CSCE 411: Design and Analysis of Algorithms

Lecture 19, Exam 2 Review

Date: April 1, 2025 *Nate Veldt, updated by Samson Zhou*

1 Test format and instructions

- Some multiple choice questions, some long answer questions
- You do not need to prove something unless you are explicitly asked to prove something
- You may be asked to explicitly prove something
- Scratch paper will be provided
- No calculators or electronic devices. You will not need them. If you have to do any computations, they will be basic enough to do by hand.
- You will put your name on the front page, and put your initials at the top of every other page.

A few other notes:

- If something was on a homework or in a lecture, it could be on the test.
- Some questions may involve a combination of a couple different things we've seen in different lectures.
- If we covered a proof of a result, you can reference it as fact without re-proving it, unless the question specifically is asking you to provide proof details.

2 Topics Covered on the Test

2.1 Summary of the summary

- Graph basics (adjacency matrix, adjacency list, directed/undirected, weighted/unweighted, cyclic/acyclic)
- (BFS) Breadth-first search
- (DFS) Depth-first search (applications: strongly connected components, topological sort)
- (MST) Minimum spanning tree (Kruskal's and Prim's algorithms)
- (SSSP) Single source shortest paths (Bellman-Ford, Dijkstra's)
- Maximum s-t flow and minimum s-t cut

For all of these graph algorithms, here's the short answer on what you should know:

- How is the problem defined? In particular, what type of graph does it apply to? Are there restrictions?
- What algorithms solve the problem, what is their basic mechanism, and their runtime?
- Don't just know how to eye-ball a solution for a small graph. You should know the mechanics of each algorithm well enough that you could apply it to a small graph, could write out pseudocode if need be, and provide a runtime analysis

2.2 Graph Basics

- Representing a graph as a list or a matrix
- Undirected and directed, weighted and unweighted (which of these can be viewed as special cases of others?)
- Strongly connected and weakly connected components. What is their relation? How are they defined?

2.3 Breadth first search specifics

- Running a BFS on a weighted graph just means ignore the edge weights. (You wouldn't say "BFS doesn't apply if the graph is weighted.")
- Remember what a breadth-first tree is and how to find it.
- Make sure you know the runtime

2.4 Depth-first search

- Know the different edge types in a depth-first search and what they mean
- Know the two main applications we discussed in class and how to use DFS for them.

2.5 Minimum Spanning Tree

- You should know the basic mechanism behind both Prim's and Kruskal's algorithm
- You should know what a cut is, and what a "safe" edge is.
- We did a mini runtime analysis for Prim's algorithm: you should know what the runtime is and how that basic argument worked (there are many other algorithms with similar types of runtime analyses).

2.6 Single source shortest paths algorithm

- Three algorithms we saw: Bellman-Ford, Dijkstra's (and we talked about two ways to implement the min-priority queue).
- You should understand the basic mechanism behind each, and different pros/cons of their runtimes

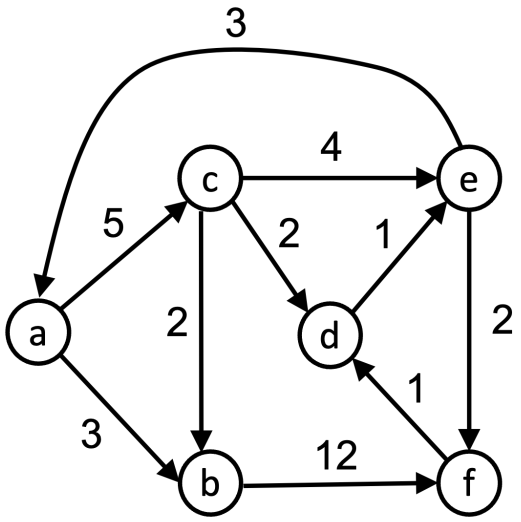
2.7 Min-cut Max-Flow Problem and algorithms

- What is the definition of an s-t flow function? What is an s-t cut in a directed graph?
- What is the basic idea behind the algorithms we have seen? Hint: it isn't that you repeatedly try to send flow from s to t as long as you can find unsaturated paths in the original graph G.

3 Practice Problems

1. When does breadth-first search solve the same problem as Dijkstra's algorithm?
2. What are the applications of a breadth first search? What problem(s) does it solve?
3. What are the two applications we saw for depth first search? Longer question: how does each one work?
4. Prove whether this is true or false: running a breadth first search in a directed graph returns the weakly connected components of that directed graph.

Figure 1: Sample graph 1



5. Consider the graph in Figure 2, but for this problem ignore the edge directions. Write down the minimum spanning tree that you would obtain from running Kruskal's algorithm, and the MST obtained by running Prim's algorithm starting from node 3.

6. Let $G = (V, E)$ be a directed graph. If you assume that $|E| = \Theta(V^2)$, what data structure should you use to implement the min-priority queue in Dijkstra's algorithm, and why?

7. What are the weakly connected and strongly connected components of sample graph 1?

8. Write down the adjacency matrix and the adjacency list for each of the sample graphs.

9. Draw the breadth-first tree of the sample graph 1 from starting node e

10. Find the minimum s-t cut of sample graph 1 when $s = a$ and $t = f$. Explain whether you get the same minimum s-t cut value when $s = f$ and $t = a$.

11. Write out a valid s-t flow for the graph in Figure 2.

12. Write out a topological ordering for each of the sample graphs or prove that it cannot exist.

13. Solve other BFS, DFS, MST, single source shortest path problems on the sample graphs (e.g., try a different source and/or sink node, change weights slightly, etc.)

Figure 2: Sample graph 2

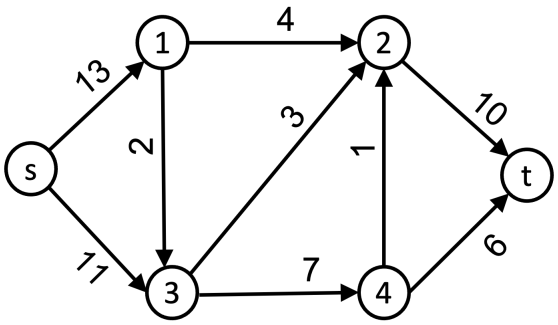


Figure 3: Sample graph 3