

CS5050 Homework 3, Chapters 7

Due on date specified by Eagle (due at class time) 10 points

Working with others is **STRONGLY ENCOURAGED**. Prepare your answers individually and then compare/contrast your answers with others. The homework questions are meant to help you prepare for exams. When you work with others, you are motivated to consider every aspect of a problem and do your best thinking. Just doing a google search for the answers is **NOT** recommended. The learning comes as you struggle to create the answer from what you already know or from material in the chapters.

Note, these exercises may be done in groups of one, two, or three. If more than one person is involved, list all the names on ONE set of answers. Groups may change throughout the semester. Answers should not be compared with others not in your group.

These are typed (or hand-written neatly) exercises, but you are certainly encouraged to actually code the programming segments if you have time.

In giving your answers, use algorithmic language, building on algorithms and terminology used in class. In previous assignments, the term “efficient” was used as code for “polynomial”. In this and future assignments, efficient will mean the best of the polynomial time algorithms.

Be careful about your algorithms. The algorithm should be specific enough that you could turn it over to an undergrad and recognize the solution he/she implements as the same algorithm you specified. I see some students who do the following. When asked to determine cross edges (like on the last assignment), they say “If the arc goes to a node which is not an ancestor or descendant, mark it as a cross edge.” It is almost like saying, “If the arc is a cross edge, mark it as a cross edge.” You must specify the algorithm for determining cross edge, not assume the information you need is already known.

Use good variable names. When specifying an algorithm, explain what comes in and what comes out.

When writing pseudo-code, remember that less is more. Do not write a complete program with variable and class declarations. Explain the algorithm intuitively in English, and then give a high-level pseudo-code description. The level of detail should be just enough that a competent programmer could take your explanation and implement it without undue ambiguity. You may assume that simple data structures have been provided for you. So, it is clearer to say "Append x to the end of list L" rather than giving low level code and pointer manipulation.

In some cases you are just to recognize, “This is merely an instance of a shortest path problem.” In other cases, the problem is similar to a well know problem, but the differences are critical. You must comment how the differences from the well-known problem change the solution. In still other cases, you are asked to improve on the solution discussed in class.

1. Give an example of a weighted, directed graph with negative weight edges (but not negative cycles) which causes Dijkstra's algorithm to run incorrectly.
2. Give reasoning which proves that if all weights in a connected graph are unique, then there is exactly one minimum spanning tree. **Hint: an example is NOT sufficient.**
3. Design an efficient algorithm for finding the longest directed path from vertex s to vertex t on an acyclic directed weighted graph. Analyze the time complexity of your algorithm.
4. NASA wants to link n stations spread over the country using communication channels. Each pair of stations has a different band width available. NASA wants to select $n-1$ channels (the minimum possible) in such a way that all the stations are linked by the channels and the total bandwidth (defined as the sum of the individual bandwidths) is maximum. Give an efficient algorithm to solve this problem. **Hint: nodes are stations, and edges are channels weighted by bandwidth. You need to specify which edges you want to select in order to connect the stations via your selected edges.**
5. You are given a timetable which consists of a set of airports (name and minimum connecting time) and a set of flight information (origin airport, destination airport, departure time, arrival time). Describe an efficient algorithm for determining the quickest way to leave airport A after time T and arrive at airport B.
6. You are in a maze. Along each corridor of the maze there is a bag of gold coins. The amount of gold in each bag varies. You will be given an opportunity to walk through the maze, picking up bags of gold as you go. You may not retrace your steps. You must enter only through the door marked ENTER and exit through the door marked EXIT. Each corridor is one-way, and you may only go down the corridor only in the given direction. There is no way to traverse a loop in the maze. You will receive a map of the maze including the amount of gold and direction of each corridor. Describe an algorithm to help you pick up the most gold.
7. Show how to modify Baruvka's algorithm so that the worst-case time is $O(n^2)$. **Hint: Baruvka's original algorithm is $O(m \log n)$. This bound comes from the fact that you do a traversal of all edges each time you find a collection of edges to add. Since the number of clusters divides in half each time, you have $\log n$ passes. In the worst case, $m=n^2$ (as there are edges between every pair of nodes). Thus, in the worst case, Baruvka's runs in $O(n^2 \log n)$. Thus, this problem asks you to come up with a NEW way of thinking about the problem which takes advantage of the fact that there are lots of edges. Anytime you change complexity classes, you must think about the problem in a different way. If you do the same thing, only twice as fast, you have only changed the complexity by a constant. That doesn't change the complexity class. You need to do things differently in order to change the complexity. When you are trying to stay within some bound, you know that any preprocessing you want to do must be under the bound. Since they are calling it an improvement to Baruvka's algorithm it would have to keep the spirit of Baruvka – rather than solve the problem a completely different way.**