Written #1: UF and PQ
Due: see Canvas

This is your first written homework assignment. You should submit your solutions as (one or more) PDF documents on Canvas. Your work may be typed up on a computer, or just hand-written and scanned, either is OK as long as it is legible.

See Canvas for the duedate and other requirements. Like all academic work at Emory College, this is governed by the Honor Code[1].

Problem 1. Read share/writ1/QuickFindUF2.java. This is a more efficient variation of the textbook class `QuickFindUF`. In this version, we keep a circular linked list of all the items in each component. When we join two components, we traverse the *smaller* list, resetting the id of each item to that of the larger component.

1(a). The textbook UF datastructures use $N + O(1)$ or $2N + O(1)$ words of memory, for a structure of size N. (A word is enough memory to store one `int`, and the "$+O(1)$" term includes some JVM overhead to keep track of the object and array.) How much space is used by a `QuickFindUF2`[2] of size N?

1(b). Fix some item i. Argue that whenever a `union` operation modifies `id[`i`]`, the size of i's component at least doubles. Conclude that `id[`i`]` is modified $O(\lg N)$ times.

1(c). Suppose we construct a `QuickFindUF2` structure of size N, and then we do some sequence of M operations (unions and finds) on it. Argue that the total running time is $O(M + N \lg N)$.

Remark: Your answer here should use the previous part, and the idea is to "charge" the work of traversing a list to the items in that list. You only need a few sentences.

Problem 2. Suppose we implement a d-ary heap in an array (see the book, d is a small integer, $d \geq 2$). Suppose we store the root item at index 0 of the array, its leftmost child at 1, its next child at 2, and so on. (The book likes to put the root at index 1, but we will put it at 0.) Suppose the heap stores $n \geq 1$ items.

2(a). Suppose an item is at array index k. What is the index of its leftmost child?

2(b). Supposing $k > 0$, what is the index of its parent?

2(c). Suppose we want to start the first phase of heapsort (putting the given array into max-heap order). What is the largest index k, so that k has at least one child?

2(d). Give an exact formula for the tree height h, in terms of the size n. (Check your base cases: when $n = 1$ you should have $h = 0$, and when $2 \leq n \leq d + 1$ you should have $h = 1$).

Remark: Each answer is just an integer-valued formula, you may need to round up or round down (like $\lfloor k/2 \rfloor$ or $\lceil k/2 \rceil$). You do not need to prove anything.

[1] http://catalog.college.emory.edu/academic/policies-regulations/honor-code.html
[2] 1/20: fixed a typo here, it had said "`QuickFindUF`".

Problem 3. Suppose we we are reading an input stream of N numbers, and after reading the stream, we want to report the $M = \sqrt{N}$ largest numbers from the stream. For both parts, describe how to solve the problem using a *binary heap.* (One of these is "TopM" in your book, pages 310-311. For the other, you need to use the linear-time heap construction method of Section 2.4, the first phase of heapsort, and you need to argue $M \lg N = O(N)$.)

3(a). Solve the problem in $O(N)$ time and $O(N)$ space.

3(b). Solve the problem in $O(N \lg M)$ time and $O(M)$ space.

Remark: In one part you can refer to the book, in the other you need to present a new algorithm. English or pseudocode is enough, you do not have to use Java. Be clear about whether you need a "min" heap or a "max" heap. "Space" counts words of memory, where a word is large enough to store an input number, an index, or a reference. These two solutions demonstrate a "trade-off" between time and space.