Homework 5

Winston (Hanting) Zhang

Tuesday, October 18, by 23:00

General Instructions. The following assignment is meant to be challenging, and we anticipate that it will take most of you at least 10–15 hours to complete, so please allow yourself plenty of time to work on it. We highly recommend reading the entire assignment right away — you never know when inspiration will strike. Please provide a formal mathematical proof for all your claims, and present runtime guarantees for your algorithms using asymptotic (big-$O/\Omega/\Theta$) notation, unless stated otherwise. You may assume that all basic arithmetic operations (multiplication, subtraction, division, comparison, etc.) take constant time.

Collaboration. Please carefully check the collaboration policy on the course website. When in doubt, ask an instructor.

Consulting outside sources. Please carefully check the policy regarding outside sources on the course website. Again, when in doubt, ask an instructor.

Submission. Homework submission will be through the Gradescope system. Instructions and links have been provided through the course website and Piazza. The only accepted format is PDF. Starting with this homework, only typed solutions will be accepted, and we highly recommend producing your solutions using LaTeX (the text markup language we are also using for this assignment).

Recommended practice problems (do not hand in): KT Problems 6.2, 6.4, 6.6, 6.8, 6.11, 6.13, 6.16, 6.20, 6.24

Problem 1. [15]

Imagine that you are playing a game resembling Chutes& Ladders with your sibling. It consists of $n + 1$ squares, labeled from 0 to n. You start on square 0. When you reach square n, you win. When it is your turn, you roll a 6-sided die, and move that number of squares forward. There are two types of special squares:

- If square i is a ladder, then it comes with a destination square $d[i] > i$ where the ladder leads. You immediately climb the ladder, and end up on square $d[i]$.
- If square i is a chute, you instantaneously die when you end up on square i by rolling a die. (However, if you climbed to square i using a ladder, you do not die.)
- Not all squares are special — some are neither chutes nor ladders.

When you reach a square i by climbing there, you do not make another move — that is, if you got to $i = d[j]$, and i is the start of a ladder, too, you do *not* follow that ladder, but instead stay there and roll a die on your next move. And if i is a chute, then — as stated above — you do not die.

To avoid annoying special cases, neither square 0 nor square n is a chute or a ladder, and no ladder leads to an undefined square $d[i] > n$.

You have practiced rolling dice in order to gain an advantage, and you have gotten so good at it that you can roll exactly the number you want. Using this skill, you now want to win the game. To do so, you want to calculate the sequence of die rolls that gets you to square n (the winning square) as quickly as possible. If it is impossible to get there (for instance, because all squares except $0, n$ are chutes), then you also want to diagnose that.

Give and analyze a polynomial-time algorithm for solving this problem. If you are following an approach based on recurrences, just the recurrence is *not* enough — you have to give a full pseudo-code implementation with correctness proof and running time analysis. (The running time analysis will likely be short.)

Problem 2. [15]

In class, we saw the algorithm for computing Edit Distance. We motivated it by spell checking, and also briefly mentioned that it is a reasonable (though far from perfect) approach for plagiarism detection. One thing that it is missing is that deletions often happen in chunks. When someone copies text, they may add or leave out whole words, sentences, or paragraphs. Even when typing, we might be more likely to be adding or leaving out multiple letters.

Here, we will consider a refined cost model. You will still be comparing a string x of length n with a string y of length m. Replacing a single character still costs B. But now, we have that for each k, the deletion of k characters in a row in a position costs $A + c \cdot (k - 1)$, and the insertion of any k characters in a row in a position costs $A + c \cdot (k - 1)$. So the previous model was the special case when only $k = 1$ was allowed, but now, we allow the operation for any k.

The goal is now again to find a minimum-cost sequence of operations that turns x into y, except that now, there are more different operations available.

Give and analyze a polynomial-time algorithm for solving this problem. If you are following an approach based on recurrences, just the recurrence is *not* enough — you have to give a full pseudo-code implementation with correctness proof and running time analysis. (The running time analysis will likely be short.)

Chocolate Problem: 1 chocolate bar

Reminder: If you solve a chocolate problem (which you can do in groups of size up to 3), please e-mail David with the solution — do not submit it on Gradescope. Also, feel free to list preferences or dietary restrictions for/against particular types of chocolate.

This problem is a significant generalization of the problem that you saw in discussion section a week ago. You want to tile the hallway of your home with rectangular tiles of size 2×1. You can use them horizontally or vertically. Your hallway is of size $k \times n$, and your tiling must not let any of those kn squares be uncovered by a tile. To illustrate tilings, two valid tilings of a 3×4 rectangle are given in Figure **??**.

Your goal is to find out how many *distinct* legal tilings there are for your hallway. If two tilings are mirrors or rotations of each other, we still count them as distinct. Give an algorithm with running time $f(k) \cdot p(n)$ for this problem, where $p(n)$ must be a polynomial, while $f(k)$ is allowed to be (singly) exponential.

(Hint: start by thinking about small values of k, like $k = 3$ or $k = 4$. $k = 2$ can also be helpful, but it is perhaps too special.)