Homework #1 Due: September 12, 2019

This HW has three regular problems, and one programming problem. You may work in groups of (up to) three on these problems. But the write-up you submit must be written by you alone. Include the names of your group members on your submission.

Solutions to the three written problems should be submitted as a single PDF file using gradescope, with the answer to each problem starting on a new page. Unless otherwise stated, you should prove all of your answers. For example when presenting an algorithm you should prove correctness and the necessary running time bounds.

Submission instructions for the programming problem will be posted on Piazza.

0. Practice Exercise (do not turn in): Solving Recurrences

Give a tight asymptotic bound for the following recurrences. In each case explain the technique you use and why your answer is correct. For all these problems $T(1) = 1$. (Hint: In some cases it’s useful to write out the recursion tree.)

(a) $T(n) = 2T(\lfloor n/2 \rfloor) + 1$
(b) $T(n) = 3T(\lfloor n/2 \rfloor) + n \log n$
(c) $T(n) = 3T(\lfloor n/2 \rfloor) + n^3$
(d) $T(n) = T(\lfloor \sqrt{n} \rfloor) + 1$
(e) $T(n) = n^{2/3} \cdot T(\lfloor n^{1/3} \rfloor) + n$.

(25 pts) 1. Finding the Longest Average-Edge-Length Cycle

You’re given a directed graph $G = (V, E)$ where each edge $e \in E$ has a positive length $w(e)$. The lengths satisfy $1 \leq w(e) \leq B$. The number of vertices is $n$, and the number of edges is $m$. The average-edge-length of a cycle is the total length of the cycle divided by the number of edges on the cycle.

Give an algorithm to approximately compute the average-edge-length of the cycle with the greatest average-edge-length in $G$. The running time of your algorithm should be $O(nm \log(B))$. The approximation should be within 1% of the correct answer.
(25 pts) 2. **Counting Subgraphs**

Throughout this problem we will assume that $G = (V, E)$ is an undirected graph with $n$ vertices and $m$ edges. Let $A$ be its $n \times n$ adjacency matrix. So $A_{i,j} = A_{j,i}$ for all $i$ and $j$, and $A_{i,j} = 1$ if $\{i, j\} \in E$ and $A_{i,j} = 0$ if $\{i, j\} \notin E$.

In an undirected graph, a $k$-clique is a set of $k$ distinct vertices such that there is an edge between every pair of these vertices. In class we gave an $O(n^{2.82})$ algorithm to determine if $G$ has a 3-clique.

(a) We also gave a formula for computing the number of 3-cliques. That is, the number of distinct subsets of three vertices that form a 3-clique.

$$\frac{1}{6} \sum_{A_{ij} \neq 0} (A^2)_{ij}$$

Prove this formula is correct.

(b) Give an $O(n^\omega)$-time algorithm for counting the number 4-cliques in $G$ for $\omega < 4$.

(c) Give an $O(n^\omega)$ time algorithm (for $\omega < 3$) to count the number of distinct 4-cycles in $G$. We can define a 4-cycle as a set of four edges such that those four edges involve four vertices, each one occurring in two of the edges.

So, in the complete graph of four vertices there are just three 4-cycles, with the following edge sets:

$$\{\{1, 2\}, \{1, 4\}, \{2, 3\}, \{3, 4\}\}$$

$$\{\{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}\}$$

$$\{\{1, 2\}, \{1, 3\}, \{3, 4\}, \{2, 4\}\}$$
3. **Summarize Data**

Let $X = [x_1, x_2, \ldots, x_n]$ be a given sequence of real numbers. Also let $C > 0$ be a given constant. The goal in this problem is to approximate $X$ with a piecewise constant function, in a specific way which is described below.

A sequence of numbers $Y = [y_1, y_2, \ldots, y_n]$ can approximate $X$ with the a cost defined as follows:

$$\text{Cost}(X, Y) = \sum_{i=1}^{n-1} C\delta(y_i, y_{i+1}) + \sum_{i=1}^{n} (x_i - y_i)^2$$

Here $\delta(y_i, y_{i+1})$ is 0 if $y_i = y_{i+1}$ and 1 otherwise. So the first summation above costs $C$ each time the $y$s change. The second term measures how closely the $x$s and $y$s match. The goal of this problem is to give an algorithm that computes the minimum over all possible $Y$s of $\text{Cost}(X, Y)$. Let’s call this quantity $Q(X, C)$.

(a) Develop a dynamic programming algorithm to compute $Q(X, C)$ in $O(n^3)$ time. (Make sure you state the meaning of the recurrence variable, state the recurrence, prove the recurrence is correct, and analyze the running time of the resulting algorithm.)

(b) Now improve the algorithm so that it runs in $O(n^2)$ time.

Hint (for both parts): To do this it will be useful to make use of some of the *moments* of the sequence $X$. The $p$th moment of $X$, $M_p$ is defined as follows:

$$M_p = \sum_{i=1}^{n} x_i^p$$

Note that $M_0 = n$, and $M_1$ is just the sum of all the elements of $X$.

We can also generalize this definition for a range, so $M_p[j, k] = \sum_{i=j}^{k} x_i^p$.

For a range $[j, k]$ what value of $y$ minimizes $\sum_{i=j}^{k} (x_i - y)^2$? With this choice of $y$ what is the value of this sum?

The moments that will be useful are the ones of order 0, 1 or 2. (I.e. $p = 0, 1, \text{or } 2$.)
Programming: Cheapest Tree Separation

There are $N$ cities, numbered from 1 through $N$, connected by $N - 1$ roads, forming a weighted tree. Countries $A$ and $B$ each occupy a set of cities (no city is occupied by both countries, and some cities may not be occupied at all).

To stop fighting between the two countries, you want to destroy roads such that no city occupied by country $A$ is connected to a city in country $B$. Destroying a road of length $x$ costs $x$ dollars. What is the minimum cost required? The time limit is 3 seconds.

INPUT: The first line contains $N$ ($1 \leq N \leq 200000$), the number of cities. The next $N - 1$ lines contain three integers $u, v, \ell$ ($1 \leq u, v \leq N, 1 \leq \ell \leq 10^9$), indicating that there is a road of length $\ell$ between cities $u$ and $v$. The next line contains an integer $h$ ($1 \leq h \leq N$), the number of cities occupied by country $A$, followed by $h$ integers, describing the numbers of the cities occupied by country $A$. The last line contains an integer $k$ ($1 \leq k \leq N$), the number of cities occupied by country $B$, followed by $k$ integers, describing the numbers of the cities occupied by country $B$.

OUTPUT: Output a single integer, the minimum cost required to separate countries $A$ and $B$.

For example if the input is:

```
6
2 1 5
4 2 4
5 2 1
3 1 2
6 3 7
2 5 6
1 4
```

then the output is

```
3
```

Hint: Consider another view of this problem:

Input: An edge weighted tree $T = (V, E)$ plus a given a partial coloring of the vertices, say, $c' : V' \subset V \rightarrow \{A, B\}$.

Output: A coloring $c : V \rightarrow \{A, B\}$ that agrees with $c'$ on $V'$ that minimizes the cost of the $A \ B$ edges. That is minimize

$$\sum_{(u,v) \in E, \ c'(u)=A, \ and \ c'(v)=B} cost(u, v).$$
B1 (Bonus) **Counting Hop-Paths**

Let \( A[1], A[2], \ldots, A[n] \) be an array of positive integers.

A *valid path* in the array starts at position 1, and hops to position \( 1 + A[1] \), then continues, where each subsequent hop from position \( j \) is to position \( j + A[j] \) or to position \( j + h \) (where \( h \) is the length of the previous hop). The valid path must end at position \( n \).

The goal is to compute, given the array \( A \), the number of distinct valid paths. Two valid paths are distinct if they visit a different set of cells. The running time of your algorithm should be \( O(n\sqrt{n}) \). As usual prove that your algorithm is correct and satisfies the running time bound.