Starting with this homework, I'll assign one or two problems a week, due approximately two weeks later. If I assign two problems in one week, your team should submit a solution for only one of them.

---

5. In this problem, I'll ask you to fill in the final remaining details of Pătrașcu and Demaine's lower bound proof for dynamic connectivity. Let $U = [u] = \{12, \ldots, u\}$. Our goal is to prove that we can specify a separator for any two k-element subsets of U using only $O(k + \log\log u)$ bits.

   (a) Let A and B be arbitrary subsets of U, where $|A| = |B| = k$. Consider an ideal random hash function $h\colon U \to \{0, 1\}$. What is the probability that $h(a) = 0$ for all $a \in A$ and $h(b) = 1$ for all $b \in B$? (Your answer should be a function of k and u.)

   (b) A set of H of hash functions ***separates*** A and B if there is a function $h \in H$ such that $h(a) = 0$ for all $a \in A$ and $h(b) = 1$ for all $b \in B$. Suppose H is a set of N independent ideal random hash functions. What is the probability that H separates A and B? (Your answer should be a function of k, u, and N.)

   (c) Now call a set H of hash functions a ***k-separator*** if it separates every pair of disjoint k-element subsets of U. Suppose H is a set of N independent ideal random hash functions. Derive a good upper bound on the probability H is a k-separator. (Your answer should be a function of k, u, and N.)

   (d) Derive a good lower bound on the smallest value of N such that the previous probability is positive. (Your answer should be a function of k and u.)

   (e) Conclude that a k-separator of size N exists. It follows that for any two disjoint k-element subsets A and B, we can specify a separator for A and B using only $\lg N = O(k + \log\log u)$ bits.

6. A ***mergeable dictionary*** stores a collection $\mathcal{S}$ of disjoint sets that cover some totally-ordered universe U of size n and supports the following operations:

   - $\text{Find}(x)$: Find the set $S \in \mathcal{S}$ such that $x \in S$.
   - $\text{Succ}(S, x)$: Return the smallest element $y \in S$ such that $y > x$. If $x > \max S$, then return ∞.
   - $\text{Split}(S, x)$: Remove set S from the collection $\mathcal{S}$ and insert the subsets $\{w \in S \mid w \leq x\}$ and $\{y \in S \mid y > x\}$.
   - $\text{Merge}(R, B)$: Remove sets R and B from the collection $\mathcal{S}$ and insert their union $R \cup B$.

   The first three operations can be implemented in $O(\log n)$ amortized time using splay trees, or even in $O(\log n)$ worst-case time using red-black trees. We can suppose Merge in the same time bounds if we are guaranteed that $\max R < \min B$. Your task is to add support for the Merge operation for *arbitrary* pairs of disjoint sets, so that Split and Merge run in $O(\log^2 n)$ amortized time, following Iacono and Özkan*.[1]

   To support the analysis, we first need to establish some notation. Fix two disjoint sets R and B, and assume $\min(R \cup B) \in R$ without loss of generality. We can decompose the union $R \cup B$ into a sequence of alternating intervals

$$R \cup B = R_0 \cup B_1 \cup R_2 \cup B_3 \cup R_4 \cup \cdots$$

   where $R_i \subseteq R$ and $\max R_i < \min B_{i+1}$ for every even index i, and $B_i \subseteq B$ and $\max B_i < \min R_{i+1}$ for every odd index i. Let $\boldsymbol{I(R, B)}$ denote the number of intervals in this decomposition.

   Now for any element x and any set $S \in \mathcal{S}$, we define three functions:

   - $\boldsymbol{rank(x)}$ is the number of elements in U that are smaller than x. In particular, $rank(\infty) = n$.
   - $\boldsymbol{gap(S, x)} := rank(\text{Succ}(S, x)) - rank(x)$.

   Finally, define the potential of the collection $\mathcal{S}$ as follows:

$$\Phi(\mathcal{S}) := \sum_{S \in \mathcal{S}} \sum_{x \in S} \lg(gap(S, x)).$$

   (a) Suppose R and B are disjoint sets, each stored in its own splay tree. Describe an algorithm that executes $\text{Merge}(R, B)$ in $O(I(R, B) \log n)$ amortized time. *[Hint: Start by calling* $\textit{Search}(R, \min B)$*.]*

   (b) Prove that calling $\text{Split}(S, x)$ increases Φ by at most $\lg n$. *[Hint: This is just an exercise in definition-chasing.]*

   (c) Prove that calling $\text{Merge}(A, B)$ decreases Φ by at least $\alpha \cdot I(A, B) - \beta \cdot \log n$, for some constants α and β.

   (d) Conclude that your algorithm for $\text{Merge}(R, B)$ in part (a) runs in $O(\log^2 n)$ amortized time.

   ★(e) Prove or disprove: The standard algorithm for Split and the algorithm for Merge in part (a) actually use only $O(\log n)$ amortized time.

[1] John Iacono and Özgür Özkan*. Mergeable dictionaries. *Proc. 37th ICALP*, 164-175. Lecture Notes Comp. Sci. 6198, Springer, 2010.