Homework 1

CS 4104 (Spring 2014)

Assigned on Wednesday, January 29, 2014.
Submit a PDF file containing your solutions on Scholar by the beginning of class on Wednesday, February 5, 2014.

Instructions:

- You can pair up with another student to solve the homework. You are allowed to discuss possible algorithms and bounce ideas with your team-mate. Do not discuss proofs of correctness or running time in detail with your team-mate. Please form teams yourselves. Of course, you can ask me for help if you cannot find a team-mate. You may choose to work alone. *Each of you must write down your solution individually, and write down the name of the other member in your team. If you do not have a team-mate, please say so.*
- Apart from your team-mate, you are not allowed to consult any sources other than your textbook, the slides on the course web page, your own class notes, the TAs, and the instructor. In particular, do not use a search engine.
- Do not forget to typeset your solutions. *Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as n^2 and not as "n^2".* Students can use the LaTeX version of the homework problems to start entering their solutions.
- Describe your algorithms as clearly as possible. The style used in the book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed pseudo-code without an explanation, we will not grade your solutions.*
- Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution.
- Do not describe your algorithms only for a specific example you may have worked out.
- You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. *You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.*
- Describe an analysis of your algorithm and state and prove the running time. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.

Problem 1 (35 points) Solve exercise 4 in Chapter 2 (pages 67–68) of "Algorithm Design" by Kleinberg and Tardos. Provide a proof for your solution. There are 21 pairs of functions. You need not compare every pair of functions in your proof. By judicial choice of which functions to compare, the number of function pairs you need to compare is much smaller than 21.

Notes.

1. Keep in mind that in order to prove that $f(n)$ is $O(g(n))$, you must exhibit the constants required by the definition of asymptotic upper bound, and show that the inequality in the definition is satisfied by the functions, given these constants.
2. Note that $g_4(n)$ is printed before $g_3(n)$ in the textbook.

To get you started, here is how you might prove that $g_3(n)$ is $O(g_4(n))$. Dividing both functions by n, it is enough to prove that $(\log n)^3$ is $O(n^{1/3})$.[1] Let us take the cube root of both functions. Now it suffices to show that $\log n$ is $O(n^{1/9})$. We know from the class slides or from equation 2.8 on page 41 of the textbook that this statement is true. Note that I will not require you to prove mathematically that $\log n$ is $O(n^{1/9})$; you can rely on the statement in the slides/textbook.

Problem 2 (25 points) (20 points) Solve exercise 5 in Chapter 2 (page 68) of "Algorithm Design" by Kleinberg and Tardos. If you decide that a statement is true, provide a short proof. Otherwise, provide a counterexample.

Problem 3 (40 points). Describe an algorithm that uses a priority queue (heap) to merge k sorted lists into one sorted list. Each sorted list has n integers. Neither k nor n is a constant.

To get you started, consider the following algorithm:

1. Insert all the kn numbers into a priority queue, with each number being its own key.
2. Repeatedly report the smallest key in the queue and delete this value from the queue, until the queue becomes empty.

What is the running time of this algorithm?

Now develop an algorithm that has a better running time. Describe your algorithm, prove its correctness, and provide an analysis of the running time. Don't forget to show that your algorithm has a faster running time than the one described above!

[1]Please note that I put a pair of parenthesis around $\log n$ before cubing it. Writing $\log n^3$ or $\log(n)^3$ to denote the cube of the logarithm of n is wrong.