGINEERING
Otur (Dumbarwadi), Tal. Junnar, Dist. Pune - 412409

Assignment No. 1

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

objective :- students should be able to perform non-recursive and recursive programs to calculate fibonacci number and analyze their time & space complexity.

prerequisite :-

1) Basic of Python or Java programming
2) concept of Recursive and non-recursive functions.
3) Execution flow of calculate Fibonacci numbers.

Theory :

1) Introduction to Fibonacci numbers
2) Time & space complexity.

1) Introduction to Fibonacci numbers -

The Fibonacci series, named

after Italian mathematician Leonardo Pisano Bogollo, later known as Fibonacci, is a series formed by Fibonacci numbers denoted as Fn.

What is the Fibonacci series?

The Fibonacci ~~sum~~ series is the sequence of numbers, where every number is the sum of the preceding two numbers, such that the first two terms are '0' & '1'.

Fibonaci sequence formula:

The Fibonacci sequence of numbers "Fn" is defined using the recursive relation with the seed value F0 = 0 & 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 & F1 = 1

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

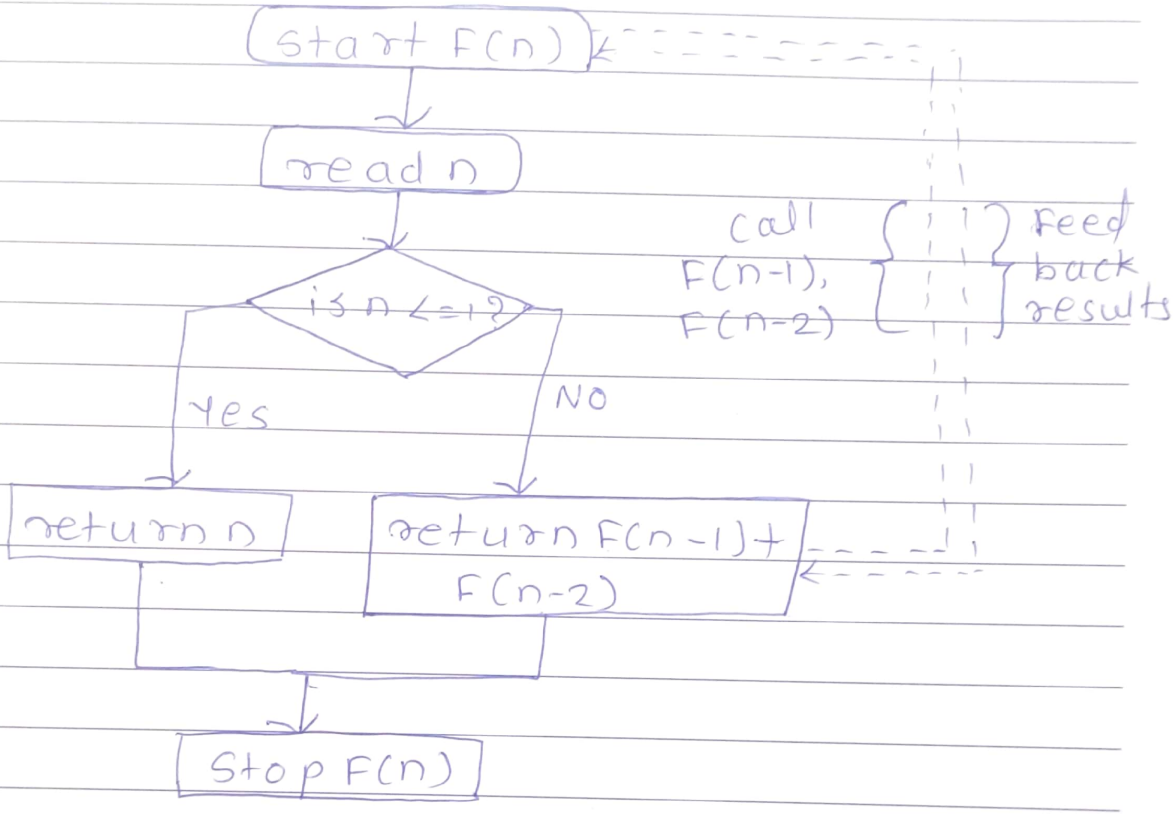
It is noted that the sequence starts with 0 rather than 1, so, F5 should be the 6th term of the sequence.

Method 1 (use Non-recursion):
A simple method that is a direct recursive implements of mathematical recurrence relation is given above.
First, we'll store 0 & 1 in F[0] & 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 as the number at position n in the sequence. Here's a visual representation of this process:

Time & space complexity of space optimized method :-
- The time complexity of the Fibonacci series in T(N) i.e. linear. We have to find the sum of two termand it is ~~rep~~ repeated n times depending on the value of n.
- The space complexity of the Fibonacci series using dynamic programming is O(1)

Method 2 (use Recursion):
Let's start by defining F(n) as the

Function that returns the value of Fn.



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.

Conclusion: Hence after successful implementation of this assignment we have achieves CO4 that is analysis the performance of algorithm & recursive & non-recursive method.

Shri Gajanan Maharaj Shikshan Prasarak Mandal's
SHARADCHANDRA PAWAR COLLEGE OF ENGINEERING
Otur (Dumbarwadi), Tal. Junnar, Dist. Pune - 412409

Assignment No 3.

Title :- Write a program to solve a fractional knapsack problem using a greedy method.

Objective :- student should be able to understand & 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 :-

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 may not produce the best result for all the problem. It's because.

it always goes for the local best choice to produce the global best result.

Greedy Algorithm -
1) To begin with, the solution set 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 & never considered again.

2) knapsack problem :-
You are given the following -
- A knapsack with limited weight capacity
- Few items each having some weight & value.

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
- we can even put the fraction of any item into the knapsack if taking the complete item is not.
- It is solved using the Greedy method.

problem- For the given set of items & knapsack capacity = 60 kg, find the optimal solution for the fractionalknapsack problem making use of greedy approach.

Item	weight	value
1	5	30
2	10	40
3	15	45
4	22	77
5	25	90

n = 5
w = 60 kg
(w1, w2, w3, w4, w5) = (5, 10, 15, 22, 25)
(b1, b2, b3, b4, b5) = (30, 40, 45, 77, 90)

solution -
step 1 - compute the value/weight ratio for each item.

Items	weight	value	Ratio
1	5	30	6
2	10	40	4
3	15	45	3
4	22	77	3.5
5	25	90	3.6

step 2 - sort all the items in decreasing order of their value / weight ratio -

I1	I2	I3	I4	I5
(6)	(4)	(3.6)	(3.5)	(3)

step3 - start filling the knapsack by putting the items into it one by one.

knapsack weight	Items in knapsack	cost
60	ϕ	0
55	I1	30
45	I1, I2	70
20	I1, I2, I5	160

< I1, I2, I5, (20/22) I4 >

Total cost of the knapsack = 160 + (20/22) x 77

= 160 + 70

= 230 units.

conclusion:- Hence after successful Implementation of this assignment we have achieved cos and analyze the performance the algorithm design strategy of Fractional knapsack using greedy method.

Shri Gajanan Maharaj Shikshan Prasarak Mandal's
SHARADCHANDRA PAWAR COLLEGE OF ENGINEERING
Otur (Dumbarwadi), Tal. Junnar, Dist. Pune - 412409

Assignment No. 4.

Title :- Write a program to solve a 0-1 knapsack problem using dynamic programming or branch & bound strategy.

objective :- students should be able to understand & solve 0-1 knapsack problem using dynamic programming.

Prerequisite :-

1) Basic of python of Java Programming
2) Concept of Dynamic programming
3) 0/1 knapsack problem.

Theory :-

Dynamic Programming :-

It is also used in optimization problems. Like divide-and-conquer method, Dynamic programming solves problems by combining the solutions of subproblem.

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.

Knapsack problem:-
You are given the following :-
- A knapsack with limited weight capacity
- Few items each having some weight & value.

1 knapsack problem using Dynamic method:-
consider -
- knapsack weight capacity = w
- Number of items each having some weight & value = n
- 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 0th row & 0th columns

with zeroes as shown -

Step 2 - Start filling the table row wise top to bottom from left to right. Use the following formula -

$T(i,j) = \max \{T(i-1,j), value_i + T(i-1, j-weight)\}$

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

step 3 -

- 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.

Conclusion :- Hence after successful Implementation of this assignment we have achieved CO5 & analyze concept of 0/1 knapsack using two different algorithm design strategies that is Dynamic programming & branch & bound.

Shri Gajanan Maharaj Shikshan Prasarak Mandal's
SHARADCHANDRA PAWAR COLLEGE OF ENGINEERING
Otur (Dumbarwadi), Tal. Junnar, Dist. Pune - 412409

Assignment No 5.

Title :- Design n-queens matrix having first queen placed. Use backtracking to place remaining queens to generate a final n-queen's matrix.

objective :- students should be able to understand & solve n-queen problem & understand basics of Backtracking.

Prerequisite :-

- Basic of python or Java programming
- concept of backtracking method
- 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 may be a soln to your problem.

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 & change the sub-solution & continue again.

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.

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=4 then 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 the row then mark this as part of the solution & recursively check if placing queen here leads to a solution.
b) If placing the queen in leads to a solution then return true.

c) If placing queen doesn't lead to a solution then unmark this [row, column] and
d) goto step(a) to try other rows.
e) If all rows have been tried & nothing worked, return false to trigger backtracking.

conclusion:-
Hence after successful Implementation of this assignment we have achieved CO5 and analyze concept of Backtracking method & solve n-Queen problem using backtracking method.