Group Exercise 1

DD2440: Advanced Algorithms

2017

Instructions

- The deadlines in this course are strict. The solution should be submitted by uploading PDF files on **www.peergrade.io**. Please select the option to upload as group when you upload your solution on peergrade.io. Please remember to add your collaborator into the group. Note that solutions for different problems should be submitted to different channels. If you submit late, the system may not accept your submission and there is nothing we can do since the grading process may start immediately. (But let the teacher knows ASAP anyway when this happens.)
- Be kind to your fellow students when you write your solution. Write legibly and neatly. (Typed solution in latex is strongly encouraged.) Avoid excess verbosity: clear writing makes it easier for graders to understand the mechanics of your solution and improves your chances of being accepted. There is no fixed format for your submission. In particular, you can choose to be anonymous by not providing your name (and any other information) in the submitted solution.
- The group exercises should be solve in a group of 2-4 people. You can search for anything and talk about the questions with friends or ask friends for hints. However, you should write down the whole solution on your own. In particular, copying a solution from somewhere will be considered as plagiarism. (Ask if you're in doubt.)
- When asked to solve a computational problem please submit, in some form or other, the useful calculations that lead to the answer
- If something seems wrong, visit the course website to see if any errata was posted. If this does not help, then email danupon@gmail.com. Don't forget to prefix your email subject with [DD2440].
- Levels of problems might not always reflect their difficulty. So you are encouraged to try all problems. Model solutions might not be available for all problems, especially for those that nobody solve.

Bonus opportunity

- You can nominate your solutions as a model solution (where all class participants can use for grading) by sending your solution to danupon@gmail.com at least 48 hours before the deadline. You will be awarded 0.5 point for solving a problem of level at most C, and 1 point for a problem of level at least B.
- The TAs will check the correctness and presentation of your solutions. Your solution might be graded with a higher standard (compared to being peer-graded). So nominate your solution only if you are confident that it is correct and neatly presented. If accepted, your solution will be available to all class participants. In this case your solution will not be peer-graded and you will not have to peer-grade solutions for such question.
- You can nominate solutions for many problems (and you do not have to nominate all of them at once). But you will receive a credit of 1 point in total per problem set. In case there are many nominated solutions, we will pick 2-3 first ones that pass the bar for each question (ordered by nomination time).

E-Question

Alice and Bob each receives a set of $n \geq 100$ distinct words. They want to check whether at least 10% of their words are the same, i.e. there are at least $n/10$ distinct words that both Alice and Bob has in common. Describe and analyze an algorithm for Alice and Bob to exchange O(log n) words such that:

- If there are at least $n/10$ common words, the algorithm answers YES with high probability.
- If there are no common words at all, the algorithm always answers NO.
- Otherwise, the algorithm can answer anything.

Remark: This problem can be interpreted in two ways. You can use either of these interpretations:

1. Alice and Bob can only exchange words they have. They cannot exchange anything else.
2. Each word that Alice and Bob have is of O(1) length. They are allow to exchange O(log n) bits between each other. These bits might represent the words they have or any other information.

In both cases, we do not care the running time Alice and Bob need to pick these words (e.g. they can sort their own list before sending).

D-Question

Describe and analyze an algorithm for Problem 1 that needs to exchange O(log n) words such that

- If there are at least $n/10$ common words, the algorithm answers YES with high probability.
- If there are at most $n/20$ common words, the algorithm answers NO with high probability.
- Otherwise, the algorithm can answer anything.

Remark: Please see the remark of the E-question.

C-Question

Let A be an array containing edges of a graph G=(V, E) with n nodes, m edges, and maximum degree Δ, where $m \geq n$. Describe and analyze an algorithm that takes A and constant $\epsilon > 0$ as inputs. Then, it takes $\tilde{O}_{\epsilon}(n)$ times to output d' such that with high probability,

$$\frac{\Delta}{1+\epsilon} \leq d' \leq (1+\epsilon)\Delta.$$

B-Question (page 1/2)

Imagine you have an array $A[1\,..\,n]$. Each value in the array is an integer between 1 and M. Consider the following algorithm for finding the approximate sum of the values in the array: Fix $s = 1/\epsilon^2$. Your boss claims that this is a good algorithm, and gives the following proof:

Algorithm 1: Sum(A, n, s)

```
1 sum = 0
2 repeat s times
3     Choose a random i ∈ [1, n].
4     sum = sum + A[i].
5 return n(sum/s)
```

Let x_i be the value of the ith random sample, and $X = \sum(x_i)$. Let $A = \sum_i A[i]/n$, i.e., A is the average value and nA is the sum of the array (i.e., what we want to find). We know that $\mathrm{E}[X] = sA$. Then, using a Hoeffding Bound, we know that:

$$\Pr[|X - \mathrm{E}[X]| \geq \epsilon \mathrm{E}[X]] = \Pr[|X - sA| \geq \epsilon sA] \leq 2e^{-2(sA\epsilon)^2/s} .$$

B-Question (page 2/2)

We know that $A \geq 1$, so by choosing $s = 1/\epsilon^2$, we conclude that:

$$\begin{aligned} 2e^{-2(sA\epsilon)^2/s} &\leq 2e^{-2s\epsilon^2} \\ &\leq 2e^{-2} \end{aligned}$$

Thus, the probability of error is $\leq 1/3$. Thus, with probability at least $2/3$, $|X - sA| < \epsilon sA$, which implies that: $|nX/s - nA| < \epsilon nA$, i.e.,

$$nA(1-\epsilon) \leq sum \leq nA(1+\epsilon) \ .$$

We thus conclude that the algorithm returns a $(1 \pm \epsilon)$ estimate of the sum of the values in the array.

What is wrong with this algorithm and proof? Can you give an example of an array A where this algorithm will almost certainly give the wrong answer?

A-Question

Let A be an adjacency list for a connected graph G=(V, E) with n nodes m edges, and maximum degree $\Delta \leq 100$, where $m \geq n$. We want to distinguish between the following cases:

1. (YES Case) G is connected.
2. (NO Case) G contains at least $\Delta n/100$ connected components.

Design and analyze an algorithm that can distinguish the above two cases with high probability in $\tilde{O}(\Delta)$ time. (If the input graph does not fall in the above two cases, your algorithm can answer anything.)

Hint: For the second case, it might be useful to first show that there are at least $\Delta n/200$ "small" components. (You have to figure out what it means by "small".)

A*-Question

Let A be an adjacency list for a connected graph G=(V, E) with n nodes and m edges, where $m \geq n$. Additionally, A contains the degree of every node v (so you can ask for the degree of a node v in O(1) time). We want to $(1+\epsilon)$-approximate the average degree of G. Your boss claims that the algorithm below works with sufficiently large k and s. Either prove that he/she is wrong or correct. For the latter, show the value of k and s that are sufficient.

Algorithm AverageDegree$(G = (V, E), n, \epsilon)$

```
1 min = n
2 repeat k times
3     total = 0
4     repeat s times
5         Choose a random u ∈ V.
6         total = total + degree(u)
7     if total/s < min then min = total/s
8 return min.
```

Disclaimer: This question is supposed to be challenging. The teacher and TAs might not be able to give you a full solution. But they can point you to relevant articles after the deadline.