CS 101 Fall 2024 - Quiz 12
December 23, 2024 - 20 Minutes **Name:** **Student ID:**

---

1. (2 points) Honor Code

I promise that I will complete this quiz independently and will not use any electronic products or paper-based materials during the quiz, nor will I communicate with other students during this quiz.

I will not violate the Honor Code during this quiz. ○ True ○ False

2. (3 points) True or False

Determine whether the following statements are true or false.

(a) (1') The original Knapsack problem can be solved in $\Theta(nW)$, where n is the number of items and W is the capacity of the knapsack. When the weight of items in Knapsack problem can be any positive rational number, We can still solve this problem in $\Theta(nW)$ time. ○ True ○ False

(b) (1') Given an array x of length n, find a contiguous sub-array whose sum is maximum. If x_i can be any rational number, we can solve this problem in $\Theta(n)$ time complexity. ○ True ○ False

(c) (1') If there are n houses and m colors, the house coloring problem can be solved in $\Theta(nm)$ time complexity. ○ True ○ False

3. (6 points) Transition of OPT states

Assume we have some dynamic programming problems whose sub-problem is denoted as OPT and their bellman-equations are given. Please give out the time complexity of reaching the answer state(s) with the naive transition. (Your time complexity must reach the upperbound.)

(a) (2') $OPT(i,j) = \begin{cases} 0 & \text{if } i = 0 \\ OPT(i-1,j) & \text{if } a_i > j \\ \max\{OPT(i-1,j), OPT(i-1,j-a_i)+1\} & \text{otherwise} \end{cases}.$

The answer is represented by $OPT(n,m)$. Time: $O($______________$)$.

(b) (2') $OPT(i) = \begin{cases} 1 & \text{if } i \le 1 \\ \max_{j=1}^{i-1}\{[a_j < a_i]OPT(j)+1\} & \text{otherwise} \end{cases}$, where $[a_j < a_i] = 1$ if $a_j < a_i$ otherwise $[a_j < a_i] = 0$. The answer is represented by $\max_{i=1}^{n}\{OPT(i)\}$. Time: $O($______________$)$.

(c) (2') $OPT(i,j) = \begin{cases} a_{i,j} & \text{if } 1 \le i = j \le n \\ a_{i-n,j-n} & \text{if } n < i = j \le 2n \\ \min_{k=i}^{j}\{OPT(i,k) + OPT(k+1,j)\} + a_{i,j} & \text{otherwise} \end{cases}.$

The answer is represented by $\max_{i=1}^{n}\{OPT(i,i+n-1)\}$. Time: $O($______________$)$.

4. (5 points) Counting knapsacks

0-1 Knapsack problem is well known for dynamic programming beginners. Now not only do we want the maximum value that can be put into the backpack, but also the number of methods to get the maximum value. We modify the algorithm, try to fill in the blank in the pseudocode.

Algorithm 1 0-1 Knapsack Problem Function

Require: W is a list of item weights, V is a list of item values, C is the maximum capacity of the knapsack

Ensure: Returns the maximum value that can be put into the knapsack and the number of different methods that can get the maximum value.

1: **function** KNAPSACK(W, V, C)
2:     $n \leftarrow$ length of W
3:     $dp \leftarrow$ array $[0..n][0..C]$ of integer, initialized with 0.
4:     $cnt \leftarrow$ array $[0..n][0..C]$ of integer, initialized with 0.
5:     $cnt[0][0..C] \leftarrow$ initialized with ______________.
6:     **for** $i = 1$ to n **do**
7:         **for** $w = 0$ to C **do**
8:             **if** $W[i-1] \leq w$ **then**
9:                 $dp[i][w] \leftarrow \max(dp[i-1][w], dp[i-1][w - W[i]] + V[i])$
10:                 **if** $dp[i][w] == dp[i-1][w]$ **then**
11:                     $cnt[i][w] \leftarrow$ ______________
12:                 **end if**
13:                 **if** ______________ **then**
14:                     $cnt[i][w] \leftarrow$ ______________
15:                 **end if**
16:             **else**
17:                 $dp[i][w] \leftarrow dp[i-1][w]$
18:                 ______________
19:             **end if**
20:         **end for**
21:     **end for**
22:     **return** $dp[n][C], cnt[n][C]$
23: **end function**

Fill in the blanks in the corresponding line:

Line 5: ____

Line 11: ____

Line 13: ____

Line 14: ____

Line 18: ____