CS 161 W19: Recitation 4 Problems

February 2019

Exercise 0

Recall from class that a universal hash family H is a family of functions from a universe U to $[1, \ldots, n]$ that satisfies

$$P_{h \in H}\left(h(u_i) = h(u_j)\right) \leq \frac{1}{n} \qquad \forall u_i, u_j \in U,\ u_i \neq u_j$$

where n is the number of buckets. Which of the following are universal hash families?

(a) Pick a prime $p \geq |U|$. $H = \{(ax + b \bmod p) \bmod n \mid a \in \{1, \ldots, p-1\}, b \in \{0, \ldots, p-1\}\}$.

(b) $H = \{x \bmod n\}$ (recall n is the number of buckets and is fixed).

(c) $H = \{ax \bmod n \mid a \in \{1, \ldots, n-1\}\}$.

(d) $H = \{ax + b \bmod n \mid a \in \{1, \ldots, n-1\}, b \in \{0, \ldots, n-1\}\}$.

Exercise 1

In section 2, we came up with a few different algorithms that take in n pairs of integers (x_i, y_i)—where for all i, j we have $x_i \neq x_j$ and $y_i \neq y_j$—and find the maximum cardinality subset of collinear points. Two points uniquely define a line, $y = mx + b$, with slope m and intercept b, and a set of points S is *collinear* if they all fall on the same line; that is, for all $(x_i, y_i) \in S$, $y_i = mx_i + b$.

In section 2, we asked you not to use hashing and you were able to arrive at a $O(n^2 \log n)$ time solution. Now, using hashing, come up with an $O(n^2)$ solution. Assume that given two points, you can compute the corresponding m and b for the line passing through them in constant time, and you can compare two slopes or two intercepts in constant time.

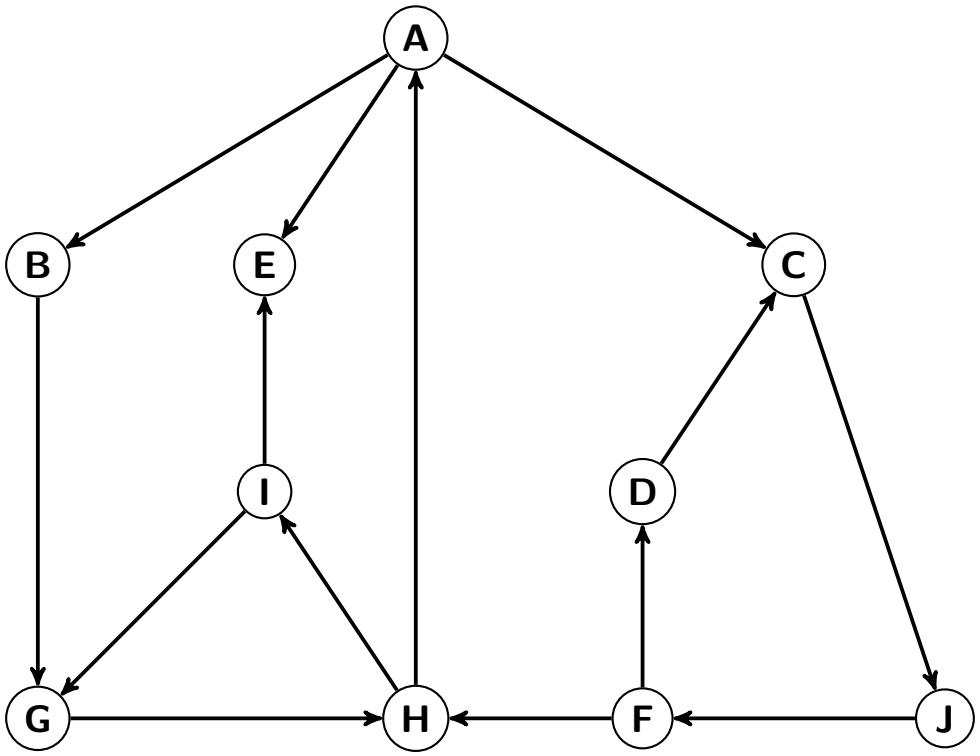
Exercise 2

Consider a hash table with n buckets and n elements. We assume that each element has a distinct key and they are hashed using a uniformly random hash function, i.e., each element has equal probability of hashing into any of the n buckets independently of other elements.

(a) For a particular bucket, what is the probability that the bucket has exactly one element? How about k elements?

(b) Let X_i be a random variable for the number of elements in bucket i. What is the expected value of X_i?

Exercise 3



(a) Perform DFS on the graph above starting from vertex A. Use lexicographical ordering to break vertex ties. As you go, label each node with the start time and the finish time. Highlight the edges in the tree generated from the search.

(b) Perform BFS on the graph above starting from vertex A. Use lexicographical ordering to break vertex ties. As you go, label each node with the discovery order. Highlight the edges in the tree generated from the search.

Exercise 4

A bipartite graph is an undirected graph whose vertices can be divided into two independent sets, U and V, such that every edge (u, v) connects one vertex in U and one vertex in V (or vice versa). Equivalently, a graph is bipartite if and only if the vertices can be colored using two colors such that no edge connects two vertices of the same color.

(a) Design an algorithm using BFS to determine whether or not a graph is bipartite.

(b) Design an algorithm using DFS to determine whether or not a graph is bipartite.

Exercise 5

Suppose you have an undirected, connected graph $G = (V, E)$ and two nodes $s, t \in V$. You then take a *random walk* on G: start at node $v_0 = s$, and at every step v_i, pick an edge coming out of v_i uniformly at random, and set v_{i+1} to be the other node on that edge.

Prove that with probability 1, your random walk will hit t: there exists some i such that $v_i = t$.