

- When asked to design an algorithm, an asymptotically inefficient solution gets very few or *even zero* points. For example, designing a $O(n^2)$ -time algorithm for a problem that also has a more efficient $O(n \lg n)$ -time algorithm does not get any points. Remember that the goal of this course is to design *efficient* algorithms so you can eventually write fast code.

General Graph Problems

Problem 1 (Transpose of a Directed Graph)

Exercise 22.1-3

Problem 2 (Copying a Graph)

Another Google interview question Assume that you have a graph G represented using adjacency list representation. Write the pseudocode of an efficient algorithm that makes a copy of the graph. It should build the adjacency list representation of another graph G' that has the same set of vertices and the same set of edges.

If you are not familiar with depth-first search, look it up in the textbooks.

Shortest Path-like Problems

Problem 3 (Fastest Airplane Trip)

A person would like to fly from city A to city B in the shortest possible time. He turns to a traveling agency that is willing to give all the departure and arrival times of all flights on the planet. Explain an algorithm that will allow the person to choose the shortest flight possible. The time of a travel is from departure to arrival at the destination, so it will include the time waiting for a connecting flight(s). [Hint: Don't think about modifying Dijkstra's algorithm. Instead modify your graph and/or data.]

Problem 4 (Faster Single-Source Shortest Path)

Assume that you have a graph $G = (V, E)$ whose all edges have a weight of 1. Describe an algorithm that runs in $O(|E| + |V|)$ time and computes the shortest distance from a source vertex s to all other vertices of G .

Problem 5 (Most Reliable Path)

Suppose you are given a graph of a computer network $G = (V, E)$ and a function $r(u, v)$ that gives a reliability value for every edge $(u, v) \in E$ such that $0 \leq r(u, v) \leq 1$. The reliability value gives the probability that the network connection corresponding to that edge will not fail. Explain and analyze the running time of an algorithm to find the most reliable path from a given source vertex s to a given target vertex t . The reliability of a path is the product of the reliabilities of its edges. Briefly explain why your algorithm is correct.

Minimum Spanning Tree (MST) Problems

Problem 6 (Non-comparison-based MST)

- a. Exercise 23.2-4
- b. Exercise 23.2-5

Problem 7 (Equal-Weight MST)

Assume you are given a weighted connected graph G . The weights of all edges of G are equal. We are interested in computing the MST of G .

- a. What is the running time of Kruskal's algorithm on G ?
- b. What is the running time of Prim's algorithm on G ?
- c. Write the pseudocode of a *custom* algorithm that runs in $O(|E| + |V|)$ time and computes the MST of G . The algorithm is *custom* in that it exploits the equal-weight property of G . A $O(m\alpha(n) + n)$ time algorithm is not acceptable for this section. [Hint: Since all edges have equal weights, it does not matter which edge you add to your MST as long as you do not make a cycle. Think of a simple way to test for existence of a cycle.]

Problem 8 (Bonus)

Suppose you are given n -points in the plane. We can define a complete graph on these points, where the weight of an edge $e = (u, v)$ is the Euclidean distance between u and v . We need to partition these points into k non-empty clusters, for some $n > k > 0$.

The property that this clustering should satisfy is that the minimum distance between any two clusters is maximized. The distance between two clusters A and B is defined to be $\min_{u \in A, v \in B} w(u, v)$, i.e., the minimum among the distances between all pairs of points, where one point is from cluster A and the other from cluster B .

Show that the connected components obtained after running Kruskal's algorithm till it finds all but the last $k - 1$ (most expensive) edges of MST produces an optimal clustering.

Dynamic Programming Problems

Problem 9 (Longest Common Subsequence)

Exercise 15.4-2

Problem 10 (Longest Increasing Subsequence)

- a. Exercise 15.4-5.
- b. Exercise 15.4-6

Problem 11 (Coin Changing)

Exercise 16-1, part (d).

