Rules: Work in groups of 3. Submit one set of solutions per group, and remember to put all your names on it. We'd prefer if you typed your answers, but if your handwriting is neat and legible, written work is fine. You may refer to whatever sources you want, but *CITE* your references! If you get a great idea from somebody, give appropriate credit: it won't lower your grade, and it's the right thing to do. If you have any questions or errors to report, email `ryanw@cs.cmu.edu`.

1 Counting paths.

1. Assume $G = (V, E)$ is a directed graph with no weights on the edges. (You may think of this as the weights having 1 for each edge.)

Give a polynomial time algorithm that, for every pair of nodes $(u, v) \in V \times V$, computes the number of *distinct* shortest paths. That is, the algorithm computes "all-pairs *number of shortest paths*" for G.

2. Prove that for any G, your algorithm correctly computes the number of distinct shortest paths between any pair of vertices u and v.

3. Prove what the running time of your algorithm is (in big-O notation, of course).

Note: Extra points will be given for the group with the fastest (and proven) running time. Here are some ground rules for what's "faster": if you can replace n^2 with m in your running time expression, that's considered faster. If you can replace m with n, that's considered faster. So $O(n)$ is faster than $O(m)$ is faster than $O(n^2)$, for example.

2 Heapifying in $O(n)$.

Is it possible to build a binomial heap with n elements in $O(n)$ worst case time, using the operations for binomial heaps given in class (and in Kozen)? "Worst case" time means that for any set of n elements, the running time is at most $O(n)$; amortized analysis is not required for this problem.

Explicitly, suppose you're given only n elements on a queue, and an empty binomial heap. Is there a linear time algorithm that always returns a binomial heap containing exactly these n elements?

If you answer yes, give an algorithm, prove it runs in $O(n)$ time, and prove it's correct.

If you answer no, prove why not.

3 A paradox.

In the following, the *general APSP* is defined as all-pairs shortest paths on general graphs where the edge weights are positive.

Consider the following three statements:

1. Strassen's algorithm for multiplying $n \times n$ matrices runs in $O(n^{2.81})$ time (in terms of arithmetic operations).

2. The general APSP can be solved in $O(M(n))$ time, where $M(n)$ is the number of arithmetic operations necessary to multiply two $n \times n$ matrices.

3. It is an open problem whether or not the general APSP can be solved in *less* than $O(mn + n^2 \log n)$ time. Observe this is $O(n^3)$ on graphs with at least ϵn^2 edges for some $\epsilon > 0$.

Taking all three of these statements at face value, there is an obvious contradiction: $O(n^{2.81})$ is less than $O(n^3)$. So why are all three of these statements true? (Hint: Under what conditions does Strassen's algorithm work? Under what conditions does the all-pairs shortest-paths algorithm in Kozen work?)

4 Open problem of the week.

This one naturally follows from the previous problem. For general graphs with arbitrary (real-numbered) edge weights, can all-pairs shortest paths be computed in less than $O(mn + n^2 \log n)$ time? (For example, could it be doable in $O(mn/\log n)$ time?) Restrictions on the general problem (restricting edge weights or considering undirected graphs) do permit such algorithms, for example Shoshan + Zwick, and Pettie + Ramachandran. Both papers are downloadable online if you Google them.

References:

S. Pettie and V. Ramachandran. *Computing Shortest Paths with Comparisons and Additions.* In Proceedings of ACM-SIAM Symposium on Discrete Algorithms, 2002.

A. Shoshan and U. Zwick, *All pairs shortest paths in undirected graphs with integer weights.* In Proceedings of IEEE Foundations of Computer Science, 605–614, 1999.