CS 161 W22: Recitation 6 Problems

February 2022

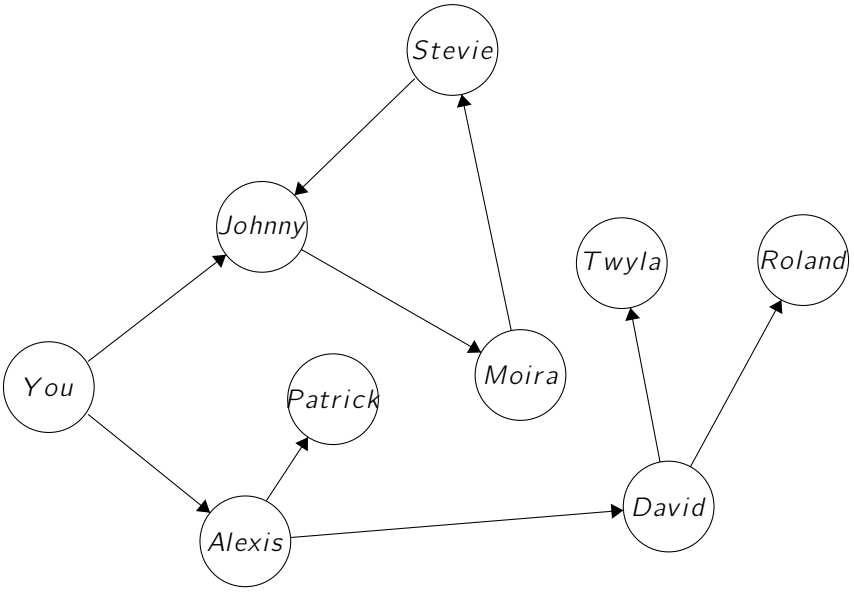
Exercise 0

You and your friends have great news - you've finished your CS 161 midterm! You'd like to celebrate with each of your friends individually, but you're not sure in what order you should visit them. To solve this problem, you construct a graph of your friend group, where each node represents the location of a friend and each edge represents a one-way bike path between two locations. Starting from the node labeled *You*, determine:

(a) The order in which your friends are visited by BFS

(b) The order in which your friends are visited by DFS, sorted by increasing DFS start time

(c) The order in which your friends are visited by DFS, sorted by increasing DFS finish time



Exercise 1

A review on hashing: Assuming you are using a uniformly random hash function,

1. if you were to hash n items into $b = n$ buckets, how many items would you expect to be hashed to the 3rd bucket? (Assume $n > 3$.)
2. in expectation, what is the number of items per bucket?
3. what is the probability that the 3rd bucket has exactly 3 items?
4. what is the big-O runtime of INSERT, DELETE, SEARCH, and SELECT where, for example, SELECT(5) finds the 5th smallest element in the hash table?

Exercise 2

Dijkstra's algorithm will in general fail on graphs with negative-weight edges. In this problem, we will explore an interesting special case with negative edges for which we can still solve the shortest paths problem as quickly as Dijkstra's algorithm.

Recall that any directed graph $G = (V, E)$ can be broken up into its *strongly connected components* (SCCs)—in other words, a partition of V into disjoint vertex sets C_i such that within each C_i, any two vertices have a path to one another.

Suppose that in addition to the graph $G = (V, E)$, we are given an *ordered* set of its SCCs, $V = C_1 \cup C_2 \cup \cdots \cup C_k$, with the promise that:

- Within each component C_i, all edges have nonnegative weight.
- Edges between different components may have negative weight. However, they all obey the following rule: for any edge (u, v) such that $u \in C_i$ and $v \in C_j$ for $i \neq j$, we are promised that $i < j$. In other words, an edge out of one component can only point to a component "to the right".

It is helpful to draw some examples of graphs that obey this structure.

Devise an algorithm that, given G, an ordered set $\{C_1, C_2, \ldots, C_k\}$ of its SCCs obeying the above constraints, and a starting node s, finds the shortest path lengths from s to all other nodes $v \in V$. It should have running time $O(|V| \log |V| + |E|)$.