CS 598TMC, Fall 2023
Timothy Chan

Homework 4 (due Dec 4 Monday 10am)

Instructions: see previous homework. This homework is shorter and is worth half the weight.

1. [*22 pts*] We are given an undirected graph $G = (V, E)$ with n vertices and m edges, where each edge e has a *time interval* $[a(e), b(e)]$ indicating when e is "alive". Assume that $a(e), b(e) \in \{1, \ldots, 2m\}$ and the numbers are all distinct. We want to determine all time values $t \in \{1, \ldots, 2m\}$ for which the subgraph $G_t = (V, \{e \in E : t \in [a(e), b(e)]\})$ (consisting of all edges that are alive at time t) is connected.

   (a) [*5 pts*] Using a data structure from class, show that the problem can be solved in $O(m \log^2 n)$ time.

   (b) [*17 pts*] Next, describe a direct algorithm that solves the problem in $O(m \log n)$ time.

   Hint: use divide-and-conquer on the time intervals, similar to segment trees. If an edge $e = uv$'s time interval $[a(e), b(e)]$ is "long", we can *contract* e, i.e., collapse its two vertices u and v into a single vertex...

2. [*28 pts*] We want to maintain a dynamic set S of n intervals, subject to insertions and deletions, so that we can quickly compute the value $c(S)$, defined as the minimum number of intervals of S that cover $[0, 1]$.

   (For example, for $S = \{[-0.3, 0.3], [-0.2, 0.05], [0.2, 0.4], [0.1, 0.6], [0.5, 0.9], [0.7, 1.1], [0.8, 0.95]\}$, we have $c(S) = 4$ since the 4 intervals $[-0.3, 0.3], [0.1, 0.6], [0.5, 0.9], [0.7, 1.1]$ cover $[0, 1]$. You may assume that all endpoints are distinct.)

   Define $\text{succ}_S([a, b])$ to be the interval $[a', b'] \in S$ that satisfies $a' \le b \le b'$ while maximizing b'. It is known that the static problem can be solved by following greedy algorithm: let $[a_0, b_0] = [0, 0]$, and $[a_i, b_i] = \text{succ}_S([a_{i-1}, b_{i-1}])$ for $i = 1, \ldots, \ell$, till $a_\ell \le 1 \le b_\ell$. Then $c(S) = \ell$. (You do not need to prove correctness of this greedy algorithm, though the proof is not difficult.)

   (a) [*4 pts*] First, show that for a static set P of n intervals, there is a data structure with $\widetilde{O}(n)$ preprocessing time and space, so that given any query interval $[a, b]$ (not necessarily in P), we can compute $\text{succ}_P([a, b])$ in $\widetilde{O}(1)$ time. The $\widetilde{O}$ notation hides polylogarithmic (i.e., $\log^{O(1)} n$) factors.

   Hint: 1D range max.

   (b) [*16 pts*] For a static set P of n intervals, give a data structure with $\widetilde{O}(n)$ preprocessing time and space, so that for any additional query set Q of q intervals, we can compute $c(P \cup Q)$ in $\widetilde{O}(q)$ time.

Note: you may use the following fact: given any tree with n nodes (not necessarily balanced), we can build a static data structure with $O(n)$ preprocessing time and space, so that given any node v and integer i, we can find the ancestor of v at level i (if exists) in $O(1)$ time (this is known as a *level ancestor query*).

Hint: consider the tree/forest $T = \{(I, \mathrm{succ}_P(I)) : I \in P\}$.

(c) [*8 pts*] Now, consider a dynamic setting where the update sequence satisfies a *first-in first-out (FIFO)* assumption: namely, whenever I is inserted before I', we are promised that I must be deleted before I'.

Using (a) and (b), give a dynamic data structure that supports insertions and deletions in S in $\widetilde{O}(\sqrt{n})$ amortized time and can compute $c(S)$ in $\widetilde{O}(\sqrt{n})$ time, assuming FIFO updates.

Hint: do periodic rebuilding after every q updates.

Bonus [up to 5 pts]: obtain the same result in the general dynamic setting without the FIFO assumption (this might require a different approach).