CS 473 ✧ Spring 2017

Homework 2

Due Wednesday, February 8, 2017 at 8pm

There are only two problems, but the first one counts double.

---

1. Suppose you are given a two-dimensional array $M[1..n, 1..n]$ of numbers, which could be positive, negative, or zero, and which are *not* necessarily integers. The ***maximum subarray problem*** asks to find the largest sub of elements in any contiguous subarray of the form $M[i..i', j..j']$. In this problem we'll develop an algorithm for the maximum subarray problem that runs in $O(n^3)$ time.

   The algorithm is a combination of divide and conquer and dynamic programming. Let L be a horizontal line through M that splits the rows (roughly) in half. After some preprocessing, the algorithm finds the maximum-sum subarray that crosses L, the maximum-sum subarray above L, and the maximum-sum subarray below L. The first subarray is found by dynamic programming; the last two subarrays are found recursively.

   (a) For any indices i and j, let $Sum(i, j)$ denote the sum of all elements in the subarray $M[1..i, 1..j]$. Describe an algorithm to compute $Sum(i, j)$ for all indices i and j in $O(n^2)$ time.

   (b) Describe a simple(!!) algorithm to solve the maximum subarray problem in $O(n^4)$ time, using the output of your algorithm for part (a).

   (c) Describe an algorithm to find the maximum-sum subarray ***that crosses L*** in $O(n^3)$ time, using the output of your algorithm for part (a). *[Hint: Consider the top half and the bottom half of M separately.]*

   (d) Describe a divide-and-conquer algorithm to find the maximum-sum subarray in M in $O(n^3)$ time, using your algorithm for part (c) as a subroutine. *[Hint: Why is the running time $O(n^3)$ and not $O(n^3 \log n)$?]*

   In fact, the subproblem in part (c) — and thus the entire maximum subarray problem — can be solved in $n^3/2^{\Omega(\sqrt{\log n})}$ time using a recent algorithm of Ryan Williams. Williams' algorithm can also be used to compute all-pairs shortest paths in the same slightly subcubic running time. The divide-ando-conquer strategy itself is due to Tadao Takaoka.

   There is a simpler $O(n^3)$-time algorithm for the maximum subarray problem, based on Kadane's $O(n)$-time algorithm for the one-dimensional problem. (For every pair of indices i and i', find the best subarray of the form $M[i..i', j..j']$ in $O(n)$ time.) It's unclear whether this approach can be sped up using Williams' algorithm (or its predecessors) without the divide-and-conquer layer.

   An algorithm for the maximum subarray problem (or all-pairs shortest paths) that runs in $O(n^{2.99999999})$ time would be a major breakthrough.

2. The Doctor and River Song decide to play a game on a directed acyclic graph G, which has one source s and one sink t.[1]

   Each player has a token on one of the vertices of G. At the start of the game, The Doctor's token is on the source vertex s, and River's token is on the sink vertex t. The players alternate turns, with The Doctor moving first. On each of his turns, the Doctor moves his token forward along a directed edge; on each of her turns, River moves her token *backward* along a directed edge.

   If the two tokens ever meet on the same vertex, River wins the game. ("Hello, Sweetie!") If the Doctor's token reaches t or River's token reaches s before the two tokens meet, then the Doctor wins the game.

   Describe and analyze an algorithm to determine who wins this game, assuming both players play perfectly. That is, if the Doctor can win *no matter how River moves*, then your algorithm should output "Doctor", and if River can win *no matter how the Doctor moves*, your algorithm should output "River". (Why are these the only two possibilities?) The input to your algorithm is the graph G.

[1]The labels s and t may be abbreviations for the Untempered **S**chism and the **T**ime Vortex, or the Shining World of the Seven Systems (otherwise known as Gallifrey) and Trenzalore, or Skaro and Telos, or Something else Timey-wimey.