1. (Fun with red-black trees) (Difficulty: Easy)

Recall the five rules for red-black trees:

- Every vertex is colored red or black.
- The root vertex is a black vertex.
- A NIL child is a black vertex.
- The child of a red vertex must be a black vertex.
- For all vertices $v$, all paths from $v$ to its NIL descendants have the same number of black vertices.

Consider the following binary search trees. Which of the following vertices must be black in any valid red-black coloring?

(a) ![Diagram of a binary search tree]  
(b) ![Diagram of a binary search tree]
2. (Randomly built BST’s) (Difficulty: Hard)

Here, we prove that the average depth of a vertex in a randomly built binary search tree with \(n\) vertices is \(O(\log(n))\). A **randomly built binary search tree** with \(n\) vertices is one that arises from inserting the \(n\) keys in random order into an initially empty tree, where each of the \(n!\) permutations of the input keys is equally likely.

Let \(d(x, T)\) be the depth of vertex \(x\) in a binary tree \(T\) (the depth of the root is 0). Then, the average depth of a vertex in a binary tree \(T\) with \(n\) vertices is

\[
\frac{1}{n} \sum_{x \in T} d(x, T)
\]

\(\)

(a) Let the **total path length** \(P(T)\) of a binary tree \(T\) be defined as the sum of the depths of all vertices in \(T\), so the average depth of a vertex in \(T\) with \(n\) vertices is equal to \(\frac{1}{n}P(T)\). Show that \(P(T) = P(T_L) + P(T_R) + n - 1\), where \(T_L\) and \(T_R\) are the left and right subtrees of \(T\), respectively. [We are expecting: A rigorous mathematical proof.]
(b) Let $P(n)$ be the expected total path length of a randomly built binary search tree with $n$ vertices. Show that

$$P(n) = \frac{1}{n} \sum_{i=0}^{n-1} (P(i) + P(n - i - 1) + n - 1).$$

[We are expecting: A rigorous mathematical proof.]

(c) Show that $P(n) = O(n \log(n))$. Hint: You might want to revisit quicksort.

[We are expecting: A short English justification.]

(d) Design a sorting algorithm based on randomly building a binary search tree. Show that its expected runtime is $O(n \log(n))$. Assume that a random permutation of $n$ keys can be generated in $O(n)$-time.

[We are expecting: An English description of the algorithm and a brief justification for its runtime.]
Graph Algorithms

1. (Fun with graphs) (Difficulty: Easy)

For each of the following graphs, assuming a vertex’s neighbors will be visited in alphabetical order, what will be the ordering of end times from lowest to highest for \texttt{dfs} and order of visited times for \texttt{bfs}, starting from \texttt{A}.

(a)
\begin{tikzpicture}

\node (A) at (0,0) {A};
\node (B) at (2,0) {B};
\node (C) at (0,-2) {C};
\node (D) at (2,-2) {D};
\node (E) at (1,-4) {E};
\draw (A) -- (B);
\draw (A) -- (C);
\draw (A) -- (D);
\draw (B) -- (E);
\draw (C) -- (D);
\draw (D) -- (E);
\end{tikzpicture}

(b)
\begin{tikzpicture}

\node (A) at (0,0) {A};
\node (B) at (2,0) {B};
\node (C) at (0,-2) {C};
\node (D) at (2,-2) {D};
\node (E) at (1,-4) {E};
\draw (A) -- (B);
\draw (A) -- (C);
\draw (B) -- (D);
\draw (C) -- (D);
\draw (D) -- (E);
\end{tikzpicture}

2. (DFS) (Difficulty: Easy)

Implement \texttt{dfs} without using recursion, still with runtime $O(|V| + |E|)$.

[We are expecting: An pseudocode of the algorithm and a brief justification for its runtime.]
Consider a robot that can take steps of $k$ distinct lengths $s_1, s_2, \ldots, s_k \in \mathbb{Z}$ in feet. The robot starts at position $p_0$ on a circular track with circumference length $T \in \mathbb{N}$ in feet and it starts walking counterclockwise around the track to calibrate itself. It takes one step for each of the lengths $s_1, s_2, \ldots, s_k$, bringing it to position $p_1$.

Describe an $O(T + kT)$-time algorithm that finds the fewest number of steps that the robot needs to take to get back to position $p_0$, assuming the robot can’t turn around (all steps must be taken in the counter-clockwise direction).

[We are expecting: An English description of the algorithm and a brief justification for its runtime.]
4. (2SAT-SCC) (Difficulty: Hard)

In 2SAT, you have a set of boolean clauses, each containing two variables. Your goal is to set all of these clauses to true, or report that doing so is not possible. For example,

$$(x_1 \lor x_2) \land (\bar{x}_1 \lor x_3) \land (\bar{x}_2 \lor \bar{x}_3)$$

can be satisfied by setting $x_1$ and $x_3$ to true and $x_2$ to false.

(a) First, it helps to decompose clauses into their implications i.e. a clause represented by $a \lor b$ would be represented by $\bar{a} \rightarrow b$ and $\bar{b} \rightarrow a$. What are the implications in the clauses above?

[We are expecting: A list of implications.]

(b) To solve this problem with a graph algorithm, first built a graph by creating two vertices for each variable $x$: $x$ and $\bar{x}$. Then, add directed edges based in the implications from each clause. What is the graphical representation of the above boolean clauses?

[We are expecting: A directed graph.]

(c) What do SCC’s in these 2SAT graphs represent? What if $x$ and $\bar{x}$ were part of the same SCC?

[We are expecting: Answers to the questions.]
(d) Assume we have a sink SCC $C$ from the graph, and that no variable and its negation both appear inside of $C$. If $C$ has vertices of the form $a, b, \ldots, z$, what corresponding component do we know must exist in the graph? Is there an edge between these two components?

[We are expecting: Answers to the questions.]

(e) Given this graph and the properties above, develop an algorithm to solve 2SAT.

[We are expecting: An English description of the algorithm.]