

### Assignment 3

Name: Mansi Jainendra Tandel

Student ID: W1606463

#### Question 1 (leetcode question 322. Coin Change)

(might use the idea of Rod Cutting problem)

- a. Analyze subproblem structure and give the recursive relationship between problem and its subproblems.

Here, we can solve this problem by this way : First, we need to find the solution for its subproblems and then we can use it for finding the solution to the problem.

For example, for recursive relation, if we have  $y=[1,2,5]$  and  $\text{amount}=11$  then we have to choose the smallest of the following:  $1 + \text{minof}(10)$ ,  $1 + \text{minof}(7)$ . So here we are considering the sum of  $5+5$  and  $5+2$ . So we can use it for finding the solution.

For the recursive relationship, the optimal solution will be  $A[j] = 1 + \min_{1 \leq i \leq k} (A[j-b_i])$  assuming  $j \geq 1$ . Where,  $A[j]$  is the minimum number of coins and  $j$  will be the value that we need to make.  $b_i$  will be the coins values.

- b. Does this problem have optimal substructure property? Explain the reason.

Yes, this problem is having the property of optimal substructure. Here, the solution of the coin change problem can be build up from the optimal solutions of its subproblems. Let us assume that  $A_n$  is the minimum number of coins that are needed for the  $n$  values. Assume that the first coin has the  $b_1$  value. Here, the total number of coins needed will be  $1 + A_{n-b_1}$ . We have taken 1 here as we have taken  $b_1$  value. We can also take the second coin initially and then we can get the optimal solution for  $n-b_2$ . This can be repeated for given number of coins so that the minimum value out of all these numbers can give us the answer. Here, if we get the optimal solution for the subproblems, then we can find the optimal solution. So, we can say that this problem has the property of optimal substructure.

- c. Does this problem have overlapping subproblems? Explain.

Yes, this problem is having overlapping subproblems. Here, the same subproblems are involved multiple times for finding the solution. Let's say that we picked a coin with value  $x$  and  $x$  will be the solution.  $A_{n-x}$  will be the value to be changed and it will be the minimum number of coins. So,  $A_n$  will be  $1 + A_{n-x}$ . here, the value of  $x$  is unknown. So we have to try it for every value of  $x$  that is possible so that we can select the minimum value out of all those. So, we can say that there are overlapping subproblems. So, this problem has both properties, which are optimal substructure and overlapping subproblems. Now, we can use dynamic programming for this problem.

d. Design and implement a DP algorithm and create a successful submission in leetcode.com (use your favorite language)

**Success** Details >

Runtime: 56 ms, faster than 84.57% of C++ online submissions for Coin Change.

Memory Usage: 10.4 MB, less than 82.98% of C++ online submissions for Coin Change.

Next challenges:

[Minimum Cost For Tickets](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
05/20/2021 10:44	Accepted	56 ms	10.4 MB	cpp

```

22  {
23      dp[i][j] = min(dp[i-1][j] , 1 + dp[i][j-coins[i-1]]);
24  }
25  else
26  {
27      dp[i][j] = dp[i-1][j];
28  }
29  }
30  }
31  return dp[n][amount] > amount ? -1 : dp[n][amount];
32  }
33  };

```

Your previous code was restored from your local storage. [Reset to default](#)

Testcase Run Code Result Debugger

**Accepted** Runtime: 0 ms

Your input [1,2,5]  
11

Output 3

Expected 3

Console Use Example Testcases Run Code Submit

- e. If you are using bottom up algorithm, then use give a concrete example to fill a bottom up tabular form.

For example, coins=[1,2,5] and amount=11

	0	1	2	3	4	5	6	7	8	9	10	11
1	0	1	2	3	4	5	6	7	8	9	10	11
2	0	1	1	2	2	3	3	4	4	5	5	6
5	0	1	1	2	2	1	2	2	3	3	2	3

We will require 3 coins to make up the amount.

## Question 2 (leetcode question 64. Minimum Path Sum).

- a. Analyze subproblem structure and give the recursive relationship between problem and its subproblems.

The subproblems here are, to find the sum which comes from left to right or top to bottom as per the given constraints. For the  $m \times n$  grid, the first row only moves to the right and the first column only moves downwards. For each grid blocks, the summation is done. The minimum value sum from top and left in addition with the current cost goes to the table. So, the recursion takes place here. For each grid positions we have the problems and its subproblems.

- b. Does this problem have optimal substructure property? Explain the reason.

Yes, this problem is having the property of optimal substructure. For the  $m \times n$  grid, the array is needed to save the minimum sum path for each point in the  $m \times n$  grid. For the first column in that array, we have to shift down in that and add that element from that grid. For the first row also, we will shift to right and add that element in the array. So, here we are saving the previous sum and then we are comparing it with the minimum sum of the path. Which is the characteristic of optimal substructure property.

- c. Does this problem have overlapping subproblems? Explain.

Yes, this problem is having the property of overlapping subproblems. Here, the same subproblems are occurring multiple times as we have to calculate their sum continuously to find the minimum sum path. The sum which have been calculated previously, have been used for multiple times to get the solution. So we can say that this problem has the overlapping subproblems.

d. Design and implement a DP algorithm and create a successful submission in leetcode.com(use your favorite language)

Description
Solution
Discuss (999+)
Submissions
C++




**Success** Details >

Runtime: **4 ms**, faster than **98.93%** of C++ online submissions for Minimum Path Sum.

Memory Usage: **10.2 MB**, less than **18.11%** of C++ online submissions for Minimum Path Sum.

Next challenges:

Unique Paths
Dungeon Game
Cherry Pickup

Show off your acceptance:   

Time Submitted	Status	Runtime	Memory	Language
05/20/2021 15:09	Accepted	4 ms	10.2 MB	cpp

e. If you are using bottom up algorithm, then use give a concrete example to fill a bottom up tabular form.

For example, the 3\*3 grid is given as following:

1	3	1
1	5	1
4	2	1

Then the bottom up tabular form will be as following:

1	4	5
2	7	6

7	9	7
---	---	---

Here, the minimum path sum will be 7 as per the table constructed above.

### Question3 (leetcode 5, Longest Palindromic Substring)

a. Analyze subproblem structure and give the recursive relationship between problem and its subproblems.

The subproblems are required to find the longest palindromic substring. Here, we have to find palindromes for all the possible substrings of the given length. This will require to have a recursion. We have to maintain a Boolean table of  $[n][n]$ . Then we have to compare them if they are palindrome or not. Let us assume that the table  $[i][j]$  is maintained. We have to check for  $[i+1][j-1]$ , if the value is true and the string values are same then we have to mark it as true, else make it false.

b. Does this problem have optimal substructure property? Explain the reason.

Yes, this problem is having the property of optimal substructure. The longest palindromic substring can be found by using the optimal solutions from its subproblems (which are substrings here for this problem). The substrings are being checked first for the palindromic property. If the substrings are palindromic, then only we are considering it for the longest palindromic substring.

c. Does this problem have overlapping subproblems? Explain.

Yes, this problem is having the property of overlapping subproblems. The solution involves the usage of same subproblem multiple times. Here, if we consider a tabular form, we are using the values over and over again to get to the solution of the problem. For example, for a string having a length of  $n$ , will have a tabular form of  $n \times n$ . While writing to the grid form, the values of the grid, where  $n=n$ , all the values of  $(n,n)$  will be 1, as the letter itself will be palindrome of itself. These values will be used again when we are considering other values for the substring. So we can say that this problem is having the property of overlapping subproblems.

d. Design and implement a DP algorithm and create a successful submission in leetcode.com (use your favorite language)

Description

Solution

Discuss (999+)

Submissions

Success

Details >

Runtime: 176 ms, faster than 42.82% of C++ online submissions for Longest Palindromic Substring.

Memory Usage: 11.4 MB, less than 55.95% of C++ online submissions for Longest Palindromic Substring.

Next challenges:

Shortest Palindrome

Palindrome Permutation

Palindrome Pairs

Longest Palindromic Subsequence

Palindromic Substrings

Show off your acceptance:

f

t

in

C++

Autocomplete

```

28         ans_length = j-index+1;
29         r=index;
30         c=j;
31     }
32 }
33
34 }
35
36 else
37 {
38     if(s[index]==s[j] && dp[index+1][j-1] !=0)
39     {
40         dp[index][j] = 1;
41         if(j-index+1> ans_length)
42         {
43             ans_length = j-index+1;
44             r=index;
45         }
46     }
47 }
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99

```

Testcase

Run Code Result

Debugger

Accepted

Runtime: 0 ms

Your input

"babad"

Output

"bab"

Diff

Expected

"bab"

Time Submitted	Status	Runtime	Memory	Language
05/20/2021 11:08	Accepted	176 ms	11.4 MB	cpp

- e. If you are using bottom up algorithm, then use give a concrete example to fill a bottom up tabular form.

For example, the given string is s= "babad".

	0	1	2	3	4
0	1	0	1	0	0
1		1	0	1	0
2			1	0	0
3				1	0
4					1

For above example, there are two possible answers which are bab and aba.

#### Question 4 (leetcode question 416. Partition Equal Subset Sum).

- a. Analyze subproblem structure and give the recursive relationship between problem and its subproblems.

If the sum of the array is odd, then there cannot be an array of two subsets who are having equal sum, so it will be false. If the sum of the array elements are even, then we need to find the sum divided by 2 and find the subset of that array having the sum which will be equal to the sum divided by 2. The subproblems here will be to find if the subsets are having the equal sum. So, the recursion relationship would be to find the sum considering the last element and reducing it by 1. Also, we have to check without considering the last element sum. If they are true then we will have our solution.

- b. Does this problem have optimal substructure property? Explain the reason.

Yes, this problem is having the property of optimal substructure. Here, the subproblems are giving optimal solutions so that we can save them and later use them in finding the solution to the problem. The subproblems here are to find the sum of the subsets.

- c. Does this problem have overlapping subproblems? Explain.

Yes, this problem is having the property of overlapping subproblems. Here we are finding the sum of the subarrays which we will use later for the finding the sum including the next element from the array. SO we can say that this problem uses the property of overlapping subproblems.

d. Design and implement a DP algorithm and create a successful submission in leetcode.com (use your favorite language)

**Success** Details >

Runtime: 68 ms, faster than 85.62% of C++ online submissions for Partition Equal Subset Sum.

Memory Usage: 341.4 MB, less than 5.22% of C++ online submissions for Partition Equal Subset Sum.

Next challenges: [Partition to K Equal Sum Subsets](#)

Show off your acceptance: [f](#) [t](#) [in](#)

Time Submitted	Status	Runtime	Memory	Language
05/20/2021 11:13	Accepted	68 ms	341.4 MB	cpp

```

11         if(sum%2 != 0) return false;
12         return find(0,0,sum/2,nums,dp);
13     }
14     bool find(int i, int cur, int sum, vector<int> &nums, vector<vector<int>> &dp) {
15         if(cur > sum) return false;
16         if(i == nums.size()) {
17             return cur == sum;
18         }
19         if(dp[i][cur] != -1) return dp[i][cur];
20         return dp[i][cur] = find(i+1, cur+nums[i], sum, nums, dp) || find(i+1, cur, sum,
21             nums, dp);
22     }
23 }

```

Testcase: Run Code Result: Debugger

**Accepted** Runtime: 4 ms

Your input: [1,5,11,5]

Output: true

Expected: true

e. If you are using bottom up algorithm, then use give a concrete example to fill a bottom up tabular form.

For example, it is given that array A=[3,1,1,2,2,1]

	Empty array	{3}	{3,1}	{3,1,1}	{3,1,1,2}	{3,1,1,2,2}	{3,1,1,2,2,1}
0	T	T	T	T	T	T	T
1	F	F	T	T	T	T	T
2	F	F	F	T	T	T	T
3	F	T	T	T	T	T	T
4	F	F	T	T	T	T	T
5	F	F	F	T	T	T	T

### Question 5 (leetcode question 55. Jump Game)

- Analyze subproblem structure and give the recursive relationship between problem and its subproblems.

Here we have an array of non negative integers. For the bottom up approach we need to consider the first approachable index or left most index. Here, we update the good index when the sum of the current index and the number is greater than or equal to the index. The subproblems here are to make a jump from the starting index to the value of that index value.

- Does this problem have optimal substructure property? Explain the reason.

Yes, this problem is having the property of optimal substructure. The subproblems are to make a jump from the starting index to the value of that index value. So the subproblems with the optimal solutions are chosen to find the solution to the jump game.

c. Does this problem have overlapping subproblems? Explain.

Yes, this problem is having the property of overlapping subproblems. Here, the subproblems are needed to be looked upon for finding the solution to the jump game problem. The overlapping subproblems would help find if the jumps that are made are reaching to the last index of the array. So we can say that this problem has overlapping subproblems.

d. Design and implement a DP algorithm and create a successful submission in leetcode.com (use your favorite language)

The screenshot shows a LeetCode submission page for the 'Jump Game' problem. The submission is successful, with a runtime of 74 ms and memory usage of 39.4 MB. The page includes tabs for Description, Solution, Discuss (999+), and Submissions. On the right, there is a list of test cases (1-16) and a 'Your' section. Below the submission details, there are buttons for 'Jump Game II' and 'Jump Game III', and social media icons for Facebook, Twitter, and LinkedIn. At the bottom, there is a table showing submission history.

Time Submitted	Status	Runtime	Memory	Language
05/20/2021 22:07	Accepted	74 ms	39.4 MB	java
05/20/2021 11:33	Accepted	0 ms	40.6 MB	java

e. If you are using bottom up algorithm, then use give a concrete example to fill a bottom up tabular form.

For example, if we have an array [3,2,1,0,4]

The starting index 0 has the value 3 so the jump would be made to the value 0, which is at index 3. The index 3 has the value 0 so it will not make a jump. So it would return false.