Design & Analysis of Algorithm
Homework 4

Due Date: 11/22/2020 11:45pm (Eastern)

1 Question

You have a set of N dice, where each die d has m faces numbered 1, ..., m. How many ways are there to arrange the dice so taht the total sum is equal to some value M? You don't need to write an algorith here just define how to calculate the answer by breaking the problem into smaller subproblems, and the base case.

2 Question

A contiguous subsequence of a list S is a subsequence made up of consecutive elements of S. For instance, if S is: $5, 15, -30, 10, -5, 40, 10, then 15, -30, 10$ is a contiguous subsequence but $5, 15, 40$ is not. Give a dynamic programming linear-time algorithm for the following task:

Input: A list of numbers $a^1, a^2, ..., a^n$

Output: The contiguous subsequence of maximum sum (a subsequence of length zero has sum zero)

For the preceding example, the answer would be $10, -5, 40, 10$, with a sum of 55. *Hint: For each $j \in (1, 2, ..., n)$, consider contiguous subsequences ending exactly at position j*

3 Question

Given two strings $x = x_1, x_2, ..., x_n$ and $y = y_1, y_2, ..., y_n$ we wish to find the length of their longest common substring that is, the largest k for which there are indices i and k with x_i, $x_{i+1}, ..., x_{i+k-1} = y_j, y_{j+1}, ..., y_{j+k+1}$.

Show how to do this in time $O(mn)$ using dynamic programming.

4 Question

Amortized analysis of stacks. There is a stack with three operations: push, pop, and multipop(k). The multipop(k) operation is implemented as a series of k pops. push and pop each take 1 unit of time, and multipop(k) takes k unit of time. Using amortized analysis, we determined that the average cost of an operation, over n operations, was 2 units of time.

Suppose that in addition to multipop(k), we want to implement a new operation multipush(k), which is a series of k pushes. Can we modify our amortized analysis to this case? Does amoritzed analysis help in this case? Why or why not? (Hint: think about what it means for amortized analysis to help. When would we want to use amortized analysis over standard analysis?)

5 Question

Consider the extensible array data structure that we learned in class. Suppose there is no extra cost for allocating memory. Suppose that we want to add another operation to this data structure: remove, which deletes the last element added. In order to make sure that the array doesn't take up too much space, we say that if the array is at least half empty, we will reallocate memory that is only half the size of the current array, and copy all the elements over. This is basically the opposite of the insertion operation from before.

a) Does amortized analysis make sense here? In other words, does it allow us to get a tighter bound than the standard analysis?

b) Instead of reallocating memory if the array becomes half empty, we reallocate/copy if the array becomes at least 3/4 empty (only 1/4 of the array cells are being used). Analyze the running time of a sequence of n operations using armotized analysis. Hint: this is a straightforward modiciation of the original extensible array analysis. If we shrink an array, how many lements get added before we next double it? If we are deleting elements, how many do we delete before shrinking it?