CMPSC 465 Data Structures & Algorithms
Spring 2022 Chunhao Wang and Mingfu Shao

Assignment 10

Due: Friday 11:59 pm, April 8, 2022

Instructions: You may work in groups of up to three people to solve the homework. You must write your own solutions and explicitly acknowledge everyone whom you have worked with or who has given you any significant ideas about your solutions. You may also use books or online resources to help solve homework problems. All consulted references must be acknowledged. The acknowledgements need to be made by answering Problem 1 below.

You are encouraged to solve the problem sets on your own using only the textbook and lecture notes as a reference. This will give you the best chance of doing well on the exams. Relying too much on the help of group members or on online resources will hinder your performance on the exams.

Submissions being late in 2 hours will be accepted with a 20% penalty. Submissions late more than 2 hours will receive 0. There will be no exceptions to this policy, as we post the solutions soon after the deadline.

For the full policy on assignments, please consult the syllabus.

Formatting: Start a new page for each problem.

Describing an Algorithm: Please make sure you use plain wording to explain your algorithm. It is always a good practice to start with a summary of the high-level idea of your algorithm to ease graders understand your solution quickly. Then, explain your algorithm, using plain wording and including enough details.

The use of pseudo-code is optional, and it is your decision. No matter you use it or not, above description in words is always required. The pseudo-code has its own advantage in explaining structured (i.e., if-else, for-loop, recursive functions, etc) algorithms and in putting details in the right place. If you think pseudo-code better explains your algorithm, and/or helps graders understand your solution, and/or contains more details not included in the plain-wording description, then use pseudo-code. If you think everything is already clearly explained in the description with words, then you don't need to include pseudo-code. An algorithm that is only written in pseudo-code (i.e., missing above plain-wording description) is not acceptable, as it is extremely hard to read just pseudo-code without any explanation.

Here is a general situation that may help you decide whether to use pseudo-code or not. An algorithm could be "designed from scratch", i.e., you will need to come up with the step-by-step procedure. This usually involves in implementing a function with clear input and output. In this case, including pseudo-code usually helps. All algorithm we've seen so far (e.g., merge-two-sorted-arrays, merge-sort, etc) falls in this category. Second, an algorithm could also be "transformed into another algorithm", i.e., you use an existing algorithm to solve this problem. In this case you usually don't need to include pseudo-code but to describe how to transform one problem into the other. We will see such examples soon.

0. (0 pts.) Acknowledgements. The assignment will receive a 0 if this question is not answered.

1. If you worked in a group, list the members of the group. Otherwise, write "I did not work in a group."
2. If you received significant ideas about your solutions from anyone not in your group, list their names here. Otherwise, write "I did not consult anyone except my group members".
3. List any resources besides the course material that you consulted in order to solve the material. If you did not consult anything, write "I did not consult any non-class materials."

1. (10 pts.) Unit-time task scheduling

Recall the unit-time task scheduling problem covered in the class. Let $S = \{a_1, \ldots, a_n\}$ be a set of n unit-time tasks, i.e., each task takes a unit time to complete. Let $d_1, \ldots, d_n$ be the corresponding deadlines for the tasks and $w_1, \ldots, w_n$ be the corresponding penalties if you don't complete task a_i by d_i. Note that $1 \leq d_i \leq n$, and $w_i > 0$ for all i. The goal is to find a schedule (i.e., a permutation of tasks) that minimized the penalties incurred.

Recall that we can model this problem as a matroid maximum independent subset problem. Consider the matroid $M = (S, \mathcal{I})$, where $S = \{a_1, \ldots, a_n\}$ and

$$\mathcal{I} = \{A \subseteq S, \text{ s.t. there exists a way to schedule the tasks in } A \text{ so that no task is late}\}.$$

Finding the maximum independent subset of M is equivalent to finding the optimal schedule (as shown in the class).

An important step in the greedy algorithm for the maximum independent subset problem is to check whether $A \cup \{x\} \in \mathcal{I}$ for $x \in S$. Show that for all $x \in S$, checking whether $A \cup \{x\} \in \mathcal{I}$ can be done in $O(n)$ time.

You may find the following lemma useful. (You can use this lemma without proving it.)

Lemma. For $t = 0, 1, \ldots, n$, let $N_t(A)$ denote the number of tasks in A whose deadline is t or earlier. Note that $N_0(A) = 0$ for any set A. Then, the set A is independent if and only if for *all* $t = 0, 1, \ldots, n$, we have $N_t(A) \leq t$.

2. (10 pts.) Apply the greedy algorithm for Horn formulas (covered in the lecture) to find the variable assignment that solves the following horn formulas:

1. $(w \wedge y \wedge z) \Rightarrow x, (x \wedge z) \Rightarrow w, x \Rightarrow y, \Rightarrow x, (x \wedge y) \Rightarrow w, (\bar{w} \vee \bar{x} \vee \bar{y}), (\bar{z})$
2. $(x \wedge z) \Rightarrow y, z \Rightarrow w, (y \wedge z) \Rightarrow x, \Rightarrow z, (\bar{z} \vee \bar{x}), (\bar{w} \vee \bar{y} \vee \bar{z})$

You need to describe each step of running the algorithm, i.e., the current assignment, which variable will be changed because of what, etc.