I think this problem set is more “interesting” than the previous two. As a result, 130 points will be available to you (plus the 2 points from the feedback form). However, the maximum score on this problem set will be 120. This means you will have the opportunity to earn 12 points of extra credit.

Have fun!

1 Graph Calisthenics (30 pts)

Consider a complete directed acyclic graph $G = (V, E)$ where $|V| = n$ and $|E| = \frac{n(n-1)}{2}$. Since the graph is complete, there exists an edge between each pair of the vertices $u, v \in V$ such that either $(u, v) \in E$ or $(v, u) \in E$ but not both (since the graph is acyclic).

Suppose we define a function contains($u, v$) which returns True if $(u, v) \in E$ and False if $(v, u) \in E$. Assume that we do not have access to the list of edges, and can only access the graph through the contains function. (We do have access to the list of vertices.)

(a) [10 pts] Give an asymptotically tight lower bound on the number of calls to contains($u, v$) required to find a topological sort of $G$.

(b) [10 pts] Prove that the bound is tight, i.e., that no algorithm can find a topological sort of $G$ with fewer than that many calls to contains($u, v$).

(c) [10 pts] Describe an algorithm that achieves this bound. No need for a correctness or runtime proof.

2 Simulating Fairness With Biased Coins (30 pts)

Suppose we are given a boolean function randP that returns true with probability $0 < p < 1$ and false with probability $1 - p$.

(a) [10 pts] (Algorithm) Describe an algorithm that returns true with probability $1/2$ and false with probability $1/2$. Your algorithm does not need
to terminate, but should, in expectation, use \( \frac{1}{p(1-p)} \) calls to \texttt{randP}. Your algorithm does not have access to the value of \( p \), and should not attempt to compute it. Your algorithm also does not have access to any sources of randomness other than \texttt{randP}.

(b) [15 pts] (Runtime) Formally prove that your algorithm runs using expected \( \frac{1}{p(1-p)} \) calls to \texttt{randP}.

(c) [5 pts] (Correctness) Informally argue that your algorithm returns \texttt{true} with probability 1/2 and \texttt{false} with probability 1/2.

3 Hashing Pages to Disk (35 pts)

In an application that requires a very large hashtable, it might be impractical to store the hash table in main memory. In particular, one might store the hash table on disk, with one disk page being one slot in the hash table.

Suppose that a disk page is large enough to hold many records. For example, a typical disk page holds \( 2^{12} \) bytes of data, and a record might contain only \( 2^{5} \) bytes. We will store the records that hash to a single page in a linear order on the page. If a page overflows because it contains too many records, the excess records are stored in an overflow area somewhere else on disk.

A search consists of hashing to the correct disk page and then linearly searching through the records on that page for one with the query key. If the page is full, then the overflow area must be searched in addition. Since the time to access a disk page is typically at least 10 ms, the cost of the linear search on the page is negligible. Thus, we shall focus the number of disk accesses as our cost measure. For the search operation, the cost is 1 if we find the record in its “proper” page, but it might be considerably greater if we must in addition search the overflow area. Consequently, the focus of this problem is to ensure that the proper pages seldom overflow, while using as little extra space as possible.

Suppose we are hashing \( n \) keys to disk pages, where each disk page holds up to \( r \) records. We want to know the number \( m \) of disk pages so that with high confidence, we can search for any of the \( n \) keys with a single disk access. Moreover, we want \( m = O(n/r) \) so that at most a constant fraction of the storage is wasted.

Assume uniform hashing.

(a) [10 pts] For some disk page \( k \) chosen arbitrarily, argue that the probability that page \( k \) overflows is at most \( \binom{n}{r} (1/m)^r \). Why is this bound an overestimate?

(b) [5 pts] Prove that the probability is at most \( m \binom{n}{r} (1/m)^r \) that any of the \( m \) pages overflow.

(c) [10 pts] Prove that the probability is at most \( m \left( \frac{en}{mr} \right)^r \) that any of the \( m \) pages overflow.
Show that by choosing \( m = 2en/r \), the probability is at most \( \frac{ne}{r^2} \) that any of the \( m \) pages overflow.

(e) [5 pts] Find the value of the expression \( \frac{ne}{r^2} \) when you are hashing \( n = 2^{20} = 1048576 \) records to \( m = 2en/r \) pages, where each page contains up to \( r = 2^7 = 128 \) records. Is overflow likely to be an issue in practice?

4 Exact Traversals (35 pts)

Let \( G \) be a directed acyclic graph. Let \( v_1, v_2, \ldots, v_n \) be a topological order on \( G \). An exact traversal is a path that touches all nodes of \( G \) exactly once.

(a) [30 pts] Formally prove that \( G \) has an exact traversal if and only if \( v_i \rightarrow v_{i+1} \) is an edge for all \( i = 1, 2, \ldots, n - 1 \).

(b) [5 pts] Describe an \( O(m + n) \) time algorithm that determines whether an exact traversal exists. No need for a correctness or runtime proof.

5 Feedback form (2 pts extra credit)

Don’t forget to fill out the feedback form, at http://goo.gl/forms/xeeblZ7vYxG2yHOD2

Thank you for helping to make CS 161 a better class!