Summer 2010 **CMSC 351: Homework 5** Clyde Kruskal

Due at the start of class Monday, June 21, 2010.

Problem 1. Consider an array of size eight with the numbers in the following order $20, 40, 60, 80, 10, 30, 50, 70$.

(a) What is the array after heap formation? How many comparisons does the standard algorithm use?

(b) Show the array after each element sifts down *after heap creation*. How many comparisons does the standard algorithm use for all of the sifts?

(c) How many comparisons does the modified algorithm (Floyd's version) use to create the heap?

(d) How many comparisons does the modified algorithm (Floyd's version) use for the remainder of the sort?

Problem 2. A d-ary heap is like a binary heap, but instead of two children, nodes have d children.

(a) How would you represent a d-ary heap in an array?

(b) What is the height of a d-ary heap of n elements in terms of n and d.

(c) Explain loosely (but clearly) how to extract the maximum element from the d-ary heap (and restore the heap). How many comparisons does it require?

(d) How many comparisons does it take to sort? Just get the high order term exactly, but show your calculations.

(e) What value(s) of d are optimal? Justify your answer.

Problem 3. Assume n is a power of 2. If you view heap creation as a recursive procedure, you get approximately the following recurrence for the number of comparisons:

$$T(n) = \begin{cases} 2T(n/2) + 2\lg n & \text{if } n > 1 \\ 0 & \text{if } n = 1 \end{cases}$$

(a) Use constructive induction to show that $T(n) = an + b\lg n + c$. Find constants a, b, and c.

(b) Solve the recurrence using the iteration method.

Problem 4. Assume you have k sorted lists and wish to merge them into a single sorted list. Show that you can do this in $O(n \lg k)$ time. (*Hint:* Use a min-heap. What values should be in the heap?)