Homework 1. Due April 23

CS180: Algorithms and Complexity
Spring 2018

GUIDELINES:

- Upload your assignments to Gradescope by 5:59 PM.
- Follow the instructions mentioned on the course webpage for uploading to Gradescope very carefully (including starting each problem on a new page and matching the pages with the assignments); this makes it easy and smooth for everyone. As the guidelines are simple enough, bad uploads will not be graded.
- You may use results proved in class without proofs as long as you state them clearly.
- Most importantly, make sure you adhere to the policies for academic honesty set out on the course webpage. The policies will be enforced strictly. Homework is a stepping stone for exams; keep in mind that reasonable partial credit will be awarded and trying the problems will help you a lot for the exams.

1. In class we gave an algorithm for finding the closest pair of points on the plane. The notion of distance we used was Euclidean distance: $d((x,y),(x',y')) = \sqrt{(x-x')^2 + (y-y')^2}$.

   Show how you can adapt the algorithm to find the closest pair for a different notion of distance. Specifically, give an algorithm with run-time $O(n \log n)$ for the following problem. Given a set of points $P = \{p_1, \ldots, p_n\}$ in the plane, find a pair of points with the smallest possible L1-distance among the points where L1-distance between two points is defined by $d_1((x,y),(x',y')) = |x-x'| + |y-y'|$.

   For full-credit your algorithm should be correct and you should explain why your algorithm works. In particular, if your algorithm is similar to what we did in class, you need to prove the analogue of the main lemma (one that allowed us to look only 15 distances ahead for Euclidean distance) we did in class. [.75 points]

2. An array $A[0, 1, \ldots, n-1]$ is said to have a *majority element* if more than half of its elements are the same. Given an array, the task is to design an efficient algorithm to tell whether the array has a majority element, and, if so, to find that element. The elements of the array are not necessarily from some ordered domain like the integers, and so there can be no comparisons of the form "is $A[i] > A[j]$?". (Think of the array elements as mp3 files, say; so in particular, you cannot sort the elements.) However you can answer questions of the form: "is $A[i] = A[j]$" in constant time.

   Give an algorithm to solve the problem. For full-credit, your algorithm should run in time $O(n \log n)$ and you need to bound the time-complexity of the algorithm. (You don't have to

prove correctness.) [.75 points]

(Hint: Split the array A into two arrays A_L and A_R of half the size each. Does knowing the majority elements of A_L and A_R help you figure out the majority element of A? If so, you can use a divide-and-conquer approach.)

3. Run the BFS algorithm for the following graph with $s = 1$ and $t = 9$: $G = (V, E)$, where $V = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$, and $E = \{\{1, 2\}, \{1, 3\}, \{2, 3\}, \{2, 4\}, \{2, 5\}, \{3, 5\}, \{3, 7\}, \{4, 5\}, \{5, 6\}, \{8, 9\}\}$. Your work should the step-by-step evolution of the lists $L_0, L_1, ...$ and of the array $DISCOVERED$ as we did in class (cf. the transcript uploaded in the calendar). [.75 points]

4. Given a graph $G = (V, E)$ in adjacency list representation, give an algorithm that runs in time $O(|V| \cdot |E|)$ to check if G has a 'triangle', i.e., a triple of distinct vertices $\{u, v, w\}$ such that all three edges between them are present in G. [.75 points]

Additional Problems. DO NOT turn in answers for the following problems - they are meant for your curiosity and understanding.

1. Problems 3.9, 3.11 from textbook [KT].

2. Problem 3.16 from Chapter 3 of [DPV].