# Homework Assignment #5
## Due: October 25, 2024 at 5:00 p.m.

**1.** This assignment investigates an interesting connection between the union-find ADT and the priority queue ADT.

Suppose we would like to process a sequence $\sigma$ of INSERT and EXTRACTMIN operations on an initially empty priority queue, and that the entire sequence of operations is known in advance. Each of the values in $\{1, 2, \ldots, n\}$ is inserted by exactly one INSERT of the sequence. There are $m$ EXTRACTMIN operations in the sequence. The sequence has the form $\sigma = \sigma_1 E_1 \sigma_2 E_2 \cdots \sigma_m E_m \sigma_{m+1}$ where each $\sigma_k$ is a sequence of INSERTS and each $E_k$ is an EXTRACTMIN operation.

Our goal is to figure out the results returned by all of the EXTRACTMINS.

[3] **(a)** Give good upper bounds (in terms of $m$ and $n$) on the total time it would take to execute the sequence $\sigma$ on a priority queue implemented using

- a binary heap,
- a binomial heap, or
- a Fibonacci heap.

[2] **(b)** In the rest of this question, we will consider an algorithm to compute the output of each EXTRACTMIN in $\sigma$ *without* actually running the sequence of operations. Instead of looking at each EXTRACTMIN and figuring out which element it extracts, we will look at each element and figure out which EXTRACTMIN (if any) extracts it.

Our algorithm will use trees to represent a union-find data structure with union by rank and path compression (as described in Section 19.3 of the textbook). We will also store some additional information in the root of each tree:

- an integer *label*, and
- pointers *prev* and *next* used to form a doubly-linked list of the roots.

```
1:  for k ← 1..m + 1 do
2:      create a root node r_k (of rank 1) with label k
3:      create a node (of rank 0) for each element inserted during σ_k and make it a child of r_k
4:      append r_k to the end of the doubly-linked list of roots
5:  for i ← 1..n do
6:      root ← FindSet(i)
7:      if root.label ≤ m then
8:          output: item i is extracted by E_{root.label}
9:          ℓ ← root.next.label
10:         perform a Union(root, root.next) and set label of root of the merged tree to ℓ
11:         remove the root that became a child of the other from the linked list of roots
12:     else
13:         output: item i is not extracted during σ
```

Draw a picture of the data structure at the end of the loop at lines 1–4 for the sequence

$\sigma =$ Insert(3) Insert(4) ExtractMin Insert(2) ExtractMin ExtractMin Insert(1).

Then, draw a picture of the data structure after two iterations of the loop at lines 5–13 have been completed.

Indicate the values of all fields of the nodes in your pictures.

[5] **(c)** Explain why the following loop invariants are true at the beginning of each iteration of the loop at lines 5–13.

(i) The doubly-linked list of roots is sorted in strictly increasing order by the root labels.

(ii) For $1 \leq j \leq n$, if $r_j$ is the root of the tree containing $j$, then $j$ is inserted before $E_{r_j.label}$ and (if $r_j$ is not the first root in the doubly-linked list) after $E_{r_j.pred.label}$.

(iii) Suppose $\ell$ is the label of the tree containing $i$. If $\ell \leq m$ then $E_\ell$ should return $i$. If $\ell = m + 1$ then $i$ should still be in the priority queue at the end of $\sigma$.

(iv) For each root $r$ in the root list with $r.label \leq m$, the values $1, 2, \ldots, i-1$ are *not* in the priority queue when $E_{r.label}$ occurs.

Note that the correctness of all the outputs of the algorithm follows from (iii).

Hint: for your induction step, consider assume that all four conditions hold at the beginning of iterations 1 to $i$ and prove that they all hold at the beginning of iteration $i + 1$.

Remark: You should prove all four claims, but the TA may base your grade for this part on a spot check of just a portion of the proof. Please structure your proof clearly so that the TA can find the relevant portion easily.

[1] **(d)** Give good upper bound on the running time of the algorithm.