Algorithms Lab 2

The in-lab problems are to be solved during the lab time. Work with your team, but write your solutions individually. Your answers will not be graded, however you need to work through these problems; please staple the answers to the in-class exercises to the assignment that you hand in.

The homework problem set is due in one week. Work with your team, but write your solutions individually. List the people with whom you discussed the problems.

1 In lab exercises

1. Finish/review the exercises from class, which you can also find on the class website.
2. Find a simple formula for $\sum_{k=1}^{n}(2k-1)$.
3. (CLRS 3-4.a) Prove or disprove: $f = O(g)$ implies that $g = O(f)$.
4. Prove or disprove: $f = O(g)$ implies that $g = \Omega(f)$.
5. Consider the folowing code for BubbleSort that we discussed in class:

```
BUBBLE-SORT(A)
1  For $k = 1$ to $n - 1$
2       // do a bubble pass
3       For $i = 0$ to $n - 2$
4            if $A[i] > A[i+1]$: swap
```

   (a) Give the best-case and worst-case running time of the algorithm. Express them as $\Theta()$ bounds and give a brief justification.

   (b) Show how to change the code so that the algorithm dos not do any redundant bubble-passes (i.e. if the input needs only 3 bubble passes to be sorted, the algorithm does only 3 passes).

6. Finish the in-class exercises on heaps.

2 Homework problems

Your assignment will be evaluated based not only on the final answer, but also on clarity, neatness and attention to details. For example, you'll want to leave plenty of space in between problems so that we can give you feedback.

For problems 1 through 6, find a tight bound for the solution of the following recurrences using iteration.

1. $T(n) = T(n/3) + 1$
2. $T(n) = T(n/3) + n$
3. $T(n) = T(\sqrt{n}) + 1$
4. $T(n) = T(n-1) + n$
5. $T(n) = 7T(n/2) + n^3$
6. $T(n) = 7T(n/2) + n^2$
7. (GT C-2.31) Develop an algorithm that computes the kth smallest element in a set of n distinct integers in $O(n + k \lg n)$ time.
8. (CLRS 6.5-9) Give an $O(n \lg k)$-time algorithm to merge k sorted lists into one sorted list, where n is the total number of elements in all the input lists.
9. (CLRS 9-1) Given a set of n numbers, we wish to find the i largest in sorted order using a comparison-based algorithm. Find the algorithm that implements each of the following methods with the best asymptotic worst-case running time, and analyze the running times of the algorithms on terms of n and i.

   (a) Sort the numbers, and list the i largest.

   (b) Build a max-priority queue from the numbers, and call EXTRACT-MAX i times.