CS 170 Efficient Algorithms and Intractable Problems
Fall 2017 Prasad Raghavendra and Sanjam Garg

Homework 5

Instructions: You are welcome to form small groups (up to 4 people total) to work through the homework, but you **must** write up all solutions by yourself. List your study partners for homework on the first page, or "none" if you had no partners.

If using LaTeX (which we recommend), you may use the homework template linked on this Piazza post to get started.

Begin each problem on a new page. Clearly label where each problem and subproblem begin. The problems must be submitted in order (all of P1 must be before P2, etc). For questions asking you to give an algorithm, respond in what we will refer to as the *four-part format* for algorithms: main idea, pseudocode, proof of correctness, and running time analysis.

Read the Homework FAQ Piazza post on Piazza before doing the homework for more explanation on the four-part format and other clarifications for our homework expectations.

No late homeworks will be accepted. No exceptions. This is not out of a desire to be harsh, but rather out of fairness to all students in this large course. Out of a total of approximately 12 homework assignments, the lowest two scores will be dropped.

Special Questions:

- *Shortcut questions*: Short questions are usually easy questions that give you opportunities to practice basic materials. However, we understand that some of you are very familiar with the topics and do not want to spend too much time on easy questions. Therefore, we design shortcut questions for this purpose. A shortcut question usually has multiple parts that build upon each other and are ordered by their difficulty level. You can work on those in order or start from wherever you like. However you only need to submit the last part you are able to solve. For example, if a question has 5 parts (a, b, c, d, e), you are confident about part e, you should submit part e without any of the previous four parts. If you are confident about d but not sure about e, you should submit d for grading purposes. Please clearly indicate in your submission which part you are submitting.

- *Redemption questions*: It is important that you carefully read the posted solutions, even for problems you got right. To encourage this, you have the option of submitting a redemption file, a few paragraphs in which you explain, for each problem you choose to cover, what you did wrong and what the right idea was in your own words (not cutting and pasting from the solution!), and appending it to your homework. For example, suppose that as you review your solutions to HW1, you realize you had misunderstood question 3 and answered it incorrectly. You would write down what you just learned, and then submit it in your HW2 assignment the following week. Because these are mainly for your benefit, feel free to format them however is most useful for you.

- *Extra credit questions*: We might have some extra credit questions in the homework for people who really enjoy the materials. However, please note that you should do the extra credit problems only if you really enjoy working on these problems and want an extra challenge. It is likely not the most efficient manner in which to maximize your score.

Due Wednesday, October 11, at 4:59pm

0. Who did you work with?

List all your collaborators on this homework. If you have no collaborators, please list “none”.

1. (★★★ level) Scheduling Homeworks

You have n homeworks to do. Each homework takes one hour to complete. Homework i has a deadline time T_i. You can think of T_i as a positive integer value, and the current time is 0, namely homework i is due in T_i hours. You get S_i points ($S_i \geq 0$) if you finish homework i before its deadline. You can only do one homework at any time. The goal is to maximize the total points you can get.

For each of the following greedy algorithms, either prove that it is correct, or give a simple counterexample (with at most three homeworks) to show that it fails.

(a) Among unscheduled homeworks that can be scheduled on time, consider the one whose deadline is the earliest (breaking ties by choosing the one with the highest points), and schedule it at the earliest available time. Repeat.

(b) Among unscheduled homeworks that can be scheduled on time, consider the one whose points is the highest (breaking ties by choosing the one with the earliest deadline), and schedule it at the earliest available time. Repeat.

(c) Among unscheduled homeworks that can be scheduled on time, consider the one whose points is the highest (breaking ties arbitrarily), and schedule it at the latest available time before its deadline. Repeat.

2. (★★★ level) Graph Coloring

Let $G = (V,E)$ be an undirected graph where every vertex has degree less than 170. Let's find a way of to color each vertex blue or red, so that every vertex has less than 85 neighbors of its own color.

Consider the following algorithm, where we call a vertex *bad* if it has at least 85 neighbors of its own color:

```
1. Color each vertex arbitrarily.
2. Let $B := \{v \in V : v \text{ is bad}\}$.
3. While $B \neq \emptyset$:
4.         Pick any bad vertex $v \in B$.
5.         Reverse the color of $v$.
6.         Update $B$ to reflect this change, so that it again holds the set of bad vertices.
```

Notice that if this algorithm terminates, it is guaranteed to find a coloring with the desired property.

(a) Prove that this algorithm terminates in a finite number of steps.

Hint: Define a function that associates a non-negative integer to each possible way of coloring the graph, in such a way that each iteration of the while-loop is guaranteed to strictly reduce the value of the function.

(b) Prove that the algorithm terminates after at most $|E|$ iterations of the loop.

Hint: You should figure out the largest possible value of the function.

3. (★★ level) 170-Graph

An undirected graph is called a 170-graph if every vertex has degree at least 170. That is, every vertex has at least 170 neighbors. Given an undirected graph $G = (V, E)$, design an algorithm to find a subgraph $G' = (V', E')$ of G that is a 170-graph. We want to make $|V'|$ as large as possible. The running time of your algorithm should be polynomial in $|V|$ and $|E|$.

Hint: There are some vertices you can rule out immediately as not in G'.

(You only need to provide the main idea and running time.)

4. (★★★★ level) Weighted Set Cover

In class (and Chapter 5.4) we looked at a greedy algorithm to solve the *set cover* problem, and proved that if the optimal set cover has size k, then our greedy algorithm will find a set cover of size at most $k \log n$.

Here is a generalization of the set cover problem.

- *Input:* A set of elements B of size n; sets $S_1, \ldots, S_m \subseteq B$; positive weights $w_1, \ldots, w_m$.
- *Output:* A selection of the sets S_i whose union is B.
- *Cost:* The sum of the weights w_i for the sets that were picked.

Here is an algorithm (pseudocode) to find the set cover with approximately the smallest cost:

```
1. While some element of B is not covered
2.     Pick the set $S_i$ with the largest ratio (Number of new elements covered by $S_i$) / $w_i$.
```

Prove that if there is a solution with cost k, then the above algorithm will find a solution with cost $O(k \log n)$.

Hint: You may find the following inequalities useful. For $x \in \mathbb{R}$, $-1 < x < 1$,

$$1 + x \leq e^x \leq 1 + x + x^2.$$

For $x \in \mathbb{R}$, $x > -1$,

$$\frac{x}{1+x} \leq \ln(1+x) \leq x.$$

For $x, y > 0$, $x \leq y \iff \ln x \leq \ln y$; For $x, y \in \mathbb{R}$, $x \leq y \iff e^x \leq e^y$;

5. (★★★★ level) Quaternary Huffman

Quadmedia Disks Inc. has released a quaternary hard disk. Each cell on a disk can now store values 0, 1, 2 or 3 (instead of just 0 or 1). To take advantage of this new technology, provide a modified Huffman algorithm for compressing sequences of characters from an alphabet of size n, where the characters occur with known frequencies $f_1, f_2, \ldots, f_n$. Your algorithm should encode each character with a variable-length codeword over the values 0, 1, 2, 3 such that no codeword is a prefix of another codeword and so as to obtain the maximum possible compression. Your proof of correctness should prove that your algorithm achieves the maximum possible compression.

Please provide the main idea and proof of correctness only. If you think pseudocode can better convey your idea, add it to the *Main Idea*.

If you are stuck on the proof of correctness, please see proof of binary huffman from lecture.

6. (★★★★ level) Simple and Naive Cluster Analysis

A long time ago in a galaxy far, far away... there are magicians and wizards who communicate with gravitational waves. You and your team just intercepted n of their messages, and you are only able to observe that two messages are either very similar or very dissimilar. You decide to start with the following simple and naive clustering analysis: partition these n messages into two groups, breaking up as many pairs of dissimilar messages as possible. There are more and more incoming messages, so your team needs to find two groups quickly, even if this means that it's not the best possible solution. Come up with an efficient algorithm that breaks up at least half the number of pairs of dissimilar messages as the best possible solution, and prove your answer.

Hint: Try assigning messages to group one at a time. Consider how many pairs of dissimilar messages are broken up with each iteration.

7. (??? level) (Optional) Redemption for Homework 4

Submit your *redemption file* for Homework 4 on Gradescope. If you looked at the solutions and took notes on what you learned or what you got wrong, include them in the redemption file.