Problem Set 1

This problem set is not due and it will not be graded, but you are welcome to come to office hours to discuss your solutions. This is a set of practice problems for the midterm and final, and it covers the first quarter of the course.

The description of your proofs and algorithms should be as *clear* as possible (which does not mean *long* – in fact, typically, good clear explanations are also short.)

When a problem asks to give an algorithm, in your solution: (i) describe shortly and informally the main ideas in your solution; (ii) give a detailed description of the algorithm, using a style similar to the pseudo-code used in class or in a textbook; (iii) prove the correctness of the algorithm; (iv) prove a bound on the time complexity of the algorithm. You can omit the proof of correctness if it is clear from the description of the algorithm.

---

1. $O(\cdot)$ **Notation**

   (a) Give the best (slowest growing) big-Oh bound for $f(n) = \sum_{k=1}^{n} k^r$, where $r > 0$ is a fixed constant.

   (b) Which of the following statements is true or false?

$$n^2 + 4n \log n = O(n^2) \tag{1}$$
$$2^n = O(n^2) \tag{2}$$
$$\log n = O(n) \tag{3}$$
$$n^3 + 3n^2 = O(n^2) \tag{4}$$

2. **Recurrence relations**

   Solve the following recurrence relations (c is a constant).

   (a) $T(n) = 5 \cdot T(\frac{n}{4}) + cn^2$

   (b) $T(n) = 3 \cdot T(\frac{n}{2}) + cn$

   (c) $T(n) = 27 \cdot T(\frac{n}{3}) + cn^3$

   (d) $T(n) = 2 \cdot T(\frac{n}{2}) + \sqrt{n}$

   (e) $T(n) = 3 \cdot T(\frac{n}{3}) + cn^2$

3. **Divide and Conquer** Given a sorted array $A[1], \ldots, A[n]$, design and analyse an $O(\log n)$ time algorithm that finds an index i such that $A[i] = i$, if such an index i exists.

4. **Strongly Connected Components**

   One of the following statements is true. Say which one and prove it.

   (a) If a directed graph G has k strongly connected components, by adding one more edge to G the number of strongly connected components can drop at most by 1 (i.e. the new graph obtained from G by adding one edge has at least $k-1$ strongly connected components).

   (b) For every k, there exists a graph G that has k strongly connected components and such that if we add one particular edge to G, we can make it be strongly connected (i.e. the new graph has only 1 strongly connected component).

5. **Minimum Spanning Tree**

   Prove that the following algorithm for the minimum spanning tree problem is correct, or show an example of a graph where the algorithm fails. In either case, discuss how to efficiently implement the algorithm, and what is the resulting running time. Assume the graph is represented with adjacency lists.

   Algorithm A(G=(V,E): graph, w: weights)
       sort the edges of G into non-increasing order of weight
       $T = E$
       for all $e \in E$ in non-increasing order of weight do
           if $T - \{e\}$ is connected then $T = T - \{e\}$
       return T