CS218, Winter 2024, **Due: 11:59pm, Fri 2/16, 2024**
Assignment #3 [1] **Required Programming Due: 11:59pm, Fri 2/9, 2024**
Name: **Student ID:**

Deadline. The homework is due **11:59pm, Fri 2/16, 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 four 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, Fri 2/9, 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.

1 (2 + 0.5 pts) Everything on sale!

Yihan is going shopping, and she finds out that the shopping mall is now having a big sale. There are n items on sale. For every item, Yihan knows the original price p_i and the current price t_i. However, there are some limitations for the sale: some items are categorized in the same group, and the discount can apply to at most one item in this group (i.e., you have to pay the original price p_i to buy the second item in the group). For each item i, we use g_i to denote its group number. Each item can only appear in one group. For each item, Yihan plans to buy **at most one of them**.

All p_i and t_i are positive integers, and $p_i > t_i$ is true for all i. Yihan thinks that the benefit she gets from an item the difference between p_i and t_i (i.e., $p_i - t_i$). Yihan has a budget of W dollars. She wants to know what is the highest total benefit she can get using the W dollars. If there are multiple ways to achieve the highest benefit, she prefers the one with the fewest items, since it makes her life easier to carry them back :)

For all problems below, if you design a dynamic programming algorithm, please specify 1) your DP recurrence, 2) what the states mean, 3) what is the decision made by your DP recurrence, and 4) what is the boundary case.

Questions:

1. (0.3 pt) Prove that, in the optimal solution, you should never buy two items in the same group.
2. (0.3 pt) Now describe an algorithm that helps Yihan select items within W dollars with the highest benefit. Your algorithm should run in $O(nW)$ time. You only need to output the highest benefit in this problem (i.e., no need to give the concrete plan of what items to buy).
3. (0.6 pt) Then, let's try to consider the preference in fewer items. Modify your algorithm in Problem 2, and also output the fewest number of items that allows for the highest benefit. In this question, you only need to output the number of items, instead of the concrete plan. Your algorithm should run in $O(nW)$ time.
4. (0.4 pt) Finally, Modify your algorithm for Problem 3, and also give the list of items to buy to achieve the highest benefit (i.e., achieving the highest benefit with the fewest number of items). Your algorithm should run in $O(nW)$ time.
5. (0.4 pt) Now consider a different setting: if for each item, Yihan can buy an unbounded number of them. Since they are the same item, the discount still applies to all units of them. In this case, can you change your algorithm to give the highest possible benefit (you only need to output the highest benefit)? Your algorithm should run in $O(nW^2)$ time.
6. (0.5 pt bonus) For Problem 5, can you optimize your algorithm to have $O(nW)$ time?

2 Morse Code (1.5 pts)

In class we learned Morse Code. Because it is not a prefix code, we've seen that a code can be interpreted in several different ways. For example, ".-.-" can be interpreted as `AA`, `EK`, `RT`, and so on.

A (3)	.-	B (2)	-...	C (1)	-.-.	D (2)	-..	E (3)	.	F (3)	..-.
G (2)	--.	H (3)	....	I (3)	..	J (1)	.---	K (2)	-.-	L (1)	.-..
M (2)	--	N (2)	-.	O (1)	---	P (2)	.--.	Q (2)	--.-	R (2)	.-.
S (1)	...	T (2)	-	U (1)	..-	V (1)	...	W (1)	.--	X (2)	-..-
Y (2)	-.--	Z (1)	--..								

We use $s[c]$ to denote the number of strokes of a letter c. Here we only consider upper case letters. The number of strokes are also noted in the parentheses after each letter. There are of course multiple ways to handwrite a letter, here I'm using a common version appearing in many letter written templates. For some reason, for the same morse code, we prefer those interpretations with fewer strokes.

Given a Morse code X (a string consisting of . and $-$), we want to know, **among all the possible interpretations of X, which one uses the smallest number of strokes**. For simplicity, assume we have a unit-cost function $\mathit{check}(X, i, j, c)$ that checks if the substring $X[i..j]$ is the Morse code for character c. For example, $\mathit{check}(X, 2, 5, \text{'B'})$ checks if the second to the fifth character in X is "-..." (the Morse code for 'B').

As an example, consider $X =$".-.-". There are many possible interpretations for X: `AA` (6 strokes), `EK` (5 strokes), `RT` (4 strokes), `ETET` (10 strokes), ... The fewest strokes we can get is 4.

Hint: when you use LaTeX, you should use `\hyp` to represent the "dash", instead of "-" because multiple consecutive "-" will be combined into one.

(1) (0.2 pt) What is the Morse code for word "UCR"? Is there any other interpretations of UCR's code (if there's no separator)? Please list at least three of them, and at least one of them should have length ≤ 4. Which of them have the fewest strokes?

(2) (0.8 pt) Design an algorithm for the above task: given a Morse coding string X, find the one with the fewest strokes among all possible interpretations of X. (Hint: You can use $f[i]$ to denote the fewest strokes you could get using first i elements in X.)

(3) (0.2 pt) What is the time complexity of your algorithm?

(4) (0.3 pt) Use your algorithm to compute: what is smallest number of strokes you can get by interpreting string "--..--."?