CS310: Advanced Data Structures and Algorithms

Fall 2022 Assignment 3

Due: Monday, October 31, 2022 on Gradescope

Goal

Practice hash tables and graphs.

Questions

1. **Hashing:** Quadratic probing is an alternative way to resolve collisions. It is similar to linear probing in the sense that if a collision occurs, another index will be probed. but if, say, the key is hashed to index i, instead of probing $(i+1)\%m$, $(i+2)\%m$, $(i+3)\%m$, etc., the probing is done in quadratically larger jumps, so the probing chain in this case is $(i+1)\%m$, $(i+4)\%m$, $(i+9)\%m$, etc., and in general the k^{th} probe step is $(i+k^2)\%m$.

   Given a hash table of size 11, the key data is the following identifiers for some inventory: A29, C42, E12, D31, F08 and G34, B10. The hash function is H(LN) = ((L-'A') + N) % 11, where L is the letter, N is the number and 'A' is the ASCII value for A. For example, H("C29") = (2 + 29)% 11 = 9 since 'C' - 'A' = 2. For your convenience– the multiples of 11 are 11, 22, 33, 44, 55 etc. (such that the two digits are the same).

   (a) Draw the final configuration of the table after all the elements are inserted in the following illustration using linear probing. Do not rehash.

7	2	4	0	2	7	0				

   (b) Do the same for quadratic probing. Again, do not rehash.

7	2	2		4	7				0	0

2. **Graphs: K&T, Ch. 3 q.4:** inspired by the example of that great Cornellian, Vladimir Nabokov, some of your friends have become amateur lepidopterists (they study butterflies). Often when they return from a trip with specimens of butterflies, it is very difficult for them to tell how many distinct species they've caught – thanks to the fact that many species look very similar to one another.

   One day they return with n butterflies, and they believe that each belongs to one of two different species, which we'll call A and B for purposes of this discussion. They'd like to divide the n specimens into two groups–those that belong to A and those that belong to B – but it's very hard for them to directly label any one specimen. So they decide to adopt the following approach.

   For each pair of specimens i and j, they study them carefully side by side. If they're confident enough in their judgment, then they label the pair (i,j) either "same" (meaning they believe them both to come from the same species) or "different" (meaning they believe them to come from different species). They also have the option of rendering no judgment on a given pair, in which case we'll call the pair *ambiguous*.

So now they have the collection of n specimens, as well as a collection of m judgments (either "same" or "different") for the pairs that were not declared to be ambiguous. They'd like to know if this data is consistent with the idea that each butterfly is from one of species A or B. So more concretely, we'll declare the m judgments to be consistent if it is possible to label each specimen either A or B in such a way that for each pair (i,j) labeled "same," it is the case that i and j have the same label; and for each pair (i,j) labeled "different," it is the case that i and j have different labels. They're in the middle of tediously working out whether their judgments are consistent, when one of them realizes that you probably have an algorithm that would answer this question right away.

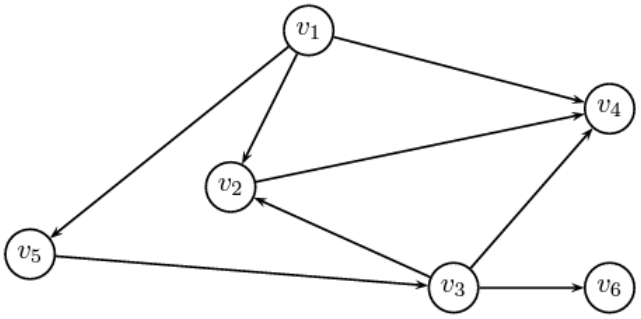
Give an algorithm with running time $O(m + n)$ that determines whether the m judgments are consistent. **Hint:** There is an underlying undirected graph problem here. Try to find out what are the vertices, what are the edges, and what algorithm(s) we showed in class can solve this problem.

3. **Graphs: K&T, Ch. 3 q.9:** There's a natural intuition that two nodes that are far apart in a communication network– separated by many hops– have a more "tenuous" connection than two nodes that are close together. There are a number of algorithmic results that are based to some extent on different ways of making this notion precise. Here's one that involves the susceptibility of paths to the deletion of nodes. Suppose that an n-node undirected graph $G = (V, E)$ contains two nodes s and t such that the distance between s and t is strictly greater than n/2.

   (a) Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all s-t paths. (In other words, the graph obtained from G by deleting v contains no path from s to t.) **Hint:** The proof is not very complicated. Just reason about the distance between s and t being greater than $n/2$, assume such a node v does not exist and prove by contradiction why it cannot be.

   (b) Give an algorithm with running time $O(m + n)$ to find such a node v. **Hint:** The algorithm involves some graph traversal.

4. **Directed Graphs:** Trace Depth-First search (DFS) on the following graph, starting from v_1. Mark all vertices and edges in order of visitation. To make the search fully specified, if two or more options exist, break tie by alphabetical order. Mark by * the edges that participate in the DFS tree.



5. **DAG:** A *Hamiltonian path* in a graph is a graph that visits every vertex exactly once.

   (a) Show that a DAG (directed acyclic graph) has a hamiltonian path if and only if it has just one topological ordering (you may assume the graph is connected). Remember this is an if and only if proof, so show both sides.

   (b) Give an $O(m + n)$ algorithm to find whether a DAG has a hamiltonian path.

6. **DAG:** A student suggested to topologically sort a DAG $G = (V, E)$ by first running BFS from a vertex s with an in-degree of 0, and then listing the vertices in the order of their distance from s. Show a simple example where this algorithm will not work. You may assume G is indeed a DAG, so no wise-assery.