CS 161 (Stanford, Fall 2025)
Optional Extra Section 4 Problems

1 Red-Black Trees

1.1

If the length of a path from the root of a red-black tree to one of the leaf NIL nodes is 50, what could be the length of another path from the root to some other NIL node?

2 Binary Search Trees

2.1 True or False

Which of the following statements are true?

(a) The height of a binary search tree with n nodes cannot be smaller than $\Theta(\log n)$.

(b) All the operations supported by a binary search tree (except OutputSorted) run in $O(\log n)$ time.

3 Hashing

3.1 Hash Tables Gone Crazy

In this problem, we will explore *linear probing*. Suppose we have a hash table H with n buckets, universe $U = \{1, 2, \ldots, n\}$, and a *uniformly random* hash functions $h : U \to \{1, 2, \ldots, n\}$.

When an element u arrives, we first try to insert into bucket $h(u)$. If this bucket is occupied, we try to insert into $h(u) + 1$, then $h(u) + 2$, and so on (wrapping around after n). If all buckets are occupied, output **Fail** and don't add u anywhere. If we ever find u while doing linear probing, do nothing.

Throughout, suppose that there are $m \leq n$ distinct elements from U being inserted into H. Furthermore, assume that h is chosen *after* all m elements are chosen (that is, an adversary cannot use h to construct their sequence of inserts).

1. (Warmup) Can we ever output **Fail** while inserting these m elements?

2. Above, we gave an informal algorithm for inserting an element u. Your next task is to give algorithms for searching and deleting an element u from the table.

   Hint: Be careful that the search and delete algorithm work together!

3. In this part, we will analyze the expected runtime of linear probing assuming that $m = n^{1/3}$ and that no deletions occur.

   (a) Give an upper bound on the probability that $h(u) = h(v)$ for some u, v that are a part of these first m elements, assuming that $m = n^{1/3}$.

   Hint: You may need that $\mathbf{P}[\textit{at least one of } E_1, \ldots, E_k \textit{ happens}] \leq \sum_{i \in [k]} \mathbf{P}[E_i]$ *given any random events* $E_1, \ldots, E_k$.

   (b) When inserting an element, define the number of *probes* it does as the number of buckets it has to check, including the first empty bucket it looks at. For example, if $h(u), \ldots, h(u) + 10$ were occupied but $h(u) + 11$ was not then we would have to check 12 buckets.

   Prove that the expected number of total probes done when inserting $m = n^{1/3}$ elements is $O(m)$.

4 Graphs, DFS, BFS

4.1 True or False

1. If (u, v) is an edge in an undirected graph and during DFS, vertex v is completely explored before vertex u, then u is an ancestor of v in the DFS tree.

2. In a directed graph, if there is a path from u to v and DFS visits u before visiting v, then u is an ancestor of v in the DFS tree.

4.2 Bipartite Graphs

A Bipartite Graph is a graph whose vertices can be divided into two independent sets, U and V such that every edge (u, v) connects a vertex from U to V or a vertex from V to U. A bipartite graph is possible if the graph coloring is possible using two colors such that vertices in a set are colored with the same color. In lecture, we saw an algorithm using BFS to determine where a graph is bipartite. Design an algorithm using DFS to determine whether or not an undirected graph is bipartite.