

Problem Set 1

CS 331H

Due Tuesday, February 14

General rules:

- For full credit, you must give proofs of every step.
- Either typeset your solutions or write in clearly legible handwriting. The TA *will* dock points for bad handwriting.
- Collaboration is encouraged, but you must write up the solutions on your own and acknowledge your collaborators at the top of your solutions.

1. Recursive time bounds: give a big-O bound for $T(n)$ given each of the following recursive formulas:

- (a) $T(n) = 3T(n/4) + n \log n$
- (b) $T(n) = 2T(n/2) + \sqrt{n}$
- (c) $T(n) = 5T(n/4) + n$
- (d) $T(n) = T(2n/3) + T(n/3) + n/6$.

with the base case $T(1) = 1$.

2. In class we discussed interval *packing* problems. Here we explore interval *cover* problems.
 - (a) You are given a set of n intervals $[s_i, f_i)$ and a range $[0, T)$. You would like to find a minimal set $I \subset [n]$ of intervals whose union covers the range. That is, we say that I is a *valid cover* if, for all $t \in [0, T)$, there exists an $i \in I$ such that $t \in [s_i, f_i)$. Give a greedy algorithm to compute a valid cover with the smallest number of intervals, in linear time after sorting.
 - (b) Now suppose that each interval also has a *cost* c_i , and your goal is to find a valid cover I minimizing the total cost $\sum_{i \in I} c_i$. Give a dynamic programming solution to this problem that takes $O(n \log n)$ time.

3. Suppose that you have n jobs that you would like to schedule. Each job takes a different duration of time $d_i > 0$ to complete, and a different “urgency” $u_i > 0$. You can only work on one job at a time, but you can choose an arbitrary order among the jobs.

For a given order of the jobs, let t_i be the time that you finish job i , which is the sum of the durations of the previous jobs and this one. Your total cost of a given order is defined as $\sum_{i=1}^n u_i t_i$: the more urgent a job is, the more important it is that it be finished earlier. Your goal is to find the job order that minimizes the cost.

- (a) (No response necessary) Think about this problem on your own for 10 minutes before reading the spoilers below.
 - (b) Suppose that $n = 2$. What order should you take?
 - (c) Consider any ordering among the jobs for general n , and look at any pair of adjacent jobs in that ordering. How would the total cost change if you swap the ordering?
 - (d) Observe that repeatedly applying the idea in the previous part would lead to an $O(n^2)$ time bubble sort of the jobs, based on some function $f(u_i, d_i)$ of each element.
 - (e) Give an $O(n \log n)$ time algorithm for the problem.
4. There’s a Jupyter Notebook linked from the class webpage. Run through it, then answer the questions at the end. Don’t wait till the last day to do this: setting up the required libraries may take some time.