CSE 565: Algorithm Design and Analysis **Due 2022-10-21T23:59**

Problem Set 3

Instructor: Dr. Antonio Blanca
TA: Jeremy Huang

Release Date: 2022-10-07

Notice: Type your answers using LaTeX and make sure to upload the answer file on Gradescope before the deadline. Recall that for any problem or part of a problem, you can use the "I'll take 20%" option. For more details and the instructions read the syllabus.

Problem 1. Too Different

Recall in class we use dynamic programming to calculate the edit distance between two strings. Given two strings A and B of the same length n, and a threshold $k, 0 < k < n$, design an algorithm to check whether the edit distance between A and B is less than k. Your algorithm should run in $O(nk)$ time.

Problem 2. LCS

Let $A = a_1 a_2 \cdots a_n$ and $B = b_1 b_2 \cdots b_m$ be two strings. Design a dynamic programming algorithm to compute the longest common subsequence between A and B, i.e., to find the largest k and indices $1 \leq i_1 < i_2 < \cdots < i_k \leq n$ and $1 \leq j_1 < j_2 < \cdots < j_k \leq m$ such that $A[i_1] = B[j_1], A[i_2] = B[j_2], \cdots, A[i_k] = B[j_k]$. Your algorithm should run in $O(\text{mn})$ time.

Problem 3. Mind the Gap

In lecture, we learned an $O(\text{mn})$ dynamic programming (DP) algorithm to compute edit distance of two strings, aka sequence alignment. Similar algorithms are widely used in bioinformatics where scientists often compare similarity of two given DNA sequences.

Here you are required to design a DP algorithm to compute the edit penalty of the optimal alignment of two given strings, but with a higher gap penalty. The biological implication is that mutations (i.e. mismatches) happens much more frequently than insertions/deletions (i.e. gaps) in a DNA sequence. Also, extending a gap (i.e. a longer gap) is more favored than opening a new gap (i.e. two short gaps). More specifically, in the edit distance problem, the penalty (i.e., the edit distance) is incremented by 1 for a mismatch or a gap between S_1 and S_2. In this problem with refined gap model, the penalty is still incremented by 1 for a mismatch, but by a when opening a new gap, and by b when extending an existing gap $(a > b \geq 3)$. For example, the total penalty for alignment AA_CC $\times$ AAGGCT is $(a+b+1)$ where a is for opening the gap at position 3, b for extending the gap at position 4 , and 1 for a mismatch at position 6 .

Problem: Sequence alignment with gap penalty.

Input: Two strings $S_1[1 \cdots n]$ and $S_2[1 \cdots m]$ over alphabet $\Sigma = \{A, C, G, T\}$, and a, b with $a > b \geq 3$.

Output: the alignment between S_1 and S_2 with minimized total penalty.

Design a dynamic programming algorithm for the above problem. Show correctness and complexity analysis of your algorithm. Your algorithm should run in $O(\text{mn})$ time.

Hint: Use three DP tables. One for ending with a gap in S_1, one for ending with a gap in S_2, one for match/mismatch only.

Problem 4. Line Fill

Consider a typesetting software (such as TEX) that converts a paragraph of text into PDF documents (say). The input text is a sequence of n words of width $w_1, \ldots, w_n$. The software needs to somehow minimize the amount of extra spaces, as follows. Every line has width W, and if it contains words i through j, then amount of extra spaces on this line is $W - j + i - \sum_{i \le k \le j} w_k$, because we leave one unit of space between words. The amount of extra space on each line must be nonnegative to avoid overflowing of words. Our goal is to minimize the sum of squares of extra spaces on all lines except the last. Give a dynamic programming algorithm for this problem that runs in $\mathcal{O}(n^2)$. Correctness proof is optional for this question.

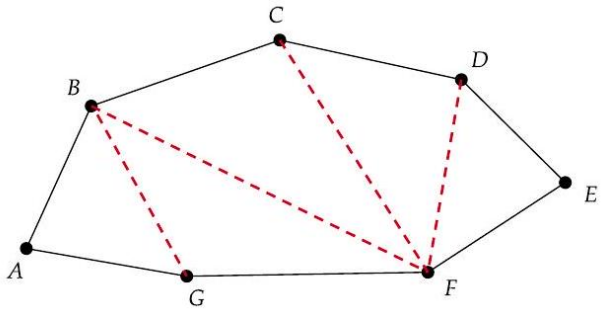
Problem 5. Re-pea-pea-pea-pea-peat

Given a string S of length n and a positive integer k, find the longest continuous repeat of a substring which length is k. For example, when $S =$ ACGACGCGCGACG, if $k = 2$, the longest continuous repeat would be *CGCGCG* composed of *CG* repeating 3 times. And if $k = 3$, the longest continuous repeat would be *ACGACG* composed of *ACG* repeating twice. Design a dynamic programming algorithm to solve this problem and analyze its running time.

Problem 6. Short Poly

You have a wood plank in a shape of a convex polygon with n edges, represented as the (circular) list of the 2D coordinates of its n vertices. You want to cut it into $(n-2)$ triangles using $(n-3)$ non-crossing diagonals, such that the total length of the cutting trace (i.e., total length of the picked diagonals) is minimized. Design a dynamic programming algorithm runs in $O\left(n^3\right)$ time to solve this problem. Your solution should include definition of subproblems, a recursion, how to calculate the minimized total length (i.e., the termination step), and an explanation of why your algorithm runs in $O\left(n^3\right)$ time.

For example, in the convex polygon given below, one possible way to cut it into triangles is illustrated with red dashed lines, and in this case the total length of cutting trace is $\overline{BG} + \overline{BF} + \overline{CF} + \overline{DF}$.



Problem 7. Push Shapes

There are n dominos of height h located at positions $x_1, \ldots, x_n$ on a line. All the positions are distinct. When a domino falls down, it either falls left with probability p or to the right with probability $1 - p$. If the domino hits another domino in its way down, that domino will fall in the same direction as the domino that hit it. A domino can hit another domino if and only if the distance between them is strictly less than h. For instance, suppose there are 4 dominos located at positions 1, 3, 5 and 8 and $h = 3$. The domino at position 1 falls right. It hits the domino at position 3 and it starts to fall too. In it's turn it hits the domino at position 5 and it also starts to fall. The distance between 8 and 5 is exactly 3, so the domino at position 8 will not fall.

While there are dominos standing, a kid will select either the leftmost standing domino with probability q or the rightmost standing domino with probability $1 - q$ and will make it fall (it will fall to the left with probability p or

to the right with probability $1-p$). We would like to know the expected total length of the line covered with fallen dominos after all of them are pushed over.

Write a DP algorithm to solve this problem. The running time should be $O(n^2)$. Hint: make sure to work out a few small examples first.

Problem 8. Price Spread

Cécile has a copper pipe of length n inches and an array of nonnegative integers that contains prices of all pieces of size smaller than n. She wants to find the maximum value she can make by cutting up the pipe and selling the pieces. For example, if length of the pipe is 8 and the values of different pieces are given as following, then the maximum obtainable value is 22 (by cutting in two pieces of lengths 2 and 6).

length	1	2	3	4	5	6	7	8
price	1	5	8	9	10	17	17	20

Give a dynamic programming algorithm so Cécile can find the maximum obtainable value given any pipe length and set of prices. Clearly describe your algorithm, prove its correctness and runtime.

Problem 9. Apples of My Tote

You decide to go apple picking at the Natural Apple Orchard, where the trees grow apples whose mass is always a (positive) natural number of grams. You pay for a bag that can hold up to m grams of apples and set out to pick as many grams of apples as you can. You were told that there are n unique masses $a_1..a_n$ among the remaining apples. Assuming there are d_i many apples that weigh a_i grams in the orchard, what is the maximum amount of apples that can you pick? Find an $O(nm\log m)$ time DP algorithm that answers this question. Please also give the space complexity of your algorithm.

Hint: Think about the binary representation of d_i.