

CSE 101
Practice Quiz 2, Spring, 2013

Answer all questions. Give informal (at least) proofs for all answers. Grading will be on completeness and logical correctness, and if applicable, efficiency, as well as correctness. Out of 80 points.

Which version of Dijkstra do we use? 20 points each We saw two implementations of Dijkstra's algorithm in class, one using arrays and the other using a heap. For each of the following kinds of graph, explain which implementation you would use and give a time analysis for that version on the graph in question. When the graphs described are undirected, that means in the directed graph, there are edges in both directions.

1. A complete bipartite graph with $n/2$ nodes on each side, and $n/2$ on the other, and edges from every node to every node on the opposite side.

Every node has degree $n/2$ in this graph, so $m = 1/2n^2$. This is very dense, within a constant factor of the maximum density. So the best version to use is the array version, which gives time $O(n^2)$.

2. A randomly chosen graph on n nodes, where we pick edges from each node to 10 randomly selected neighbors.

Whichever neighbors are chosen, each node has degree 10. So $m = 10n = O(n)$, which is sparse. The best version is the heap data structure, which takes time $O((n + m) \log n) = O(n \log n)$ since $m = O(n)$ in this case.

Using algorithms to solve new problems— 40 points Explain how we can modify or use one of the known graph search algorithms to solve the following problem in the given time:

Shortest distance between sets of points You are given a directed graph G with non-negative edge weights, and two sets of nodes $S, T \subseteq V$. You wish to find the smallest distance of any path that starts at some node $u \in S$ and ends at some node $v \in T$. Show how to do this in $O((n + m) \log n)$ time.

Construct a new graph G' where we add two new nodes s and t and add edges from s to each element of S with weight 0, and edges from each element of T to t with weight 0.

We claim that the minimum distance from a node in S to a node in T in G is the same as the minimum distance from s to t in G' . Let p be a path from some $u \in S$ to $v \in T$ in G . Then let p' be p adding the edge from s to u first, and the edge from v to t last. Since the edges we added have weight 0, the distance of p is equal to the distance of p' . Thus, the distance from s to t in G' is at most that from any $u \in S$ to any $v \in T$ in G . On the other hand, if p' is a path from s to t in G' , its first edge must go from s to some $u \in S$, its middle from u to some $v \in T$, and its last edge from v to t . Again, the cost of p' is the same as the cost of its middle path from u to v . So the distance from s to t in G' is exactly that from u to v .

To compute the graph G' , we need to extend the array of nodes with two new columns, and add t to the adjacency list of each $v \in V$, and create an adjacency list for s listing all the elements of S . All of these take $O(n)$ time. Then we can find the shortest path in G' from s to t in time $O((n' + m') \log n')$ using Dijkstra's algorithm, where n' is the number of nodes in G' and m' the number of edges in G' . Since $n' = n + 2 \in O(n)$. and $m' \leq m + 2n$, the total time is $O(n + (n + m) \log n) = O((n + m) \log n)$.