1. **Space of Van Emde Boas Trees**

In class we saw the Van Emde Boas (VEB) Tree data structure and showed it supports all ordered dictionary operations in $O(\log \log U)$ time if all elements are integers in the range \{0, 1, 2, \ldots, U - 1\}. In this question we will analyze (and improve) its space requirement.

   a) Write a recurrence relation for $S(U)$, the space usage of a VEB tree with elements in the range \{0, 1, 2, \ldots, U - 1\}, and prove that $S(U) = \Theta(U)$.

   b) Note that in the worst case $U$ could be significantly larger than $n$, so this data structure’s space complexity is not linear (or even near linear) in the number of elements stored in it. We will address this issue here.

   Using dynamically resizing hash tables, which support insert/delete/find operations in $O(1)$ expected amortized time while only using $O(m)$ space when storing $m$ elements, show how to decrease the space usage to $O(n \log \log U)$ at the cost of increasing the operations of the VEB tree to $O(\log \log U)$ expected amortized.

   **Direction:** Outline only the changes made to the VEB tree’s operations and running time, and give a brief explanation as to why each of the $n$ elements accounts for $O(\log \log U)$ space of the data structure.
2. **Long Path or Large Independent Set**

Here we will see an application of DFS to approximating one of two problems in an undirected graph $G(V, E)$ – longest path and shortest independent set. (An *independent set* is a set of vertices $U \subseteq V$ with no edge in $E$ connecting any two vertices in $U$.)

a) Prove that if $(u,v) \in E$, then in any DFS tree of $G$, either $u$ is an ancestor of $v$ or vice versa.

b) Give an $O(n + m)$-time algorithm which either computes a simple path of length $\Omega(\sqrt{n})$ or an independent set of cardinality $\Omega(\sqrt{n})$.

**Context:** Both *longest path* and *independent set* are NP-hard problems, and it is even NP-hard to approximate them within any $n^{1-\epsilon}$ factor, for any constant $\epsilon > 0$. Nonetheless, this exercise shows that for any graph you can approximate (at least) one of these problems to within an $O(n^{1/2})$ factor, in linear time.

3. **Efficiently Lighting the Tree**

Given a rooted tree $T$ on $n$ vertices with light bulbs on each vertex, we would like a data structure supporting the following operations.

1. **Init($T$)** - initialize the tree. All the vertices are unlit.
2. **TurnOff($v$)** - given a pointer to the vertex $v$, turn $v$ off (if it was lit).
3. **TurnOn($v$)** - given a pointer to the vertex $v$, turn $v$ on if $v$ was off AND none of $v$’s descendants in $T$ is lit. (If a descendant of $v$ is lit, don’t turn $v$ on – output nil.)

Answer the following.

a) Give an $O(n)$ time algorithm which computes two functions $\ell : V \rightarrow [O(n)]$ and $r : V \rightarrow [O(n)]$ satisfying
   1. $\ell(v) < r(v)$ for all $v \in V$.
   2. $\ell(u) \neq \ell(v)$ and $r(u) \neq r(v)$ if $u \neq v$, and
   3. $u$ is a descendant of $v$ in $T$ if and only if $[\ell(u), r(u)] \subseteq [\ell(v), r(v)]$.

b) Using the encoding from part (a), show how to maintain the above data structure. Running time requirements: **Init** should take $O(n)$ time. **TurnOn/TurnOff** should both take $O(\log \log n)$ time. (Partial credit for $O(\log n)$ time.)

c) **(Bonus)** solve (b), but this time a vertex $v$ can be turned on if none of its descendants or ancestors in $T$ is lit.

**Note:** Partial credit will be given for a solution for (b) assuming an $O(n)$-time computation of an encoding as in part (a) as a black box.

**Note:** The tree $T$ may have arbitrary height, so a solution supporting TurnOn/TurnOff in $O(h)$ time, where $h$ is the height of $T$, will receive no credit.
4. **Saving Energy in Radio Broadcast**

Two radio towers with identifiers in the range \(\{1, 2, \ldots, n\}\) are given. Each tower knows its ID, but not that of the other tower. The towers would like to discover each other’s ID. These radio towers communicate over a channel, during discrete time steps. At each time step a tower may take one of three actions: transmit, listen, or be idle. Transmitting and listening require one unit of energy, whereas idleness is free. If one tower listens and the other transmits a message in the same step, then the listening tower learns the content of the transmitted message. If both transmit at the same time, then they hear noise, and cannot decipher the messages sent. A tower that listens when the other tower does not transmit hears nothing (not even noise). An idle tower hears nothing. The towers’ goal is to use as little energy as possible to learn each other’s IDs.

a) Give a protocol using \(O(1)\) energy to determine if the towers have the same ID.

A communication protocol using \(w\) energy and resulting in either of the towers learning the other tower’s ID implies the existence of a protocol using \(w + 1\) energy and resulting in both towers learning the other tower’s ID. We will therefore focus on minimizing the energy of a protocol which allows either one of the towers to learn the other tower’s ID.

b) (Warm up 1) Give a randomized protocol which requires \(O(1)\) energy in expectation.

c) (Warm up 2) Give a deterministic protocol which requires \(O(\log n)\) energy.

d) Give a deterministic protocol which requires \(O(\log \log n)\) energy.

5. **Prize Collecting**

You are given a directed graph \(G = (V, E)\) with positive values (prizes) \(p : V \to \mathbb{R}^+\) on its nodes. Our goal is to find a (not necessarily simple) path in the graph which collects the highest overall prize. That is, a path which visits a subset of vertices \(U \subseteq V\) which maximizes \(\sum_{u \in U} p(u)\). Give a linear-time algorithm to compute such a path, if

a) \(G\) is a DAG (directed acyclic graph).

b) \(G\) is a general directed graph.

6. **Alice and Bob and Numbers on a Line**

Alice and Bob are playing the following game: there are \(n\) numbers on a line, \(a_1, a_2, \ldots, a_n\). Alice and Bob take alternating turns picking one of the extreme (leftmost or rightmost) remaining numbers on the line. Alice’s goal is to maximize the sum of numbers that she picked minus the sum of numbers that Bob picked, while Bob’s objective is the exact opposite. For example, if the initial numbers were 3, 8, 1 then Alice will have gain \(-4\), regardless of her first choice, as Bob will pick 8 as his first (and only) number. Our goal will be to design an algorithm which computes the maximum gain Alice can guarantee herself. That is, the highest gain she can guarantee against any strategy Bob plays.

a) Show that greedily picking the largest extreme number is not an optimal strategy.

b) Given an \(O(n^2)\)-time algorithm to compute Alice’s maximum gain.
(No credit): Prove: if $n$ is even, then Alice’s gain is nonnegative (for any $a_1, a_2, \ldots, a_n$).