NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SC1007 Data Structures and Algorithms **2021/22 Semester 2**

Solution 6: Backtracking & Dynamic Programming

School of Computer Science and Engineering *Nanyang Technological University*

Q1 Given a positive number n, write a pseudo code of backtracking algorithm to count the number of ways to express N as sum of 1, 3 and 4.
Examples:

```
Input :  N = 4
Output : 4
Explanation: 1+1+1+1
             1+3
             3+1
             4
-----------
Input : N = 5
Output : 6
Explanation: 1 + 1 + 1 + 1 + 1
             1 + 4
             4 + 1
             1 + 1 + 3
             1 + 3 + 1
             3 + 1 + 1
```

Draw the recursive tree with n=5.

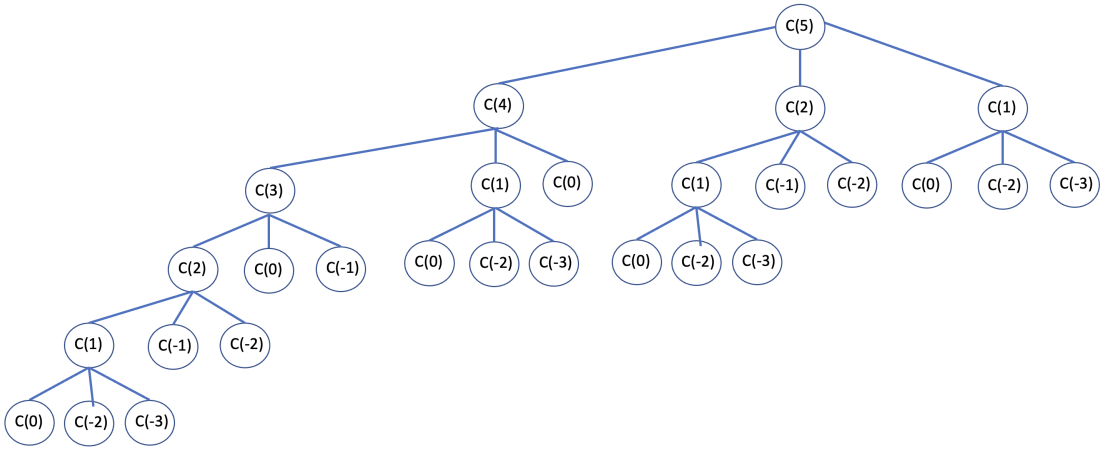
S1 Idea: Let C(n) be the be the number of ways to write n as the sum of 1, 3, and 4. Consider one possible solution with $n = x_1 + x_2 + x_3 + \ldots x_k$. At every step, we can choose x_1 amongst the values of 1, 3, and 4, and the sum of the remaining numbers will be n-1, n-3, and n-4 respectively. If C(n-1), C(n-3) and C(n-4) can be calculated, the final recurrence would be:

$$C(n) = C(n-1) + C(n-3) + C(n-4)$$

```
// function to count the number of
// ways to represent n as sum of 1, 3 and 4
int C(int n)
{
    // check whether n is valid to calculate
    if (n<0)
        return -1;

    // base case
    if (n == 0)
        return 1; //the previous choice is correct
```

```
    int total = 0;
    int a;

    a = C(n-1);
    if (a != -1) //C(n-1) is successfully calculated
        total += a;

    a = C(n-3);
    if (a != -1) //C(n-3) is successfully calculated
        total += a;

    a = C(n-4);
    if (a != -1) //C(n-4) is successfully calculated
        total += a;

    return total;
}
```

Recursive tree:



Q2 Write the pseudo code of dynamic programming for the problem in Q1.

S2 Idea: Use an array DP[n+1] to store the intermediate value of C.

```
// function to count the number of
// ways to represent n as sum of 1, 3 and 4
int C(int n)
{
    int DP[n + 1];

    // base cases
    DP[0] = DP[1] = DP[2] = 1;
    DP[3] = 2;
```

```
    // iterate for all values from 4 to n
    for (int i = 4; i <= n; i++)
        DP[i] = DP[i - 1] + DP[i - 3] + DP[i - 4];

    return DP[n];
}
```

Q3 What is the complexity for the algorithms in Q1 and Q2?

S3 For the recursive tree of backtracking algorithm in Q1, every node has three children. As the depth of tree is n, the complexity of the algorithm will be O(3^n).

For the dynamic programming algorithm in Q2, we only calculate the value of C(i) one time. Therefore, the complexity of the algorithm is O(n).