CS41 Lab 6: Shortest Paths, Midterm Review, MST Preview

In typical labs this semester, you'll be working on a number of problems in groups of 3-4 students. You will not be handing in solutions; the primary purpose of these labs is to have a low-pressure space to discuss algorithm design. However, it will be common to have some overlap between lab exercises and homework sets.

The learning goal of lab this week is to get some practice reasoning about Dijkstra's Algorithm and Shortest Paths with negative weights, and prepare for the midterm.

1. **Bellman-Ford:** In lab today, we will see how we can use the "update" procedure from Dijkstra's Algorithm to find shortest paths with negative edge weights.

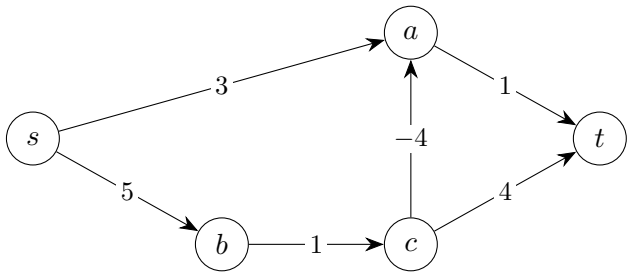
   Recall that Dijkstra's Algorithm makes use of the following UPDATE subroutine, run on every outgoing edge of a vertex when we add it to the "known region".

   UPDATE(u, v)
   1 $\texttt{dist}[v] = \min\{\, \texttt{dist}[v], \texttt{dist}[u] + \ell(u, v) \,\}$

   Notice that running UPDATE on an edge can never hurt: it will never update `dist` to a value smaller than the length of the shortest path (can you see why?).

   Before working on the rest of the problems, we will look at this together, and see how we can use this subroutine to design a new algorithm for shortest paths, called the Bellman-Ford algorithm.

   Now, consider the graph below.



   (a) Run Bellman-Ford on the graph, processing edges in the following order: (s, a), (s, b), (b, c), (c, a), (a, t), (c, t). Show the values of `dist` at the end of each round.

   (b) Run Bellman-Ford again, but this time processing edges in the opposite order from the previous part. Show the values of `dist` at the end of each round.

   (c) Does Dijkstra's Algorithm give the right answer on this graph?

   (d) Suppose we add an edge (a, b) with $\ell(a, b) = 2$. How does this affect the shortest paths in this graph? What does Bellman-Ford find? Can you find a way to modify Bellman-Ford to handle this scenario?

(e) What is the running time of Bellman-Ford? What about for the modified version you came up with in the previous part?

2. **All-pairs Shortest Paths.** Design and analyze an algorithm that, given a graph $G = (V, E)$ and edge weights $\ell(u, v)$ for all (u, v) inE, computes the (weighted) length of the shortest paths between every pair of vertices $u, v \in E$, or determines that no such path exists.

3. **More on DFS.** Recall the recursive DFS. Suppose we run DFS on a connected, undirected graph G starting from vertex u, and let T be the resulting DFS tree.

   (a) Prove that if (x, y) is *non-tree edge* (i.e., an edge in E but not in the DFS tree T), then one of x and y is an ancestor of the other in T. How is this different from BFS?

   (b) Suppose we run DFS on a graph G starting from a vertex s, and then run BFS on the same graph again starting from s, and the BFS and DFS trees are the same. Prove that G is a tree, specifically that $G = T$ (i.e., the DFS and BFS trees are the entire graph G).

4. **Hamiltonian Paths in DAGs.** A *Hamiltonian path* in a directed graph $G = (V, E)$ is a path that visits every vertex of G exactly once. It is believed that this problem is hard in general: there is no known efficient (polynomial-time) algorithm for it. We will revisit this idea later in the semester. However, it turns out that this problem can be solved efficiently when G is a DAG.

   Give an $O(m + n)$-time algorithm that, given a DAG G, determines whether G has a Hamiltonian path, and outputs one if it exists. Prove that your algorithm is correct.

   (*Hint:* if G has a Hamiltonian path $v_1, v_2, \ldots, v_n$, what can you say about the topological orderings of G?)

   (**Fun Note:** This problem is related to another problem from the midterm practice problems. Can you see which one?)

5. **(Preview) Connecting Buildings.** Imagine that Swarthmore wants to connect n buildings on campus with network cables. For each pair of buildings A, B, the college would incur a cost of $w(A, B) > 0$ to run a cable directly between them. Suppose we wish to spend as little money as possible while ensuring that there is some (possibly indirect) connection between every pair of buildings on campus.

   (a) Try to prove that the minimum-cost solution never contains a cycle.

   (b) See if you can devise a greedy rule for solving this problem. Then test it: can you find a graph where your greedy rule fails to find the optimal solution? If so, try to think of another rule.

6. Finally, feel free to take some time to work on problems from the practice problem set that was posted, or ask any content-related questions you might have.