Data Structures and Algorithms Assignment 6: Backtracking and Dynamic Programming

This assignment contains two parts: 5 MCQ and 3 Programming questions. The templates for Programming Questions 1-3 are given as separated files (Q1_template.c, Q2_template.c, and Q3_template.c). You must use them to implement your functions. You need to submit your code to the https://www.hackerearth.com (Email invitation will be sent to your school account). You need to submit your code to the NTULearn (State your name and your lab group in your submission) as well. Considering April 10, 2024 is a public holiday and April 23/24, 2024 are for lab test 2, deadline for this assignment submission is April 26, 2024 (Friday) 11.59 pm.

Programming Question 1 (Coloring Problem): Write a function to count the number of solutions of the coloring problem with V regions and m colors. You should use the global variable "count" in the template file for this purpose.

int graphColoring(int** graph, int m, int* color, int v).

Herer graph is a matrix to store the adjacency of two regions.

A sample input is given as follows:

```
3
4
0 1 1 1
1 0 1 0
1 1 0 1
1 0 1 0
```

where 3 is the number of colors to be used (numbered from 1-3), 4 is the number of regions. Lines 3-6 specify the adjacency of region 0, region 1, region 2, and region 3 to other regions, respectively. 1 represents two regions are adjacent. The output is 6 solutions, meaning that there are in total 6 solutions to color these 4 regions using 3 colors.

Programming Question 2 (Recursive Function): Given the function F defined as:

F(n) = F(n-1) + 2*F(n-2) - 3F(n-3), with F(0) = 0, F(1) = 1, and F(2) = 2.

Write a function using top-down dynamic programming technique to calculate the value of F(n) with n >= 0.

int top_down_dp(int n)

You need to create and initialize the memory to be accessed globally for this question, and the main function may need some additional codes.

A sample input is: 5. The output is 7, which equals to F(4)+2*F(3)-3*F(2).

Programming Question 3 (0/1 Knapsack): Given n items, where the ith item has the size s_i and the value v_i. Put these items into a knapsack of capacity C.

Write a function:

int bottom_up_dp(int n, int *s, int *v, int C) based on the bottom-up dynamic programming technique to calculate the maximum values of the items that fit in the knapsack. Here n is the number of items, s is the array to store items' size, v is the array to store items' values, and C is the knapsack's capacity.

Hint: use the following recursive formula as in the lecture:

M(i,j) = max {M(i-1,j), M(i-1, j-si) + vi}

You need to create and initialize the memory to be accessed globally for this question.

A sample input is:

```
4
5
2 1 3 2
12 10 20 15
```

where 4 is the number of items, 5 is the knapsack's capacity. 2 1 3 2 are the items' sizes, and 12 10 20 15 are the items' values. The output is 37.