University of Washington January 27, 2023
Department of Computer Science and Engineering
CSE 417, Winter 2023

Homework 4, Due Friday February 3, 11:59 PM, 2023

Turn in instructions: Submit a PDF on Gradescope with each problem on a separate page.

Problem 1 (10 points):

Let S be a set of intervals, where $S = \{I_1, \ldots, I_n\}$ with $I_j = (s_j, f_j)$ and $s_j < f_j$. A set of points $P = \{p_1, \ldots, p_k\}$ is said to be a *cover* for S if every interval of S includes at least one point of P, or more formally: for every I_i in S, there is a p_j in P with $s_i \leq p_j \leq f_i$.

Describe an algorithm that finds a cover for S that is as small as possible. Argue that your algorithm finds a minimum size cover. You algorithm should be efficient. In this case $O(n \log n)$ is achievable but it is okay if your algorithm is $O(n^2)$. You may assume that the intervals are sorted in order of finishing time.

Problem 2 (10 points):

The paragraphing problem is: Given a set of words $w_1, \ldots, w_n$ with word lengths $l_1, \ldots, l_n$, break the words into consecutive groups, such that the sum of the lengths of the words in each group is less than a fixed value K. (We will ignore the issue of putting spaces between words or hyphenation; these are minor details.) The words remain in the original order, so the task is just to insert line breaks to ensure that each line is less than length K.

Describe a greedy algorithm for paragraphing that attempts pack in as many words as possible into each line, e.g., to put words into a line one at a time until the length bound K is reached, and break the line before the word w_r that caused the the bound to be exceeded.

Is your algorithm optimal, in the sense that it minimizes the total number of lines of output? Why or why not. If you think it is optimal, given an explanation of why (we will be looking for the general idea as opposed to a formal proof.) If it is not, give a counter example.

Problem 3 (10 points):

Here is another version of the homework scheduling problem with partial credit. Suppose that you have a collection of homework assignments $\{H_1, \ldots, H_k\}$. Assignment H_j has a time requirement t_j and a value p_j. If you spend less time on an assignment than required, you will get partial credit that it proportional to the time spent on it. So if you spend time t on assignment H_j, where $0 \leq t \leq t_j$ you will received $\frac{t}{t_j} p_j$ points.

You have total time T available for homework, and, unfortunately, $T < \sum_j t_j$. You want to maximize the points for the assignments that you either complete or get partial credit on, so you need to come up with an algorithm for allocating your time on the assignments.

Argue that there is an optimal solution where only one assignment gets partial credit.

Describe an algorithm that finds an optimal solution to the problem, which maximizes the number of points you receive on homework, subject to the constraint that the time spent is at most T. Give a justification as to why your algorithm finds an optimal solution. You should also give the run time for your algorithm.

Note: For this problem, it is critical that partial credit is allowed, as otherwise it is NP-Complete. More on that later in the course.

Programming Problem 4 (10 points):

Implement the greedy algorithm for graph coloring discussed in class (Lecture 9, Slide 10, Coloring Algorithm version 2). Run the algorithm on random graphs. Use values of n of 1000 (or larger). You should report results for values of p in the range 0.002 and 0.02. How many colors are needed on the average? Since you are generating random graphs, taking several graphs with the same value of p will give more interesting results. Averaging over 10 graphs (per value of p) is probably sufficient.

Programming Problem 5 (10 points):

Create two new version of your graph coloring algorithm from Problem 4 that process the vertices in order of increasing vertex degree, and in order of decreasing vertex degree. Compare your results with the maximum degree of the graph, along with the algorithm from Problem 4.. You should report results for values of p in the range 0.002 and 0.02. How many colors are used on the average.