CS218, Winter 2024, Due: 11:59pm, Friday 02/02, 2024
Assignment #2 [1] **Training Programming Due: 11:59pm, Friday 01/26, 2024**
Name: Student ID:

Deadline. The homework is due **11:59pm, Friday 02/02, 2024**. You must submit your solutions (in pdf format generated by LaTeX) via GradeScope. The training programming assignment is due earlier, see more details below.

Late Policy. You have up to two grace days (calendar day) for the entire quarter. You don't lose any point by using grace days. If you decide to use grace days, please specify how many grace days you would like to use and the reason at the beginning of your submission.

Collaboration Policy. You can discuss the homework solutions with your classmates. You can get help from the instructor, but only after you have thought about the problems on your own. It is OK to get inspiration (but not solutions) from books or online resources, again after you have carefully thought about the problems on your own. However, you **cannot copy anything from other source**. You **cannot** share your solution/code with anyone else. You **cannot** read other's solution/code. If you use any reference or webpage, or discussed with anyone, you must cite it fully and completely (e.g., *I used case 2 in the examples in the Wikipedia page* `https://en.wikipedia.org/wiki/Master_theorem_(analysis_of_algorithms)` *about Master theorem for Problem 3*, or *I discussed problem 2 with Alice and Bob.*). *If you use ChatGPT or similar AI tools, you have to attach the full conversation to make it clear what help you obtained from them.* **Otherwise it will be considered cheating.** We reserve the right to deduct points for using such material beyond reason. **You must write up your solution independently, and type your answers word by word on your own: close the book/notes/online resources when you are writing your final answers.**

Write-up. Please use LaTeX to prepare your solutions. For all problems, **please explain how you get the answer** instead of directly giving the final answer, except for some special cases (will be specified in the problem). For all algorithm design problems, please describe using **natural language**. You could present pseudocode if you think that helps to illustrate your idea. Please do not only give some code without explanation. **In grading, we will reward not only correctness but also clarity and simplicity. To avoid losing points for hard-to-understand solutions, you should make sure your solutions are either self-explanatory or contains good explanations.** See some more details here `https://www.cs.ucr.edu/~ygu/teaching/218/W24/assignments/index.html`

Programming Problems. You will need to submit your code on CodeForces (a readme file about submitting code is available on the course webpage). You also need to submit a short report through GradeScope along with your solutions of the written assignments. In the report, you need to **specify your submission id**, describe the algorithm you designed, and show cost analysis if necessary. Note that your code will be automatically saved by CodeForces, so you do not need to submit the code again. For each problem, there will be 10-20 test cases. **For the training programming problems, you need to finish before 11:59pm, Friday 01/26, 2024.** With reasonable implementation, using C++ or Java is guaranteed to be able to pass all tests. You can use other languages, but it's not guaranteed that the implementations can be within the time limit.

[1] Some of the problems are adapted from existing problems from online sources. Thanks to the original authors.

Training and Challenge Problems (3.5pts for training + bonus for challenge)

Available on codeforces.

1 (3pts) Deadlines, again!

In class, we tried to use different greedy strategies to deal with homework deadlines — we tried several of them, but none of them could always gave an optimal solution.

Now, let's consider a slightly simpler problem and try to find a good greedy strategy for that — assume for each of the assignment, you need exactly one day to work on that. In particular, now you have a set of n homework assignments. The i-th one is worth p_i points, and is due on day d_i, where $1 \leq d_i \leq n$. For each of them, you need exactly one day to work on them, and as long as you spend one day on assignment i, you can get all p_i points of it. But if you do not finish an assignment before the deadline, you lose all p_i points.

Design a greedy strategy to find out which assignment you should work on for each day, such that you can earn the most number of points. For simplicity, assume all p_i are distinct.

Note: You can work on the problem on the due day (it's due midnight of the day).

Questions:

1. (0.2pts) There are several straightforward greedy strategies. First, consider you use "deadline first" strategy, where, from the first day, you start to work on the one with the earliest deadline (may need to break the tie based on some rules). Please show an counterexample with three assignments (i.e., $n = 3$) where this strategy does not give you the optimal solution.

2. (0.2pts) There is another simple idea: highest point first. From the first, day, you work on the assignments from the highest score to lowest. Please show an counterexample with three assignments (i.e., $n = 3$) where this strategy does not give you the optimal solution.

3. (0.5pts) Prove that, an optimal solution must contain the assignment with the highest number of points.

4. (0.5pts) Show a greedy algorithm to find out which assignment you should work on for each day, such that you can earn the most number of points.

5. (0.3pts) Simulate your algorithm on the following input instance:

Assignment	Points	Deadline
A	10	13
B	8	2
C	7	2
D	15	5
E	1	7
F	12	3
G	4	4
H	5	5

   Show what solution your algorithm will get (which assignment you will work on for each day?), and the total number of points you can earn.

6. (0.8pt) What is the greedy choice in your algorithm? Prove that your greedy choice is always "good"/"safe" to be included in the optimal solution

7. (0.5pt) What does "optimal substructure" mean? Prove that this problem exhibits "optimal substructure".

Combining your proofs in the last two subquestions will prove that your algorithm will always give an optimal solution.

(For "counterexample" it means to show: 1) an input instant, 2) the solution S obtained by the given algorithms, and 3) a better solution than S (probably an optimal one). This means that your algorithm does not always generate the best solution).

2 (1.5pts) Don't wait too long!

After teaching CS 218 in SOMED, Yan will walk back to his office on the third floor in Winston Chung Hall. The third floor is not high, so Yan usually uses the stairs, which he knows will take $S = 60$ seconds. However, Yan is tired after giving the lecture, so in this case he prefers using the elevator which takes $E = 20$ seconds. As you probably know, the elevator in Winston Chung Hall is old, so sometimes it just never comes. Hence, Yan does not want to wait forever, so he decided to use the following strategy: he will wait for at most W seconds, and if the elevator does not come, he will use the stairs.

Please calculate the best W that gives Yan the best approximation ratio. Here, the approximation ratio is defined as the time Yan used divided by the time for the optimal strategy, in the worst case. You can use 60 and 20 seconds when you are trying to figure out the solutions, but please use S and E to formally state the result.