CS 170 Fall 2017

Efficient Algorithms and Intractable Problems
Prasad Raghavendra and Sanjam Garg

# Homework 3

**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, September 20, 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) Can you win OneSeventy?**

Congratulations! UnElectronic Arts has hired you for Summer 2018. As it turns out, they've been working on a game called OneSeventy for the past 25 years, and the game has a ton of quests. **For a player to win, the player must finish all the quests.** There are a total of $N$ quests in the game. Here's how the game works: The player can *arbitrarily* pick one of the $N$ quests to start from. Once the player completes a quest, they unlock some more quests. The player can then choose one of the unlocked quests and complete it, and so on.

So, for instance, let's say that this game had only 4 quests, $A,B,C,D$.

Let's say that after you complete

- quest $A$, you unlock quests $[B,D]$.
- quest $B$, you unlock quests $[C,D]$.
- quest $C$, you unlock nothing, $[]$.
- quest $D$, you unlock quest $[C]$.

Is this game winnable? Yes, because of the following scenario:

The player picks quest $A$ to start with. At the end of quest $A$, their unlocked list contains $[B,D]$. Say that they choose to do quest $B$, then their unlocked list will contain $[C,D]$. Say that they choose to complete quest $C$, then their unlocked list will contain quest $[D]$. Finally, they finish quest $D$.

Note that if the player had started with quest $C$ instead of quest $A$, they would have lost this game, because they wouldn't have unlocked any quests and would be stuck. *But the game is still winnable, because there is **a** starting quest which makes the player win.*

OneSeventy has $N$ quests, enumerated as $\{n_1, n_2, \ldots, n_N\}$. Each quest $n_i$ unlocks $k_i$ quests. So in the example, if $n_1 = A$, then $k_1 = 2$. Let $K = \sum_{i=1}^{n} k_i$.

UnElectronic Arts' concern is: is their game with $N$ quests winnable? You are provided with information about the quests in a similar manner as the example above. Design an algorithm to find out whether or not the game is winnable. The algorithm should run asymptotically faster than $O(NK)$. You must provide the main idea and runtime for your algorithm.

**2. (★★ level) Dijkstra's Durability**

For the following questions, answer yes or no and provide justification. Consider a directed graph $G = (V, E)$ with positive edge lengths, and two vertices $s$ and $t$. We want to find the shortest path from $s$ to $t$ in $G$, so we run Dijkstra's algorithm. Now:

(a) We also want to find the shortest path from $s$ to another node $t'$. Do we need to run Dijkstra's algorithm again?

(b) Suppose we add a positive number $k$ to all edge lengths. Is the shortest path from $s$ to $t$ always the same as before?

(c) Suppose we subtract a positive number $k$ from all edge lengths. Is the shortest path from $s$ to $t$ always the same as before?

(d) Suppose we multiply all edge lengths by a positive number $k$. Is the shortest path from $s$ to $t$ always the same as before?

**3. (★★★ level) Graph Basics**

(a) An undirected graph $G$ is called *bipartite* if we can separate its vertices into two subsets $A$ and $B$ such that every edge in $G$ must cross between $A$ and $B$. Show that a graph is bipartite if and only if it has no odd cycles.

*Hint: Consider a spanning tree of the graph, which is a subset of the graph's edges which allow it to be a tree on all of its vertices.*

(b) A directed acyclic graph $G$ is *semiconnected* if for any vertices $A$ and $B$ there is either a path from $A$ to $B$ or a path from $B$ to $A$. Show that $G$ is semiconnected if and only if there is a directed path that visits all of the vertices of $G$.

**4. (★★★ level) Count Shortest Paths**

Given a directed graph $G = (V, E)$ with positive edge lengths, and two vertices $s$ and $t$. The shortest paths from $s$ and $t$ are not always unique: there could be two or more different paths with the same minimum length. Design an algorithm to find the number of shortest paths from $s$ to $t$. Your algorithm should be as efficient as possible.

*(Please turn in a four part solution to this problem.)*

*Hint: Try to count the number of shortest paths from s to every other vertex.*

**5. (★★★★★ level) Premium Member**

There is a set $V$ of cities connected by flights operated by Algorithmic Airline (AA), and each flight has a cost. Let $H \subseteq V$ be a set of AA hubs, and AA would like to encourage its members to choose flights landing at these hubs. If an AA member has landed in these hubs for a total of $k$ times, he/she will become a premium member. Prof. Garg just became an AA member and will start traveling. He is currently in Berkeley and the only way he will take to travel between cities is taking AA flights. What is the minimum cost for him to become a premium member?

Formally, given a directed graph $G = (V, E)$ with positive edge lengths, a non-empty set $H \subseteq V$, a positive integer $k$, and a starting vertex $s$. Design an algorithm with running time polynomial in $|V|$, $|E|$, and $k$ to find the shortest path starting at $s$ and containing at least $k$ vertices in $H$ (with cycles permitted). If there doesn't exist such a path, your algorithm should output $\infty$.

*(Please turn in a four part solution to this problem.)*

*Hint: Try to construct a new graph, and try to solve the case $k = 2$.*

**6. (★★★★★ level) Alternate Universe Theory**

It is the year 3050, and humankind is divided into three – the magicians, the ordinary, and you. The ordinary and the magicians don't interact with each other, but you are special. You can interact and use the transport systems of both, the ordinary and the magicians. Here is the idea: the magicians can transport you across different universes. Sometimes, they charge you money for this service, but sometimes they give you money instead. A magician can only teleport you one-way. Also, it so happens that if a magician teleports you from $u$ to $v$, then there is **no way for you to go back** from $v$ to $u$ no matter how you travel. In other words, if there is an edge from $u$ to $v$ (where $u$, $v$ are in different universes), then there is no path of edges that leads from $v$ to $u$.

Now, at each separate universe, exist the ordinary folk and taxi-stops. They drive taxis, charge you some money and transport you from one taxi-stop to another. Taxis drive both-ways, so if you get to stop $a$ from $b$, you can just take a taxi back from $b$ to $a$. Some of these taxi-stops are **also** magician-stops, where you can find magicians to get teleported by.

In summary, taxis drive from taxi-stop to taxi-stop. Some taxi-stops may also be magician-stops. Magicians are found at magician-stops, and they teleport you from one universe's magician-stop to another universe's taxi-stop.

Formally, our graph looks like $G = (V, T \cup U)$ where $V$ is the set of all stops (taxi or magician), $T$ is the set of edges where taxis can drive (between taxi-stops) and $U$ is the set of universe-teleportation edges, i.e., if $(x,y) \in U$, then there is a magician that will take you from universe $x$ to universe $y$.

An edge in $T$ is specified as an **undirected** edge $(v_1, v_2, M_{v_1,v_2})$, which means that there is a taxi that will take you from $v_1$ to $v_2$ and charge you $M_{v_1,v_2}$. Note that the cost of going from $(v_2) to (v_1)$ need not be the same as $M_{v_1,v_2}$.

An edge in $U$ is specified as a **directed** edge $(u_1, u_2, M_{u_1,u_2})$. Remember that magicians could give you money to use their services.

Your goal is to find the cheapest path from some $s_1 \in V$ to some $s_2 \in V$ in time $O((|V| + |E|) \log |V|)$ where $E = T \cup U$. A four part solution is required for this problem.

*(Please turn in a four part solution to this problem.)*

*Hint: Think about the fact that if there is an edge from u to v, where $u, v$ are in different universes, then there is no path back from v to u. What does this mean about each universe?*

*Hint 2: If you are stuck, try thinking about the following questions (no credit for answering).*

- *Let there be a directed graph, where there are some negative edges. These negative edges are all of the form $(s, t_i)$ for some vertex s, i.e., all these edges go out of s. If Dijkstra's algorithm is run starting at s, will it work correctly?*
- *Now consider the set up from the previous hint, i.e., all the negative edges leave some vertex s. Say that your starting node is $v_1$. Find an algorithm to compute the shortest path from $v_1$ to $v_2$. The runtime of this efficient algorithm should be the same as that of good old Dijkstra's.*

**7. (??? level) (Optional) Redemption for Homework 2**

Submit your *redemption file* for Homework 2 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.