CS 3510: Design & Analysis of Algorithms November 10th 2024

Programming Problems

Bonus Assignment Due: December 3rd 2024

1 Guidelines

1. Please read this document in its entirety. It contains important information regarding what you are allowed to submit and what modifications you are permitted to make to the templates.
2. Start early. As noted in the submission limits section, you are allowed only three submissions every twenty-four hours. There is no late acceptance for this assignment, and we will not increase the number of allowed submissions as the deadline approaches.
3. You are allowed to collaborate with your peers, but any code you submit must be your own. You are not permitted to share solution code or view someone else's solution code. Any violations will be reported directly to the Office of Student Integrity.
4. This is a new assignment and autograder. We have tested it thoroughly, but if you encounter any issues with the assignment itself, please report them on Ed Discussion as soon as possible.
5. If you have anything to discuss on Ed, tag it with the "EC Assignment" tag and include the question number in the title.

2 Overview

This programming assignment serves as extra credit toward your final grade in CS 3510. We are currently offering up to 3% extra credit for this assignment. You may choose to work in C++, Java, or Python. To begin the project, simply download the Solutions file from Canvas for the language you wish to use. Implement each problem defined in Section 6 using the provided method headers. For more details and restrictions, refer to Section 4.

The assignment has a total of 150 points. There are fourteen problems detailed in Section 6 that are each worth ten points. If all tests pass for all fourteen problems, an additional ten points will be awarded. Partial credit is awarded on the overall assignment, but each problem is all-or-nothing. This means you must pass all the test cases for a problem to earn the ten points for that problem, though you may choose to complete only a subset of the problems.

To receive full credit for a problem, you must not only pass the autograder but also include an explanation and runtime analysis. Each method header has space for these details in the comment block above. We expect a 4-5 sentence explanation of why your algorithm works, similar to the correctness proofs we've done in class. We also expect a runtime analysis in big-O notation with a 1-2 sentence justification for the provided runtime. These written submissions will be manually graded, and no points will be awarded for a problem if the written submission is missing.

3 Autograder

The autograder calls each of the methods defined in the `Solutions` class with input from our test cases. The autograder compares the output from your method to the expected output. If the output for any test case differs from the expected result, no points will be awarded for that problem. Test cases will not be disclosed to students.

The autograder also enforces a runtime limit for each individual test case. The timeout is set to three times the runtime of our reference implementation. This ensures students cannot brute-force solutions and encourages efficient implementations. If the autograder indicates that you are exceeding the time limit, consider optimizing your implementation by removing unnecessary loops or processing. If you are certain that your solution is as optimal as possible and you are still exceeding the time limit, make a private post on Ed and we'll take a look at your submission.

The autograder only reports whether all test cases were passed for each problem. If not all test cases are passed, the autograder will indicate whether the failure was due to mismatched outputs or a timeout. If you are failing tests due to mismatched outputs, you must write your own local tests and consider edge cases. **Students are allowed to share local testing tools**, and we encourage you to do so through Ed Discussion. If you share local tests, they must be in a separate file from Solutions.py, Solutions.java, or Solutions.cpp (e.g., a wrapper program that calls methods in the Solutions files). You are never allowed to copy-and-paste someone else's code into your Solutions file, but you can copy-and-paste tests that interact with the Solutions file onto your local machine.

For reference and consistency in running local tests, the autograder currently uses:
Python 3.10.12 for Python code
g++ 11.4.0 for compiling C++ code
OpenJDK 11.0.24 for compiling Java code.

4 Submission Format and Limits

Submit either Solutions.py, Solutions.java, or Solutions.cpp to the autograder. The autograder will fail if more than one file is submitted or if the file is named anything other than Solutions.py, Solutions.java, or Solutions.cpp. The autograder simply calls the methods defined inside the `Solutions` class, so you may add public or private helper methods within the class. **However, do not modify any of the existing method headers.**

Students are limited to three submissions within any 24-hour period. Any submissions beyond this limit will not be graded, and the score from the previous submission will be retained.

Finally, do not add any imports or includes to the Solutions files. You must work with the imports/includes already defined, and adding any more will result in a zero for the entire assignment.

5 Honor Code

Read the below excerpt from the Georgia Tech Academic Honor Code. Plagiarism includes but is not limited to copying another student's code, copying code from websites such as Stack Overflow, and using generative AI tools to complete the assignment. We will be checking each submission for signs of honor code violations and any violation will be directly reported to the Office of Student Integrity.

Honor Code Excerpt: Students are expected to act according to the highest ethical standards. The immediate objective of an Academic Honor Code is to prevent any Students from gaining an unfair advantage over other Students through academic misconduct. The following clarification of academic misconduct is taken from Section XIX Student Code of Conduct, of the Rules and Regulations section of the Georgia Institute of Technology General Catalog: Academic misconduct is any act that does or could improperly distort Student grades or other Student academic records. Such acts include but need not be limited to the following:

- **Unauthorized Access:** Possessing, using, or exchanging improperly acquired written or verbal information in the preparation of a problem set, laboratory report, essay, examination, or other academic assignment.
- **Unauthorized Collaboration:** Unauthorized interaction with another Student or Students in the fulfillment of academic requirements.
- **Plagiarism:** Submission of material that is wholly or substantially identical to that created or published by another person or persons, without adequate credit notations indicating the authorship.
- **False Claims of Performance:** False claims for work that has been submitted by a Student.
- **Grade Alteration:** Alteration of any academic grade or rating so as to obtain unearned academic credit.
- **Deliberate Falsification:** Deliberate falsification of a written or verbal statement of fact to a Faculty member and/or Institute Official, so as to obtain unearned academic credit.
- **Forgery:** Forgery, alteration, or misuse of any Institute document relating to the academic status of the Student.
- **Distortion:** Any act that distorts or could distort grades or other academic records.

While these acts constitute assured instances of academic misconduct, other acts of academic misconduct may be defined by the professor.

6 Problem Descriptions

Problem 1: Real Estate Profits

You are working for a real estate firm based out of Atlanta, GA. The city has decided to buy your properties in Midtown in order to develop new (unaffordable) housing for Georgia Tech students. You are forced to sell your properties one by one, but the price you can sell each property is dependent on the property value of its neighbors.

Each of your n properties currently have a value of v_i, which are provided as an array of size n. When you sell the property at position i, you earn a profit equal to $v_{i-1} \cdot v_i \cdot v_{i+1}$. If $i-1$ is out of bounds of the array, you can assume $v_{i-1} = 1$. The same applies for v_{i+1}.

Your objective is to determine the maximum possible profit you could obtain by selling your properties to the city.

Example:
Input: $[5, 1, 8, 3]$
Output: 180

Explanation: You sell the second property, obtaining a profit of $5 \cdot 1 \cdot 8 = 40$ and leaving the array $[5, 8, 3]$. You again sell the second property, receiving $5 \cdot 8 \cdot 3 = 120$ and leaving $[5, 3]$. You then sell second property and then last remaining property, which net 20 together. The final profit obtained is 180 and all properties have been sold.

Problem 2: Warehouse Package Stacking

You finally got an internship at Amazon, congrats! Unfortunately, you have to start as a warehouse worker and work your way up. You are managing the storage of packages in the warehouse. Each package has a specific width and height, and you want to stack as many packages as possible in a single column so that each package in the stack fits within the dimensions of the package directly below it. More specifically, a package can only be stacked on top of another package if **both its width and height are strictly smaller than the package below it**.

Bezos will only promote you if you are good at this task, so return the **maximum number of packages you can stack up.**

Note: You cannot rotate the packages.

Input Format: You are given an array `packages` where each entry `packages`$[i]$ is a tuple $[\texttt{width}_i, \texttt{height}_i]$ representing the width and height of the i-th package.

Example 1:
Input: `packages` $= [[5,4],[6,4],[6,7],[2,3]]$
Output: 3
Explanation: The maximum number of packages you can stack is 3 ($[2,3] \rightarrow [5,4] \rightarrow [6,7]$).

Example 2:
Input: `packages` $= [[8,9],[1,1],[6,10],[3,4],[5,8],[2,3]]$
Output: 5
Explanation: The maximum number of packages you can stack is 5 ($[1,1] \rightarrow [2,3] \rightarrow [3,4] \rightarrow [5,8] \rightarrow [8,9]$).

Problem 3: Building Blocks

Two toddlers started learning Dynamic programming in kindergarten to start early on their CS interview preparation. They decided to take a break and play with some building blocks. They each have a sequence of blocks of varying heights that may not be sorted, ie, $blocks1[a_1, a_2, .., a_n]$ and $blocks2[b_1, b_2, .., b_n]$. They both have the same number of blocks. They each want a sequence of blocks with **strictly increasing heights** to build a nice-looking structure.

As their sequence of blocks may not be sorted they decide to do this by exchanging **corresponding blocks only with each other, ie, they will only swap a_i with b_i** to get a strictly increasing sequence. They want to know the minimum number of swaps they need to do to achieve this. They recognize that they need to use DP to solve this, but can't come up with an efficient solution, so they ask you. Please provide an efficient DP solution to help them. If no solution is possible return -1.

Example
Input: `N = 4, blocks1 = [3, 4, 5, 4], blocks2 = [1, 2, 3, 8]`
Output: 1
Explanation: They can swap their last blocks to achieve strictly increasing sequences. After swapping their last blocks, they will have `blocks1 = [3, 4, 5, 8], blocks2 = [1, 2, 3, 4]`.

Problem 4: Modular Two Sum

Given a list of integers A: $[a_1, a_2, ..., a_n]$ and an integer k, count all pairs of integers whose sum is divisible by k. In mathematical terms, count pairs where $k \mid (a_i + a_j)$ **and** $i < j$. You may assume $k \geq 2$ and $\text{len}(A) \geq 2$.

A brute force approach entails summing all possible pairs of the list's integers and checking their divisibility by k, running in $O(n^2)$. Produce a more efficient algorithm utilizing properties of modular arithmetic.

Example:
Input: $A = [5, 7, 1, 10, -4, 119]$ $k = 6$
Output: 5
Explanation:
In the given list, there are five pairs with a sum divisible by 6.
6 | (5 + 7)
6 | (5 + 1)
6 | (7 + 119)
6 | (10 + (-4))
6 | (1 + 119)

Problem 5: Maximum Magic Power Path

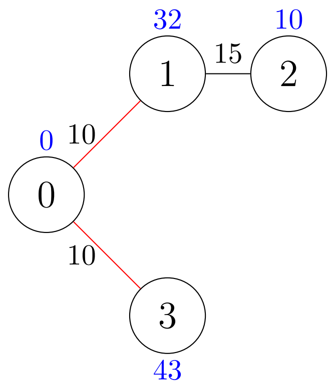
Imagine a magical kingdom represented by an enchanted forest with n mystical locations (nodes) numbered from 0 to $n-1$. Each location i holds a unique magical energy level given by the integer array energies where energies[i] represents the magical power of location i. The forest paths connecting these locations are enchanted, with each path taking a certain amount of time to travel. The paths are described in a 2D integer array paths, where each entry paths[j] = $[u_j, v_j, \text{time}_j]$ indicates a magical path between locations u_j and v_j, with time_j seconds required to travel this path. All paths are bi-directional, allowing travel in both directions. The kingdom has an ancient time-binding spell, maxTime, which limits how long a traveler can spend journeying in the forest.

A **valid journey** in the enchanted forest is a path that:

- Starts and ends at location 0,
- Takes at most `maxTime` seconds to complete,
- May revisit locations as often as needed.

The **power** of a journey is the sum of the magical energies of the unique locations visited along that journey (each location's energy contributes only once per journey). Write an algorithm to find the maximum possible power of any valid journey. **Note:** Due to the ancient protections on each location, each one is connected by at most four paths.

Example:



Input: `energies = [0,32,10,43], edges = [[0,1,10],[1,2,15],[0,3,10]], maxTime = 49`
Output: 75
Explanation: One example route is $0 \to 1 \to 0 \to 3 \to 0$. The total time is $10 + 10 + 10 + 10 = 40$, which is within the limit of 49. The visited locations are 0, 1, and 3, giving a total power of $0 + 32 + 43 = 75$.

Problem 6: Divide the Harvest

You have a long row of harvest baskets, each filled with a certain quantity of fruit. The quantity in each basket is given by the array `quantity`. You want to share the baskets with your k neighbors, so you'll divide the row into exactly $k + 1$ portions by making k cuts. Each portion will consist of consecutive baskets.

To be fair, you want to ensure that the portion with the **smallest total fruit quantity** is as large as possible. However, to keep the peace, you'll take the portion with the **second-smallest total fruit quantity** for yourself, while giving the others the remaining portions.

Find the highest possible quantity of fruit in the portion you can keep, assuming you always take the second-smallest portion after dividing. Assume that $1 \leq k < len(quantity)$.

Example:
Input: `quantity = [1,2,3,4,5,6,7,8,9], k = 5`
Output: 6
Explanation: You can divide the harvest into `[1,2,3]`, `[4,5]`, `[6]`, `[7]`, `[8]`, `[9]`. The portion with the least fruit quantity has 6, so you take the next one which also has 6.

Problem 7: Coloring Sidewalks

Woohoo! All the sidewalk construction around campus is finally finished. However, as the last step, GT authorities want to add more color to the new sidewalks by painting them gold, white, or blue. With n added sidewalks, they task you with painting the sidewalks one of the 3 colors. The time to color each sidewalk a certain color is different, which is given to you by a n x 3 matrix. For example, time[0][0] represents the number of minutes required to color sidewalk 0 gold; time[1][2] represents the number of minutes required to color sidewalk 1 blue; and so on.

Your job is to color all the sidewalks such that no two adjacent sidewalks are the same color. Being the busy student you are, you want to minimize the amount of time you spend painting the sidewalks. Determine the minimum time required to satisfy the required coloring.

Example:
Input: `time = [[3,2,5],[3,4,6],[3,1,2], k = 5`
Output: 6
Explanation: Given a row of 3 sidewalks, the shortest amount of time to color the row such that no two adjacent tiles are the same color is to color sidewalk 0 white (2 minutes), sidewalk 1 gold (3 minutes), and sidewalk 2 white (1 minute), giving a total of 6 minutes.

Problem 8: Chemical Concoctions

A scientist recently discovered some new chemicals. There are at most 26 new chemicals, and each one is labeled with a distinct letter from ‘a’-’z’ (all lower case). The scientist needs to find the natural ordering of these chemicals, but the only hint is a formula sheet. The sheet contains a list of non-empty chemical formulas sorted based on the special ordering. Every chemical found by the scientist is guaranteed to be on the sheet. There are two known rules about the natural order given below.

For two formulas X and Y, X comes before Y if either condition holds true:

- The first chemical in X is less than the corresponding chemical in Y.
  - Ex: X=“abp” and Y=“abi”. Assuming that X<Y, this means that ‘p’<‘i’.
- X contains n chemicals which are the exact same as the first n chemicals in Y, but X uses fewer chemicals than Y.
  - Ex: X=”zpeos” and Y=”zpeosd”. X<Y since the first 5 chemicals of X and Y match, but Y contains more chemicals.

Your goal is to help the scientist find a valid ordering. If multiple orderings exist, return any one of them. If no ordering is possible, return an empty string.

Example:
Input: [“bz”, “buzz”, “fizz”, “fz”, “zi”]
Output: “bfizu” or “bifzu”
Explanation:

- from “bz” and “buzz”, we know that ‘z’ < ‘u’
- from “buzz” and “fizz”, we know that ‘b’ < ‘f’
- from “fizz” and “fz” we know that ‘i’ < ‘z’
- From “fz” and “zi” we know that ‘f’ < ‘z’

Using the conditions discovered by comparing the words, we find that two possible valid orderings are “bfizu” or “bifzu”.

Problem 9: Maximum Sum of Non-Adjacent Subsequence

You are given an integer array nums containing n integers. You need to select a subsequence of elements from nums such that no two elements in the subsequence are adjacent in nums. Your goal is to maximize the sum of the selected elements.

Return the maximum possible sum of the non-adjacent subsequence.

Example:
Input: nums $= [3, 2, 7, 10]$
Output: 13
Explanation: The optimal solution is to pick 3 and 10, resulting in a maximum sum of $3 + 10 = 13$.

Problem 10: DigitGPT

As part of their work at ClosedAI, Scam Altman and Melon Musk have developed a new model called DigitGPT that generates a random string of length m as output whenever prompted. Each of the characters in this string is a number between 0 and 9.

Dr. Ilya Cutskewer recently left ClosedAI due to conflicts with management and, in order to take revenge, has developed a bot that randomly erases an even number of characters in DigitGPT's output. Specifically, whenever DigitGPT is prompted, Dr. Cutskewer may launch an attack that replaces an even number of characters (possibly 0) with $*$ symbols (without affecting the order of the remaining digits) before the output reaches the end user.

Internally, while ClosedAI engineers are being overworked to prevent these attacks from affecting any more users, Scam and Melon have placed a bet on the strings generated by DigitGPT. For a given string, Scam and Melon will take turns (with Scam going first) replacing $*$ symbols with digits between 0 and 9, stopping when there are no more $*$ symbols in the string. Melon will replace Scam as CEO if, after all the turns, the sum of the digits in the first half of the string is equal to the sum of the digits in the second half of the string. In any other case, Scam remains CEO. Assuming both Scam and Melon are perfect logicians and play optimally, will Melon replace Scam as CEO?

Given the number of characters in the string generated by DigitGPT and the string of length m (where m is even) generated by DigitGPT following Dr. Cutskewer's attack (this string will contain a combination of digits and $*$ symbols such that the number of $*$ symbols is even), return whether or not Melon will replace Scam as CEO.

Example 1:
INPUT:
4
2433
OUTPUT: True

Example 2:
INPUT:
6
000**0
OUTPUT: False

Explanation: In example 1, there are no remaining moves and the sum of the digits in the first half is equal to the sum in the second half. In example 2, Scam can replace any $*$ symbol with a 1, preventing the sum from being equal.

Problem 11: Building a Brick Wall

You are constructing a wall around your house using bricks of different lengths. Each type of brick is available in unlimited quantities, and you want the total wall length to cover three sides of your house, which is given by the length l.

Determine the number of unique ways in which you can build this wall using the available bricks.

If you cannot make the wall with the given lengths of bricks, return 0.

Example:
Input: B = [1,2,5]; l = 6
Output: 5
Explanation: Given 3 different lengths of bricks 1,2,5. Given a target wall length of 6. There are exactly 5 ways to make this wall:

6 = 5+1
6 = 2+2+2
6 = 2+2+1+1
6 = 2+1+1+1+1
6 = 1+1+1+1+1+1

Problem 12: Stuck in an Archipelago

You work for an infrastructure company which is building bridges for a nation-state located in an archipelago. The archipelago is prone to many disasters like hurricanes, and tsunamis; these may take out bridges between the islands of the archipelago. The government wants you to build bridges such that even if one bridge from an island to another island is destroyed, there will always be another bridge that can be crossed to leave a group of islands.

You are on a subteam tasked with finding needed bridges which are bridges where if any one of them are removed, it will disconnect some island from some other island.

You're given the number of islands, and the number of bridges between those islands. Following that, you'll be given pairs of numbers, each represents a bridge from the island with the id of the first number to the id of the second number. You should print out, the number of needed bridges followed by their values.

Example:
Input: $n = 4, m = 4, E = \{(0, 1), (1, 2), (2, 0), (1, 3)\}$
Output: $\{(1, 3)\}$
Explanation: This is because (1,3) is the only bridge that, if removed, would disconnect some islands from the rest. All other edges have redundant paths, ensuring connectivity even if they are removed.

Problem 13: Search Engineer

You are a smart student at Georgia Tech and well-known for your skills in algorithms. One of your old friends has started his company which is building a super-fast search engine. He asks you for your help in optimizing his search engine. Now, you are faced with the following task:

You are given two strings `text` and `pattern`. You need to find the number of distinct subsequences of `text` which are equal `pattern`.

A subsequence of the string `a` is defined as a sequence that can be obtained from `a` by deleting some elements (possibly none), without changing the order of the remaining elements.

Example:
Input: `text` = `rabbbit` and `pattern` = `rabbit`
Output: 3
Explanation: As shown below, there are 3 ways in which you can generate `rabbit` from `rabbbit`.

```
ra)b(bbit
rab)b(bit
rabb)b(it
```

Problem 14: Buzz's Bees

Buzz has recently decided to pick up beekeeping! He has N worker bees and wants to establish a communication network for the bees to share the latest pollen locations with each other.

The i^{th} bee is located at a distinct position (x_i, y_i) in the field. Establishing a connection between bee i and j costs $(x_i - x_j)^2 + (y_i - y_j)^2$. Can you help Buzz calculate the minimum cost needed to build a network connecting all N bees?

Example
Input: `N = 3, locations = [(1, 4), (3, 2), (0, 3)]`
Output: `10`
Explanation: Build a connection between `(1, 4)` and `(0, 3)` for cost 2 and a connection between `(1, 4)` and `(3, 2)` for cost 8.