1A Lent Algorithms

Supervision 5: Graphs (Shortest Paths, Trees)

Hayk Saribekyan

February 22, 2019

When you are asked to describe an algorithm, also analyse its time and space complexity. Mention how you would implement the solution.

In this supervision, we will finish shortest-path algorithms on graphs and work on tree and DAG algorithms. This example sheet is a hybrid of lecturer's example sheets 5 and 6 and mine[1]. Some of the problems added by me are not directly useful for the exams or are out of the scope of the course. These are marked with *. These problems, however, show how some of the standard algorithms that you have learnt can be used. If you have time, you should work on them as they will develop your intuition.

I would rather see a failed attempt or ideas handed in, than a beautifully written correct solution of a problem that you found easy. I do not mind if you skip a problem and write "I know how to do it." So focus on the concepts that you have most trouble with rather than writing down solutions to problems that are easy for you.

1 Refresher

1. Explain how to modify BFS to find shortest paths from multiple sources $s_1, \ldots, s_k$. The goal of the algorithm is to find the length of the shortest path from all vertices v to their closest source s_i. The time complexity of BFS should not change.

   Can you do the same with Dijkstra's algorithm?

2. There are n cities in a country, which are connected by m train lines. The train between city a and b takes $w(a, b)$ time. To reduce operating costs, the government decides to remove as many lines as possible such that the shortest distance from the capital s to any other city remains the same[2]. Give an algorithm that decides which roads to remove.

2 Shortest Path Algorithms

Some problems are from `https://www.cl.cam.ac.uk/teaching/1819/Algorithms/ex5.pdf`

1. Problem 13 from the sheet 5.
2. Problem 14 from the sheet 5.

[1] Lecturer's sheets: `http://www.cl.cam.ac.uk/teaching/1819/Algorithms/materials.html`

[2] `https://www.youtube.com/watch?v=MCntdOZcXjA`

3. Problem 15 from the sheet 5.

4. Consider the problem of converting currencies modelled by a directed graph $G = (V, E)$ with $|V|$ vertices representing currencies and $|E|$ directed edges. Edge (u, v) has a strictly positive weight $w(u, v)$ representing the exchange rate. For instance, for any real number x, we have x USD $= w(\text{usd, gbp}) \cdot x$ GBP. Our goal is, given a pair of currencies $s, t \in V$, to find the least expensive way of exchanging from s to t, possibly by using more than one exchange.

   (a) How could you transform the graph by reweighting the edges so that the problem could be solved with a shortest path algorithm? Indicate which shortest path algorithm is used.

   (b) How would you deal with negative-weight cycles if they occurred in the transformed graph? Give the perspective of the currency trader as well as that of a computer scientist.

3 Trees, DAGs

Some problems are from `https://www.cl.cam.ac.uk/teaching/1718/Algorithms/ex6.pdf`

1. Problem 1 from sheet 6.

2. Problem 2 from sheet 6.

3. How would you find a *maximum* spanning tree of an edge-weighted graph?

4. Given an MST for an edge-weighted graph G and a new edge e. Describe how to find an MST of the new graph.

5. Describe an algorithm that finds all edges that are part of *some* MST. *Make it run in $O(|E| \log |V|)$ time.

   Hint: Find some MST T. What can you say about edges that are not in T?

6. Problem 4 from sheet 6.

7. Problem 6 from sheet 6.

8. Describe an algorithm that determines whether the topological ordering is unique.

9. * Describe an algorithm that finds the longest path in a DAG.

4 Implementation

Implementations are not part of the course but are critical to sharpen your understanding of graph algorithms. I would suggest that you work on these during the break if you do not have time now.

1. Find the MST: `http://www.spoj.com/problems/MST/`

2. Problem 1.3 above: `http://codeforces.com/problemset/problem/160/D`

3. Related to problem 3.9 above: `http://www.spoj.com/problems/BABTWR/`