Homework Assignment #4
Due: October 4, 2024 at 5:00 p.m.

1. Suppose we want to add a new operation KTH-SMALLEST(k) to a priority queue that returns the k smallest *key* in the priority queue without changing the contents of the priority queue. For simplicity, you can assume throughout this question that all keys stored in a priority queue are distinct.

Obviously, we can perform this operation in $O(k \log n)$ time on a binomial (or binary) heap containing n keys by simply performing k EXTRACTMIN operations and then re-inserting the k extracted keys. This question describes how to add the new operation to a (modified) binomial heap so that it runs in $O(k \log k)$ time.

In a standard binomial heap, the children of a node are sorted according to the degrees of those children. Suppose that we instead order the children of each node so that they are sorted according to the *values* stored in the children.

To store the children of a node, we will use a *super-list* ADT (sorted by the children's keys) instead of a singly-linked list. In a super-list, you can still find the head of the list and the successor of a given element in constant time, just like a singly-linked list, but you can also perform insertions into a super-list of length ℓ (maintaining the sorted order) in $O(\log \ell)$ time. (In lectures, we shall soon see how to implement such a super-list; for this exercise you can just assume that you are given an implementation of a super-list.)

We will also keep the singly-linked list of roots of the binomial trees that make up the binomial heap sorted by key instead of by degree.

[4] **(a)** Describe how to perform UNION on our modified binomial heaps so that it takes $O(\log n \log \log n)$ time in the worst case, where n is the size of the larger heap being UNIONED. If you wish, you can just explain how your UNION differs from the UNION routine on a standard binomial heap. You need not give a detailed proof of correctness, but you should briefly explain why your algorithm is correct and why it achieves the stated running time.

[4] **(b)** How long would the INSERT and EXTRACT-MIN operations take in this modified version of binomial heaps? Briefly explain why your answer is correct.

[3] **(c)** Draw a picture that shows one possible shape of a (modified) binomial heap of size 19. Do *not* fill the nodes with keys. Colour all nodes that might possibly contain the 5th smallest key.

[4] **(d)** Describe how to implement KTH-SMALLEST(k), which outputs the kth smallest key in the (modified) binomial heap. The operation should not modify the heap. Your algorithm should run in $O(k \log k)$ time in the worst case. Explain why your algorithm is correct and why it achieves this running time.

Hint: your algorithm can use a small, auxiliary priority queue to find the answer.