CS2351 Data Structures

Homework 1

Due: March 28, 2011 (Before Class)

1. In a binary heap that we have learnt in class, each node has up to two children (left and right children), and the tree is filled such that (i) all levels except the last one are full, and (ii) leaves are filled from left to right in the last level.

   Now, we study a variation of the heap called *ternary heap*, in which each node has up to three children (left, middle, and right). The rules of filling are the same as in a binary heap.

   (a) (10%) Suppose we implement the ternary heap with an array, where the root has index 1. Explain how to find the left, the middle, and the right children of a node with index x.

   (b) (10%) Explain briefly how to perform Extract-Min in a ternary heap.

   (c) (10%) Assume that there are n elements in the heap. What is the asymptotic time complexity to perform Extract-Min in a ternary heap? How does it compare with the asymptotic time complexity of Extract-Min in a binary heap?

2. We have n electric light bulbs, with labels from 1 to n. The light bulbs are all turned OFF initially. Each light bulb has a switch, so that flipping the switch will change its status from ON to OFF, or from OFF to ON. At the ith round, for $i = 1, 2, \ldots, n$, we will flip the switches of those light bulbs whose labels are a multiple of i, and we are interested to know which light bulbs will be ON after the nth round.

   We can compute the final status of each light bulb using the following code:

```
Initialize A[1..n] so that each entry is OFF;
for (round i = 1, 2, ..., n) {
    for (position j = i, 2i, 3i, ...) {
        if (j ≤ n)   Flip A[i];
        else         break;
    }
}
```

   (a) (10%) Show that the above code runs in $\Theta(n \log n)$ time.

   (b) (10%) Design a faster algorithm that can compute the final status in $O(n)$ time.

3. Bubble Sort is a very simple algorithm that can sort an array $A[1..n]$ of n numbers. It works in $n - 1$ rounds as follows:

```
for (round j = 1, 2, ..., n − 1) {
    for (position i = 1, 2, ..., n − j) {
        if (A[i] > A[i + 1])
            Swap A[i] with A[i + 1];
    }
}
```

(a) (10%) Show that Bubble Sort is correct.
Hint: What will happen to the last entry after the first round?

(b) (10%) Suppose x and y are two numbers in the array, with $x > y$, such that x appears earlier than y in the array before the sorting starts. Show that x will be swapped with y *exactly once* during the sorting process.

(c) (10%) Construct an example to show that in the worst case $\Omega(n^2)$ swaps will be performed.

4. We have an array of n numbers, and we want to find out the k smallest elements. One can do so in $O(n \log n)$ time by first sorting the n numbers, or in $O(kn)$ time by repeatedly selecting the smallest element from the array. However, we can do better.

   (20%) Design an algorithm to perform the task in $O(n \log k)$ time. Give a clear explanation of its correctness and time analysis. *Hint:* Use a heap.