PRINCETON UNIV SPRING '26 COS 423: THEORY OF ALGORITHMS

Homework 4

Out: *Mar 24*

Problems:

§1 Redo the amortized analysis for y-fast trie with both insertion and deletion, and prove its amortized time is $O(\log \log U)$. Your argument should prove that any sequence of $O(m)$ operations consisting of insertions and deletions takes $O(m \log \log U)$ time. You may assume there is no query for simplicity.

§2 (a) Suppose we modify the Fibonacci heap such that each node maintains a counter of how many branches it has lost. Instead of cutting a node when it loses the second branch, we do this when this counter reaches four. Does this version have the same amortized time? Prove it.

(b) Suppose we never cut a node from its parent after losing any number of branches. Does this version have the same amortized time? Prove it.

§3 Given a sequence of $O(n)$ *heap* operations, i.e., Insert(x), and ExtractMin(), compute the output for each ExtractMin. Suppose all Insert(x) have *distinct* x, and x is in $\{1, \ldots, n\}$. Note in this problem, we are given the sequence in advance, and can process the whole sequence of operations before making an output. Below, we will show how to compute all ExtractMin in a total of $O(n\alpha(n))$ time, for the inverse-Ackmann function $\alpha(n)$.

Suppose there are m ExtractMin operations, we first process the sequence to label ExtractMin by $\{1, \ldots, m\}$. Consider each chunk of Insert between two adjacent ExtractMin, which may be empty. Let I_k be the chunk of Insert between $(k-1)$-th and k-th ExtractMin.

(a) Prove that if Insert(1) is in I_k, then the output of k-th ExtractMin is 1. Then we "remove" this ExtractMin by *merging* I_k into I_{k+1}, prove that if then Insert(2) is in $I_{k'}$, the output of k'-th ExtractMin is 2.

(b) Show how to compute all ExtractMin in $O(n\alpha(n))$ time using Union-Find data structures.

§4 Consider Union-Find data structure with only path compression.

(a) Suppose all items $\{1, \ldots, n\}$ have been merged into one set, forming an arbitrary tree. Prove that if we now run Find(x) for all x in $\{1, \ldots, n\}$ once in any order, these n Find operations take a total of $O(n)$ time.

In the following problems, we construct a bad sequence of $O(n)$ operations that makes path compression to take $\Omega(n \log n)$ time. Let *binomial trees* B_k be the following: B_0 is a single node; B_k is formed by connecting the root of one B_{k-1}

to the root another B_{k-1}, the two B_{k-1} have different nodes. Note that the two copies of B_{k-1} differ by one level, the root of B_k is the root of one B_{k-1}, and the root of the other B_{k-1} is its child.

(b) [optional] Prove that the depth of B_k is k, and the size of B_k is 2^k.

(c) [optional] Suppose we have constructed a binomial tree B_k with root node r. Now we Union r and a single node p such that p becomes the new root, then run Find on the deepest node in the tree. Prove that this operation takes $\Omega(k)$ time, and the new tree after path compression is B_k with one node under the root.

(d) [optional] Show that there is a sequence of $O(n)$ operations from all singleton sets, such that the Union-Find data structure with only path compression takes $\Omega(n \log n)$ time.