CS4234: Optimization Algorithms

Problem Set 1

Due: August 18th, 11:9pm

Instructions. This problem set reviews some basic algorithms that we will need in the coming weeks, asks you to design an approximation algorithm, and asks you to analyze a parameterized algorithm.

The problem set begins with a set of exercises (E-problems) that you do not need to hand in. They are primarily for review, and on this problem set, are primarily related to material you have already seen in previous modules.

Solve all the standard problems (S-problems).

- Start each problem on a separate page.
- Make sure your name is on each sheet of paper (and legible).
- Staple the pages together, and hand it in before class starts, or submit it on IVLE. Alternatively, if you submit it late, you can put it in the envelope next to my office door (and send me an e-mail).

Remember, that when a question asks for an algorithm, you should:

- First, given an overview of your answer. Think of this as the executive summary.
- Second, describe your algorithm in English, giving pseudocode if helpful.
- Third, give an example showing how your algorithm works. Draw a picture.

You may then give a proof of correctness, or explanation, of why your algorithm is correct, an analysis of the running time, and/or an analysis of the approximation ratio, depending on what the question is asking for.

Advice. Start the problem set early—some questions take time. Come talk to me about the questions. (Different students have different questions. Some have questions about how to write a good proof. Others need pointers of designing an algorithm. Still others want to understand the material from lecture more deeply before applying it to the problem sets.) I'm here for you to talk to.

Collaboration Policy. The submitted solution must be your own unique work. You may discuss your high-level approach and strategy with others, but you must then: (i) destroy any notes; (ii) spend 30 minutes on facebook or some other non-technical activity; (iii) write up the solution on your own; (iv) list all your collaborators. Similarly, you may use the internet to learn basic material, but do not search for answers to the problem set questions. You may not use any solutions that you find elsewhere, e.g. on the internet. Any similarity to other students' submissions will be treated as cheating.

Exercises (and Review) (*Do not submit.*)

Ex-1.

Problem 1.a. Show that for any real constants a and b, where $b > 0$, $(n + a)^b = \Theta(n^b)$.

Problem 1.b. Show that $\sum_{k=1}^{n} \frac{1}{2k-1} = \ln \sqrt{n} + O(1)$ by manipulating the harmonic series.

Ex-2. Unimodal Search

An array $A[1..n]$ is unimodal if it consists of an increasing sequence followed by a decreasing sequence, or more precisely, if there is an index $m \in \{1, 2, \ldots, n\}$ such that:

- $A[i] < A[i + 1]$ for all $1 \leq i < m$
- $A[i] > A[i + 1]$ for all $m \leq i < n$.

In particular, $A[m]$ is the maximum element, and it is the unique "locally maximum" element surrounded by smaller elements ($A[m - 1]$ and $A[m + 1]$). Give an algorithm to compute the maximum element of a unimodal input array $A[1..n]$ in $O(\log n)$ time. Prove the correctness of your algorithm, and analyze its running time.

Ex-3. Given two sorted sequences X and Y with n elements each, design and analyze an efficient divide-and-conquer algorithm to find the median in the merged list of X and Y. The optimal (comparison-based) algorithm runs in time $O(\log n)$.

Ex-4. Show that vertex cover (as a decision problem) is NP-complete by reduction from Max-Clique. Recall that the Max-Clique problem is defined as follows:

> Given a graph $G = (V, E)$ and a parameter k, decide whether G contains a clique of size k.

(See the lecture notes if you need a hint. Be careful in cases where you may need to prove both directions of the "if-and-only-if.")

Standard Problems (to be submitted)

S-1. You are given a graph $G = (V, E)$ with n nodes and m edges. (Perhaps the graph represents a telephone network.) Each edge is colored either blue or red. (Perhaps the blue edges are owned by Singtel and the red edges are owned by M1.) You are also given a parameter k as part of the input. (Perhaps your contract with the Singapore government specifies the precise value of k.)

Give an algorithm that finds a spanning tree of G with *exactly* k blue edges (and exactly $n-k-1$ red edges). Give the running time of your algorithm, and show that it is correct.

S-2. You are given an undirected graph $G = (V, E)$. The goal of this problem is to remove a minimum number of edges such that the residual graph contains no triangles. (I.e., there is no set of three vertices (a, b, c) such that edges (a, b), (b, c), and (a, c) are all in the residual graph.)

Give a 3-approximation algorithm that runs in polynomial time. (That is, it should guarantee that there are no triangles, and that the number of edges removed is within a factor of 3 of optimal.)

S-3. Consider a set of n points in the Euclidean (2-dimensional) plane:

$$P = (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n).$$

We say that a point (x_i, y_i) **dominates** a point (x_j, y_j) if it is bigger in both coordinates, i.e., if $(x_i > x_j)$ and $(y_i > y_j)$. A **maximal** point is not dominated by any other. In this problem, we will develop and analyze an algorithm for finding the set of maximal points.

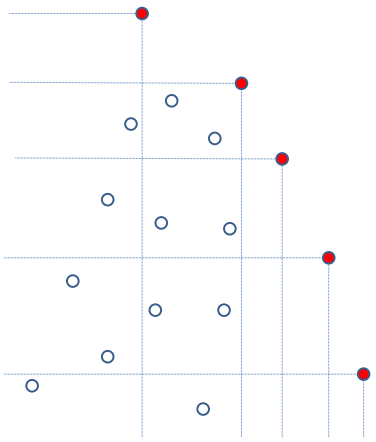

Figure 1: The solid circles are the maximal points, while the empty circles are dominated. The dashed lines indicate which points are dominated by a given solid circle.

Assume, for this problem, that all the x-coordinate and y-coordinate values are distinct. Consider the following solution, known as the Kirkpatrick-Seidel algorithm:

```
1  Algorithm: KirkpatrickSeidel(P)
2  Procedure:
3      if P = ∅ then return ∅
4      if |P| = 1 then return P
5      Let x be the median x-coordinate of the points in P.
6      Let P_ℓ be all the points with x-coordinate ≤ x.
7      Let P_r be all the points with x-coordinate > x.
8      Let q be the point in P_r with the maximum y-coordinate.
9      Delete q from P_r.
10     Delete every point dominated by q in P_ℓ.
11     Delete every point dominated by q in P_r.
12     S_ℓ = KirkpatrickSeidel(P_ℓ)
13     S_r = KirkpatrickSeidel(P_r)
14     return S_ℓ ∪ S_r ∪ {q}
```

Problem 1.a. Prove that the `KirkpatrickSeidel` algorithm correctly returns the set of maximal points.

Problem 1.b. Prove (briefly) that the `KirkpatrickSeidel` algorithm runs in time $O(n \log n)$.

Problem 1.c. Assume that there are only h maximal points in the set P (but h is not known in advance). Show that the `KirkpatrickSeidel` algorithms runs in time $O(n \log h)$. (Hint: Beware that the recursion is not always balanced: for example, in one half, all the points may be deleted in line 10 or 11.)

Discussion: Notice that this is an example of parameterized analysis, where the running time depends on the parameter h. The `KirkpatrickSeidel` algorithm is, in fact, even better in that it is *instance optimal*, i.e., for any given input set P, there is no algorithm that is faster, in a sense.