CS4102 - ALGORITHMS - SPRING 2022

DUE FRIDAY, APRIL 15, 2022 AT **11:30 pm** VIA GRADESCOPE C2BASIC(RTG5XKH)

Collaboration Policy: You are encouraged to collaborate with up to 3 other students, but all work submitted must be your own *independently* written solution. List the computing ids of all of your collaborators in the `collabs` command at the top of the tex file. Do not share written notes, documents (including Google docs, Overleaf docs, discussion notes, PDFs), or code. Do not seek published or online solutions for any assignments. If you use any published or online resources (which may not include solutions) when completing this assignment, be sure to cite by naming the book etc. or listing a website's URL. Do not submit a solution that you are unable to explain orally to a member of the course staff. Any solutions that share similar text/code will be considered in breach of this policy. Please refer to the syllabus for a complete description of the collaboration policy.

Collaborators:

Sources: Cormen, et al, Introduction to Algorithms.

PROBLEM 1 *Load Balancing*

You work for a print shop with 4 printers. Each printer i has a queue with n jobs: $j_{i,1}, \ldots, j_{i,n}$. Each job has a number of pages, $p(j_{i,m})$. A printer's workload $W_i = \sum_\ell p(j_{i,\ell})$ is the sum of all pages across jobs for for that printer. Your goal is to *equalize* the workload across all 4 printers so that they all print the same number of total pages. You may only remove jobs from the end of their queues, i.e., job $j_{i,n}$ must be removed before job $j_{i,n-1}$, and you are allowed to remove a different number of jobs from each printer. Give a **greedy algorithm** to determine the maximum equalized workload (possibly 0 pages) across all printers. Be sure to state your greedy choice property.

Solution: This algorithm will repeatedly select which printer to remove a job from until the Workload of all 4 printers is the same. If we ever reach a state where $W_1 == W_2 == W_3 == W_4$, we will return this workload. Otherwise, we will choose which job to remove next in the following greedy manner:
Remove a job from printer i such that:
$W_i - p(j_i, n) ==$ max($W_1 - p(j_1, n)$, $W_2 - p(j_2, n)$, $W_3 - p(j_3, n)$, $W_4 - p(j_4, n)$)
That is, we will consider what would happen if we removed a job from *all 4* printers, and determine which printer would have the maximum resulting workload of the 4 if we did this. Whichever one would have the maximum in this situation is the printer we will choose to actually remove a job from. If there is a tie, it doesn't matter which one we choose to remove one so simply remove whichever one is first in the order.
This greedy choice strategy will be repeated until we reach a result (which may be 0).

PROBLEM 2 *Short Answer Questions. (You don't have to explain your answers in your submission, but you should understand the reason behind your answer.)*

A. True or false? Issuing the largest coin first will always solve the *coin change problem* if only two coins are available: the penny and one larger coin. Assume the amount of change is $\geq$ the larger coin.
Solution: True.

B. True or false? The *interval scheduling problem* is always guaranteed to have an optimal solution that contains the interval with the earliest finish time.
Solution: True.
The greedy strategy that we proved to be correct adds the interval that ends earliest, adds it to the solution, and removes all intervals that conflict with it. Thus at the first choice it will always add the interval with the overall earliest finish time.

C. Choose one: In our proof of the correctness of the greedy solution to the *interval scheduling problem*, we exchanged the interval i selected by our greedy choice with another interval that finished earlier/later than interval i. (Your answer is one of "earlier" or "later.")
Solution: later
i is the earliest finishing interval left so any other interval must finish later.

D. True or false? A *feasible solution* for the *Huffman encoding problem* is any valid prefix-free code-word table T.
Solution: True
T described above is a feasible solution, but it may not be optimal.

PROBLEM 3 *Optimal Substructure*

Please answer the following questions related to *Optimal Substructure*.

A. What's the difference in how dynamic programming algorithms versus greedy algorithms use *optimal substructure*?

Solution: Dynamic programming uses optimal substructure to make use of solutions to overlapping subproblems using memoization.
On the other hand, greedy algorithms use optimal substructure simply to take advantage of the fact that solutions to subproblems will be part of the overall solution. This property ensures that a series of greedy choices that provide correct solutions to subproblems will result in the solution to the overall problem.

B. Why did we need to prove the optimal substructure for our greedy Huffman coding algorithm?

Solution: Because we needed to be sure that, by making a series of greedy choices to construct a parts of the Huffman encoding tree, the resulting tree would represent an optimal encoding.

PROBLEM 4 *Gradescope Submission*

Submit a version of this `.tex` file to Gradescope with your solutions added, along with the compiled PDF. You should only submit your `.pdf` and `.tex` files.