CS4102 - ALGORITHMS - SPRING 2022

DUE FRIDAY, APRIL 15, 2022 AT **11:30 pm** VIA GRADESCOPE

C2BASIC(MST3K)

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: list collaborators's computing IDs

Sources: Cormen, et al, Introduction to Algorithms. *(add others here)*

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: Compute the initial workloads of all of the printers. While they are not equalized, find the printer with the **highest total load** and remove the last job in its queue. Repeat until the workloads are all equal, or all queues are empty.

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. We can't go wrong here. No chance to get fooled by an strange-valued coin. Consider the proof-by-cases done in class. The proof done for nickels there applies to this problem.

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. Our greedy algorithm works by choosing that item first, and we proved that it will find an optimal solution.

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. (Our greedy choice picked the one that finished earliest, so any other must have a later finished time.

D. True or false? A *feasible solution* for the *Huffman encoding problem* is any valid prefix-free code-word table T.
Solution: True. A feasible solution meets the problem criteria, but may not be the most optimal solution based on the objective function, which here is the smallest sum of character-frequency multiplied by code-length.

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: DP let us choose the best of multiple smaller subproblems (i.e., best last cut for log cutting); greedy requires us to only look at and choose one, and we must make that choice the same each time.

B. Why did we need to prove the optimal substructure for our greedy Huffman coding algorithm?
Solution: We needed to know that an optimal solution to the problem with σ instead of the two other characters would still produce an optimal solution for the problem that did not have σ.

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.