ICS 621: Analysis of Algorithms Fall 2019

Problem Set 2

Prof. Nodari Sitchinava *Due: Friday, September 20, 2019 at the* **start** *of class*

You may discuss the problems with your classmates, however **you must write up the solutions on your own** and **list the names** of every person with whom you discussed each problem.

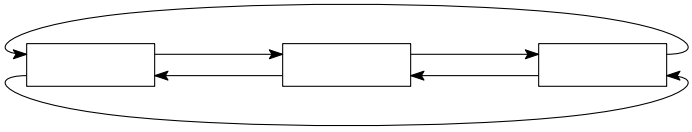
1 Lazy Binomial Heaps (100 pts)

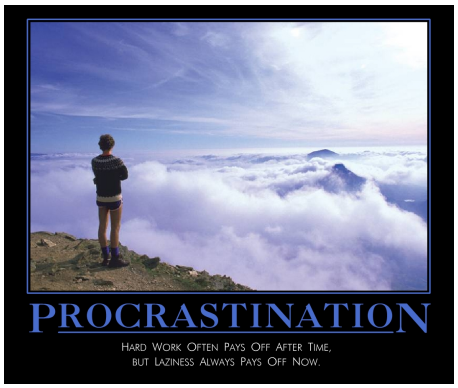
Being lazy is nothing to be proud of, but occasionally it can improve the overall execution.

In this exercise, we will see how delaying the work of the UNION operation in the binomial heap can improve the runtime of several operations of the binomial heap.

DELETE and DECREASE-KEY operations in the lazy binomial heap will remain unchanged.

To implement the new UNION we will store the list of roots as a *circular* doubly-linked list. That is, *every* node in the linked list has two pointers $left$ and $right$, including the $head$ and the $tail$ of the list, such that $head.left = tail$ and $tail.right = head$:





We will also maintain the number of elements stored in the heap as a field $Q.n$ (accessible in $O(1)$ time).

Consider the following description of UNION(Q_1, Q_2). It simply appends the root list of Q_2 to the right of the root list of Q_1:

```
function UNION(Q1, Q2)
    if Q1.head = NIL
        return Q2
    else
        if Q2.head ≠ NIL
            tail1 ← Q1.head.left; tail2 ← Q2.head.left      ▷ Pointers to the tails of Q1 and Q2
            Q2.head.left ← tail1; tail1.right ← Q2.head     ▷ Link head of Q2 to the right of the tail of Q1
            Q1.head.left ← tail2 ; tail2.right ← Q1.head    ▷ Link the tail of Q2 and the head of Q1
            Q1.n ← Q1.n + Q2.n                              ▷ Update the heap size
                                                            ▷ Update the minimum
            if Q2.min.value < Q1.min.value                  ▷ Remember: Q.min is a pointer to a node
                Q1.min ← Q2.min
        return Q1
```

Observe that this implementation of UNION takes only $O(1)$ time in the worst case. However, it no longer preserves the invariant that the list is ordered by the sizes of trees. Moreover, it is also no longer true that the binomial heap will have at most one binomial tree of each size. In fact, the root lists can contain arbitrarily many trees of the same size. You will restore these invariants during EXTRACT-MIN operation.

On the bright side, we can also implement INSERT(Q, x) in $O(1)$ *worst-case* time:

```
function INSERT(Q, x)
    Q' ← new BINOMIALHEAP(x)        ▷ Create a new binomial heap containing a single element
    return UNION(Q, Q')
```

(a) **(45 pts)** Design an algorithm that implements EXTRACT-MIN(Q). Your algorithm should extract and return the minimum element x from Q. Moreover, it should append the children of x to the root list and consolidate the trees in the newly created root list, so the resulting root list contains at most one tree of each size ordered by their sizes.

Remember: The trees in the root list are not necessarily in the order of the tree sizes anymore. The running time of your algorithm in the worst case should be $O(\max\{k, \log n\})$, where k is the number of trees in the root list after extracting x but before the consolidation and n is the number of elements in the heap. *Hint: Think about how many trees will remain in the root list after consolidation and use an additional intermediate data structure to perform consolidation efficiently.*

Important: Throughout this course, when you are asked to "design an algorithm", you should (a) describe/explain your algorithm in plain English first, (b) write down the pseudocode, (c) prove its correctness (e.g., by stating what invariant(s) it maintains and showing/proving that they are actually maintained in your pseudocode), and (d) analyze its running time.

(b) **(45 pts)** Using amortized analysis prove that the *amortized cost* of EXTRACT-MIN(Q) is $O(\log n)$. You may use either the potential or the accounting method. *Hint: To come up with the potential function, you might want to do the exercise of proving the $O(1)$ amortized cost of* INSERT*(Q, x) in the regular binomial heap (in class I mentioned that it is similar to incrementing a binary counter, but thinking about the direct proof, will help you with analyzing* EXTRACT-MIN*(Q)).*

(c) **(10 pts)** Observe that restoring the order of trees in the root list serves no purpose. Explain how this observation can help you optimize the lazy binomial heap by saving on the storage of a single pointer. Can you optimize even further by storing the root list as a singly-linked list? Explain why or why not. *Hint: consider what happens during every operation.*