

Homework 3

CS 4104 (Spring 2024)

Assigned on February 21, 2024.
Submit PDF solutions on Canvas
by 11:59pm on February 28, 2024.

Instructions:

- The Honor Code applies to this homework with the following exception:
 - You can pair up with another student to solve the homework. Please form teams yourselves. Of course, you can ask the instructor for help if you cannot find a team-mate. You may choose to work alone.
 - You are allowed to discuss possible algorithms and bounce ideas with your team-mate. **Do not discuss proofs of correctness or running time in detail with your team-mate. You must write down your solution individually and independently. Do not send a written solution to your team-mate for any reason whatsoever.**
 - *In your solution, write down the name of the other member in your team. If you do not have a team-mate, please say so.*
 - Apart from your team-mate, you are not allowed to consult sources other than your textbook, the slides on the course web page, your own class notes, the TAs, and the instructor. In particular, do not use a search engine or any other online sources including ChatGPT or generative AI software of that ilk.
- **Do not forget to typeset your solutions.** *Every mathematical expression must be typeset as a mathematical expression, e.g., the square of n must appear as n^2 and not as “ n^2 ”. You can use the L^AT_EX version of the homework problems to start entering your solutions.*
- Do not make any assumptions not stated in the problem. If you do make any assumptions, state them clearly, and explain why the assumption does not decrease the generality of your solution.
- You must also provide a clear proof that your solution is correct (or a counter-example, where applicable). Type out all the statements you need to complete your proof. *You must convince us that you can write out the complete proof. You will lose points if you work out some details of the proof in your head but do not type them out in your solution.*
- If you are proposing an algorithm as the solution to a problem, keep the following in mind (the strategies are based on mistakes made by students over the years):
 - Describe your algorithms as clearly as possible. The style used in the book is fine, as long as your description is not ambiguous. Explain your algorithm in words. A step-wise description is fine. *However, if you submit detailed code or pseudo-code without an explanation, we will not grade your solutions.*
 - Do not describe your algorithms only for a specific example you may have worked out.
 - Make sure to state and prove the running time of your algorithm. You will only get partial credit if your analysis is not tight, i.e., if the bound you prove for your algorithm is not the best upper bound possible.
 - You will get partial credit if your algorithm is not the most efficient one that is possible to develop for the problem.

- In general for a graph problem, you may assume that the graph is stored in an adjacency list and that the input size is $m + n$, where n is the number of nodes and m is the number of edges in the graph. Therefore, a linear time graph algorithm will run in $O(m + n)$ time.

My team-mate is -----

Problem 1 (20 points) Solve exercise 5 in Chapter 4 (pages 190–191) of “Algorithm Design” by Kleinberg and Tardos. Let’s consider a long, quiet country road with house scattered very sparsely along it. (We can picture the road as a long line segment, with an eastern endpoint and a western endpoint.) Further, let’s suppose that despite the bucolic setting, the residents of all these houses are avid cell phone users. You want to place cell phone base stations at certain points along the road, so that every house is within four miles of one of the base stations.

Give an efficient algorithm that achieves this goal, using as few base stations as possible. Just in case the problem statement is not completely clear, you can assume that the road is the x -axis, that each house lies directly on the road, and that the position of each house can be specified by its x -coordinate.

Hint: Sort the houses in increasing order of x -coordinate. Develop your algorithm now. To design your proof of correctness, follow one of the proof strategies we discussed in class (and present in the textbook) for greedy scheduling problems.

Problem 2 (30 points) Solve exercise 13 in Chapter 4 (pages 194–195) of your textbook. A small business—say, a photocopying service with a single large machine—faces the following scheduling problem. Each morning, they get a set of jobs from customers. They want to do the jobs on their single machine in an order that keeps the customers happiest. Customer i ’s job will take t_i time to complete. Given a schedule, i.e., an ordering of jobs, let C_i denote the finishing time of job i . For example, if job j is the first to be done, we would have $C_j = t_j$, and if job j is done right after job i , then $C_j = C_i + t_j$. Each customer i also has a given weight w_i that represents his or her importance to the business. The happiness of customer i is expected to be dependent on the finishing time of i ’s job. So the company decides to order the jobs to minimize the weighted sum of completion times, $\sum_{i=1}^n w_i C_i$.

Design an efficient algorithm to solve this problem, i.e., you are given a set of n jobs with a processing time t_i and weight w_i for each job. You want to order the jobs so as to minimize the weighted sum of completion times, $\sum_{i=1}^n w_i C_i$.

Hint: Try to use one of the techniques we have seen for proving the correctness of greedy algorithms. Working “backwards” from what you need to prove might help you to discover the algorithm. Note that the completion times are not part of the input. The algorithm has to compute them. So you cannot sort the jobs by completion time.

Problem 3 (20 points) Consider the version of Dijkstra’s algorithm shown below written by someone with access to a priority queue data structure that supports *only* the INSERT and EXTRACTMIN operations. Due to this constraint, the difference between this version and the one discussed in class is that instead of the CHANGEKEY($Q, x, d'(x)$) operation in Step 8, this version simply inserts the pair $(x, d'(x))$ into Q . The danger with this algorithm is that a node x may occur several times in Q with different values of $d'(x)$. Answer the following questions.

1. (6 points) When the algorithm inserts a pair (x, d_1) into Q , suppose the pair (x, d_2) is already in Q . What is the relationship between d_1 and d_2 ?
2. (8 points) Using this relationship, how will you fix this algorithm? You just have to describe your correction in words, e.g., by saying “I will add the following command after Step X: ...” You do not have to prove the correctness of your algorithm.
3. (6 points) What is the running time of this modified algorithm? Just state the bound in terms of the number of nodes n and the number of edges m in G .

Algorithm 1 DIJKSTRA'S ALGORITHM(G, l, s)

```
1: INSERT( $Q, s, 0$ ).
2: while  $S \neq V$  do
3:    $(v, d'(v)) = \text{EXTRACTMIN}(Q)$ 
4:   Add  $v$  to  $S$  and set  $d(v) = d'(v)$ 
5:   for every node  $x \in V - S$  such that  $(v, x)$  is an edge in  $G$  do
6:     if  $d(v) + l(v, x) < d'(x)$  then
7:        $d'(x) = d(v) + l(v, x)$ 
8:       INSERT( $Q, x, d'(x)$ )
9:     end if
10:  end for
11: end while
```

Problem 4 (30 points) The Flint Water Study (<https://flintwaterstudy.org/>) was a group of Virginia Tech researchers who helped resolve scientific uncertainties regarding drinking water in the City of Flint, MI. Inspired by this group, you start thinking about using multiple sensors embedded in different parts of a city to automatically test water quality. Sensors can communicate with each other. However, each sensor has limited resources (e.g., it can contact only “nearby” sensors) and these communications can be unreliable and asymmetric. The goal of each sensor is to relay the information it has collected back to a “master” sensor. After deploying the sensors, you can measure which pairs can communicate and how reliable each link is. Your goal is to determine the maximum number of sensors that can communicate with the master through a reliable series of links.

Fortunately, you have taken CS 4104! You model the problem as follows. You have a directed graph $G = (V, E)$. Each node in V corresponds to a sensor. The master sensor is a specific node s in V . Every edge $e = (u, v)$ directed from node u to node v has a reliability $r(e)$ that is positive and at most 1, i.e., $0 \leq r(e) \leq 1$. Since communications are asymmetric, it is not necessary that $r(u, v) = r(v, u)$. The reliability of a path is the product of the reliabilities of the edges in that path. Given such a graph G and a parameter $q > 0$, the **Sensor Network** problem is to compute the largest set S of nodes in V such that for every node u in S there is a directed path from u to s with reliability at least q . Describe an algorithm for the **Sensor Network** problem.

Note: I have used the problem of placing sensors to inspire this question. You should develop your solution using the language of graphs. In other words, you should be able to present your solution even if you read only the second paragraph of this question.

Hint: You may develop your own algorithm or modify an algorithm we have designed in class. In either case, you must not forget to prove the correctness of your algorithm, as we always require in this class. An alternative you can use is to transform the *input* to the **Sensor Network** problem so that it becomes a suitable input for a different problem X we have studied in class. In this case, you can use an algorithm A for problem X on this transformed input, take the output of algorithm A , and then convert it to a solution to the **Sensor Network** problem. If you use this option, you do not have to prove the correctness of algorithm A (since we have done so in class) but you must show that your transformations yield the correct answer to the **Sensor Network** problem.