Homework 3

CS 4104 (Spring 2017)

Assigned on Wednesday, February 15, 2017.
Submit a PDF file containing your solutions on Canvas by the beginning of class on Monday, February 27, 2017. Yes, you have 12 days to work on this homework assignment.

Instructions:

- The Honor Code applies to this assignment as follows:
  - You can pair up with another student to solve the homework. Please form teams yourselves. Of course, you can ask me 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 indepedently. 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 any 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.
- 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 LaTeX version of the homework problems to start entering your solutions.
- 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 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.
- Do not describe your algorithms only for a specific example you may have worked out.
- 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.*
- Describe an analysis of your algorithm and state and prove the running time. 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.

Problem 1 (7 points) The first two greedy algorithms we discussed in class were `Interval Scheduling` and `Interval Partitioning`. In this problem, we are going to examine whether we can use first algorithm to solve the second.

Let S be the set of jobs that are input to `Interval Partitioning`. Recall that our goal for this problem is to partition these jobs into k sets so that

(a) each job is in exactly one set,

(b) all the jobs in every set are mutually compatible with each other (their time intervals do not overlap), and

(c) k is the smallest possible over all partitions.

Our goal for `Interval Scheduling` was quite different. We had to compute the largest set of mutually compatible jobs. We developed the Earliest-Finish-Time (EFT) algorithm to solve this problem optimally.

The idea of using the algorithm for the first problem to solve the second is simple:

1. Run the EFT algorithm on S.
2. Assign the jobs output by this algorithm from S to one set, output this set, and delete these jobs from S.
3. Repeat the first two steps until S is empty.

The intuition behind this algorithm is that after every execution of the EFT algorithm, the depth of the remaining jobs must decrease by 1. Therefore, this algorithm must stop after d iterations, where d is the depth of S.

Unfortunately, this intuition is incorrect. Your task is to come up with a counterexample, i.e., construct an input of jobs for which this algorithm outputs more sets than the depth. Briefly explain why your input has this property. *Hint:* Do not provide a complicated construction. Four jobs should suffice.

Problem 2 (13 points) In class, we stated that the running time of the depth-based algorithm for `Interval Partitioning` was $O(n \log n)$ (where n is the number of input jobs) but we did not clearly describe how to implement it in that time. You will rectify this situation here. I repeat the algorithm here:

Sort the intervals by their start times
Let $I_1, I_2, \ldots, I_n$ denote the n intervals in this order.
For $j = 1, 2, \ldots, n$
  For each interval I_i that precedes I_j in sorted order and overlaps it
    Exclude the label of I_i from consideration for I_j
  Endfor
  Assign a non-excluded label to I_J

Note that I have not included the "else" clause since we proved in class that it is not necessary.

The critical lines are in blue. Describe how you will implement these lines so that they take $O(n \log n)$ time to execute over the course of the algorithm. Your algorithm need not explicitly exclude labels from consideration. All it needs to do is to find a label correctly for each job.

Problem 3 (35 points) Solve exercise 5 in Chapter 4 (pages 190–191) of your textbook. 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.

Given 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. You can also assume that the houses are sorted in increasing order of x-coordinate.

Problem 4 (45 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.