Computer Science B63
Scarborough Campus

May 16, 2008
University of Toronto

Homework Assignment #1
Due: May 29, 2008, by 1:10 pm

1. *Please complete and attach (with a staple) an assignment cover page to the front of your assignment.*
2. *If you do not know the answer to a question, and you write "I do not know the answer to this question", you will receive 20% of the marks of that question. If you just leave a question blank with no such statement, you get 0 marks for that question.*
3. *Unless we explicitly state otherwise, you should justify your answers. Your paper will be marked based on the correctness and completeness of your answers, and the clarity, precision and conciseness of your presentation.*

Question 1. (12 marks) Consider the following algorithm for sorting an array of n integers (indexed as $A[0..n-1]$) using HeapSort (the objects stored in the heap are integers, and the key for each integer is its value).

```
HeapSort(A):
    $n \leftarrow A$.length
    $H \leftarrow$ BuildBinaryHeap($A$)
    for $i \leftarrow n$–1 downto 0 do
        $A[i] \leftarrow H$.ExtractMax()
    return $A$
```

a. Prove that the worst-case running time of the above HeapSort algorithm is in $O(n \log n)$.

b. Prove that the worst-case running time of the above HeapSort algorithm is in $\Omega(n \log n)$.

Hint: Remember that even though we saw that the worst case time for an Extract operation is $\Omega(\log n)$, not every Extract operation may require this much time.

Question 2. (15 marks) Suppose you are given k lists of integers, each list sorted in nondecreasing order. We want to merge these k lists into a single sorted list.

Describe an algorithm that takes as input k and the k sorted lists, and constructs a new list containing all the elements of the the k input lists in nondecreasing order. Justify the correctness of your algorithm and prove that its worst-case running time is in $O(n \log k)$, where n is the total number of elements over all k lists.

Question 3. (18 marks) This question is about the worst-case cost of successively inserting k elements into a binomial heap that initially contained n elements.

a. (6 marks) Prove that a binomial heap with n elements has exactly $n - \alpha(n)$ edges, where $\alpha(n)$ is the number of 1's in the binary representation of n.

b. (12 marks) Consider the worst-case total cost of successively inserting k new elements into a binomial heap H that contained n elements. In this question, we measure the worst-case cost of inserting a new element into H as the maximum number of pairwise comparisons between elements of the binomial heap that is required to do this insertion. In class we proved that for $k = 1$ (i.e., inserting one element) the worst-case cost is $O(\log n)$. Show that if $k > \log n$, then the worst-case *total* cost of successively inserting

k elements into H is only $O(k)$. In other words, if $k > \log n$, then the *average* cost of an insertion (i.e., the worst-case total cost divided by k) is bounded by a *constant*.

Hint: Note that the cost of each one of the k consecutive insertions varies — some can be expensive, other are cheaper. Relate the cost of each insertion, i.e., the number of pairwise comparisons that it requires, with the number of extra edges that it forms. Then use part (a).

Question 4. (20 marks) This question is a programming assignment. To see its description follow the link given in the "Assignments" section of the course web page.