Homework 6

CS41, Spring 2023 — Due Sunday, March 12

1. You need to choose a route through a network in which nodes charge you for service and each link has a possibility of failing.

   This network is modeled as a digraph $G = (V, E)$, with $|V| = n$ and $|E| = m$, and $m \geq n$. You are at some node $s \in V$, and you need to send a message to another node $t \in V$. Every link in this network will successfully transmit a message with probability $p \in (0, 1)$, independent of all other links.[1] Each node $v \in V \setminus \{s, t\}$ will charge you c_v cents to relay your message to one of its successors. (The cost c_v may be different for each node v.) Your message needs to have probability at least q, for some given $q \in (0, 1)$, of making it to t intact.

   Design a dynamic programming algorithm to find the minimum cost of sending this message. Your algorithm should run in $O(m \log_p q)$ time. Briefly explain the reasoning behind your algorithm and your running time analysis.

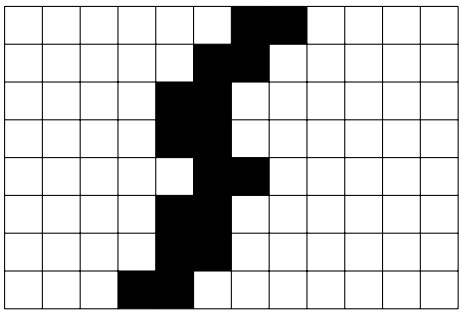
2. A soccer stadium is hosting a match between two rival teams with unruly fan bases, so they need to put up a barrier in the stands to separate opposing hooligans. This barrier will occupy some seats, and the goal is to minimize the resulting reduction in ticket revenue.

   The seats are arranged in a grid with m rows and n columns. The completed barrier will extend from row 1 to row m, and it will occupy a pair of adjacent seats in each row. There is no restriction on where in row 1 the barrier starts, and it is allowed to zig-zag left and right, but it can only shift by one seat per row.

   That is, if the barrier occupies seats (i, j) and $(i, j+1)$ in row i, then the seats it occupies in row $i+1$ must be either $(i+1, j-1)$ and $(i+1, j)$; $(i+1, j)$ and $(i+1, j+1)$; or $(i+1, j+1)$ and $(i+1, j+2)$.

   (There is no requirement about balancing the number of seats on each side of the barrier. For example, placing the entire barrier in the rightmost two columns is allowed.)

   Here is an example of a valid barrier with $m = 8$ and $n = 12$:

   There is an array $P[1..m][1..n]$ of positive prices for the seats. The *cost* of a complete barrier is the sum of the prices of the $2m$ seats it occupies. Design a dynamic programming algorithm that computes the minimum possible cost of a complete barrier and runs in $O(mn)$ time. Briefly explain the reasoning behind your algorithm and your running time analysis.

[1] In particular, this means that the probability of a messages being successfully transmitted over ℓ consecutive links is p^ℓ.

3. We saw in class how to calculate the edit distance between two strings, and I mentioned that this is often applied to comparing genetic sequences. Let's make this edit distance more realistic for the way genomes evolve, in two ways:

   - Different bases are more or less likely to be replaced by other bases. To model this, we will have a function $c : \{A, C, G, T\}^2 \to \mathbb{N}$, giving the cost of replacing each nucleobase with each other one. For example, $c(A, G)$ is the cost of replacing A with G. Naturally, $c(A, A) = c(C, C) = c(G, G) = c(T, T) = 0$.
   - In nature, large chunks of DNA are often deleted or inserted simultaneously, so many *consecutive* deleted or inserted bases are less indicative of evolutionary difference than many *separate* deletions or insertions. For example, the pair

     $$\begin{array}{cccccccccc} A & G & - & - & - & - & - & T & A & C \\ A & G & A & A & A & A & A & T & A & C \end{array}$$

     is considered more similar than the pair

     $$\begin{array}{cccccccccc} A & - & G & - & T & - & A & - & C & - \\ A & A & G & A & T & A & A & A & C & A \end{array}$$

     even though they both involve five insertions of A, because the insertions in the first case are all consecutive. (It is irrelevant for this example that the inserted characters are all the same. The important difference is the number of contiguous blocks.)

     To model this, we will have two positive integer constants p and q, where any k consecutive insertions or k consecutive deletions incur a cost of $p + kq$. In other words, p is the cost to *start* a chain of insertions (or a chain of deletions), and q is the cost per base (including the first) of *continuing* that chain. Thus, the costs of the two alignments above would be $p + 5q$ and $5p + 5q$, respectively

   Suppose that you get as input two n-nucleobase strings $x[1..n]$ and $y[1..n]$, the function c, and the constants p and q. Carefully define:

   - a set V of vertices,
   - a set E of directed edges,
   - a weight function $w : E \to \mathbb{N}$,
   - a start vertex $s \in V$, and
   - a terminal vertex $t \in V$,

   such that $G = (V, E)$ is a DAG and finding the minimum cost of editing x into y is equal to the minimum total weight of an s–t path in G. Briefly explain your solution, and argue that finding that total weight could be done in $O(n^3)$ time.

   (*Note:* The answer to this problem should be significantly shorter than the problem itself.)