Data Structure and Algorithm, Spring 2022

Homework 3

Due: 13:00:00, Tuesday, May 17, 2022

TA E-mail: dsa_ta@csie.ntu.edu.tw

Rules and Instructions

- Any form of cheating, lying, or plagiarism will not be tolerated. Students can get zero scores and/or fail the class and/or be kicked out of school and/or receive other punishments for those kinds of misconducts.
- In Homework 3, the problem set contains a special Problem 0 (see below) and 5 other problems. The set is divided into two parts, the non-programming part (Problems 0, 1, 2, 3) and the programming part (Problems 4, 5).
- For problems in the non-programming part, you should combine your solutions in ONE PDF file. Your file should generally be legible with a white/light background—using white/light texts on a dark/black background is prohibited. Your solution must be as simple as possible. At the TAs' discretion, solutions which are too complicated can be penalized or even regarded as incorrect. If you would like to use any theorem which is not mentioned in the classes, please include its proof in your solution.
- The PDF file for the non-programming part should be submitted to Gradescope as instructed, and you should use Gradescope to tag the pages that correspond to each sub-problem to facilitate the TAs' grading. Failure to tagging the correct pages of the sub-problem can cause a 20% penalty on the sub-problem.
- For the programming part, you should have visited the *DSA Judge* (https://dsa-2022.csie.org) and familiarized yourself with how to submit your code via the judge system in Homework 0.
- For problems in the programming part, you should write your code in C programming language, and then submit the code via the judge system. In each day, you can submit up to 5 times per day for each problem. To encourage you to start early, we allow 10 times of submissions per day in the first week (2022/04/26-2022/05/03). The judge system will compile your code with

```
gcc main.c -static -O2 -std=c11
```

- Discussions on course materials and homework solutions are encouraged. But you should write the final solutions alone and understand them fully. Books, notes, and Internet resources can be consulted, but not copied from.
- Since everyone needs to write the final solutions alone, there is absolutely no need to lend your homework solutions and/or source codes to your classmates at any time. In order to maximize the level of fairness in this class, lending and borrowing homework solutions are both regarded as dishonest behaviors and will be punished according to the honesty policy.
- The score of the part that is submitted after the deadline will get some penalties according to the following rule:

$$\text{LateScore} = \max\left(0, \frac{86400 \cdot 5 - \text{DelayTime (sec.)}}{86400 \cdot 5}\right) \times \text{OriginalScore}$$

- If you have questions about HW3, please go to the discord channel and discuss (*strongly preferred*, which will provide everyone a better learning experience). If you really need an email answer, please follow the rules outlined below to get a fast response:
  - The subject should contain two tags, `"[hw3]"` and `"[Px]"`, specifying the problem where you have questions. For example, `"[hw3][P5] Is k in sub-problem 3 an integer"`. Adding these tags allows the TAs to track the status of each email and to provide faster responses to you.
  - If you want to provide your code segments to us as part of your question, please upload it to Gist or similar platforms and provide the link. Please remember to protect your uploaded materials with proper permission settings. Screenshots or code segments directly included in the email is discouraged and may not be reviewed.

Non-Programming Part

Problem 0 - Proper References (0 pts)

For each problem below, please specify the references (the Internet URL you consulted with or the classmates/friends you discussed with; you do not need to list the TAs/instructors) in your PDF submitted to Gradescope. If you finished any problem all by yourself (with the help of TAs/instructors), just say "all by myself." While we did not allocate any points on this problem, failure to complete this problem can lead to penalty (i.e. negative points). Examples are

- Problem 1: Alice, Bob, and https://stackoverflow.com/questions/982388/
- Problem 2: all by myself
- ...

Listing the references in this problem does *not* mean you could copy from them. We cannot stress this enough: *you should always write the final solutions alone and understand them fully.*

Problem 1 - Cracking The Interview With How-How! (100 pts)

How-How, a junior student at NTU CSIE, is busy looking for intern opportunities inside big tech companies such as FAANG, just like other junior students. However, he always felt that he was not well prepared before the interview.

He went through his secret folder, where he stored all the great programming problems he had collected from previous courses (including DSA, obviously) and some online platforms like LeetCode, to find problems to practice. There are n problems, $P_1, ..., P_n$, inside the folder with **distinct** difficulties, $d_1, ..., d_n$, respectively.

How-How decided to set up mock interviews for himself to practice. Typically, tech interviews consist of two problems, a basic one and a more difficult follow-up one. The mock interviews should follow this pattern. However, as a well-organized person, How-How would only solve the problems in order. That is, only when the two problems P_i, P_j satisfy $i < j$ and $d_i < d_j$ can they be used as a mock interview.

1. (5 pts) Consider the case where there are 6 problems, $P_0, ..., P_5$, in the folder, and their difficulties are $(7, 1, 3, 9, 5, 2)$. Please list all pairs of problems that can be used as a mock interview. For example, (P_1, P_2) is a feasible pair since $1 < 2$ and $d_1 = 1 < 3 = d_2$. You do not have to explain your answer in this question.

2. (20 pts) Consider the case where there are n problems in the folder. Please design an algorithm that finds the total number of feasible problem pairs in $O(n \log n)$-time.
   (Hint: Use the concept of merge sort!)

After solving all the mock interviews, How-How discovered that only three days were left before his first interview! He decided to practice all the problems for the last time. He sorted the problems according to their difficulty and wanted to do them one by one. However, somebody who hates How-How (possibly the CEO of LeetCode, since How-How uses LeetCode every day but still refuses to pay for the premium membership) messes the order up by reordering k problems, where $k < n$.

Assume that $order(P)$ is the index of problem P when How-How sorted all the problems **according to their difficulty**, and $order'(P)$ is the index of problem P after reordering the problems, there are exactly k misplaced problems that $order(P) \neq order'(P)$.

3. (5 + 20 pts) How-How designed an algorithm that removes the out-of-order pairs of the problems to sort the remaining problems by their difficulty. Below is the pseudo code of it.

```
Remove-out-of-order-pairs(Problem P[]):
    while (!Ordered(P)):
        // function Ordered() check if P is ordered
        // and it runs in O(n)-time
        // where n is the number of problems
        i = 0
        while (i+1 < len(P)):
            if (P[i].difficulty > P[i+1].difficulty):
                remove(P[i]) // remove() runs in O(1)-time
                remove(P[i+1])
            else:
                i += 1
```

   (a) (5 pts) Consider the case where there are 7 problems with difficulties $(1, 4, 5, 2, 3, 6, 7)$. List the difficulties of problems that the above pseudo code would remove. You do not have to explain your answer in this question.

   (b) (20 pts) Please analyze the time complexity of the above pseudo code.

After running the pseudo code for a few times, How-How discovered that there would be at most $2k$ problems removed. He thought it's quite a magical property, so he named this property HowMagic!.

4. (15 pts) Please prove the HowMagic! property. 就 worst case 說明就好

5. (15 pts) The above pseudo code ran slowly even on How-How's powerful workstation with the latest Intel Xeon Platinum 8380 processor and NVIDIA RTX 3090Ti graphic card. Please help How-How to design an algorithm with the same effect as the above pseudo code but runs in $O(n)$-time, where n is the number of problems.

6. (20 pts) Design an algorithm that sorts **all** the problems according to their difficulty without removing any problems. Your algorithm should runs in $O(n + k \log k)$-time, where n is the number of problems, and k is the number of misplaced problems.

Problem 2 - DSA Chief Scientist How-How (100 pts)

After years of hard work, How-How has become the chief scientist of the **Der-tian Space Agency (DSA)**. He now leads a team to find mineral deposits of valuable resources on other planets. After searching for a long time, they have discovered K candidate planets that are likely to contain abundant resources. They have created detailed maps by marking $N \times M$ points on the (spherical) planets with lowercase English letters. The $N \times M$ points are selected by intersections of N different latitude lines and M different longitude lines. Since How-How wants to compare the planets, they use the same set of latitude and longitude lines for each planet. The maps can be simplified to K different N by M grids. Recently, the team led by How-How has made enormous progress on this study. They found out that if a particular pattern appears on the map of a planet (e.g. "a b c" appears on the map), it is very likely that the planet is full of valuable minerals! As a scientist of the DSA, could you help them find the perfect planet? In this problem, if you are using rolling hash (Rabin-Karp), you may assume that there is no collision or the amount of collisions is sufficiently small.

1. (30 pts) How-How's team wants to find all maps that contain the pattern of shape $1 \times h$ ($h \leq M$). Please design an algorithm that finds all maps consisting of the desired pattern with **worst case running time of** $O(K \times M \times N + h)$. In other words, your algorithm should output a subset of the K candidate maps that satisfy the constraint. Note that the pattern may be across the vertical map borders. For example, given the desired pattern "i s q", the following 3 maps are examples that contain the pattern:

   Map 1:

   $a\ b\ c\ d\ e$
   $q\ p\ o\ i\ s$
   $q\ q\ p\ r\ r$

   Map 2:

   $q\ i\ s\ q\ p$
   $h\ r\ c\ y\ e$
   $a\ b\ c\ d\ e$

   Map 3:

   $i\ h\ f\ x\ z$
   $a\ b\ v\ d\ e$
   $p\ b\ i\ s\ q$

   and the following 3 maps are examples that do not contain the pattern:

   Map 1:

   $a\ b\ c\ d\ e$
   $d\ p\ o\ i\ s$
   $q\ q\ p\ r\ r$

   Map 2:

   $q\ i\ a\ x\ p$
   $h\ s\ c\ y\ e$
   $a\ q\ c\ d\ e$

   Map 3:

   $i\ h\ f\ x\ z$
   $a\ b\ v\ d\ e$
   $p\ b\ o\ s\ l$

2. (10 pts) Please prove that your algorithm in the previous subproblem has an **worst case running time of** $O(K \times M \times N + h)$.

3. (20 pts) How-How's team found out that the 1-dimensional constraint (pattern) is too weak and only eliminates a few candidate maps. To make the candidate set much smaller,

they made the target pattern 2-dimensional with shape $g \times h$ ($g \leq N$, $h \leq M$). Please design an algorithm that finds all maps that consisting of the desired pattern with **worst case running time of** $O(K \times M \times N + g \times h)$. Note that the pattern may be across the vertical map borders. For example, given the desired pattern

$$\begin{matrix} i & s & q & p \\ r & r & q & q \end{matrix}$$

the following 2 maps are examples that contain the pattern:

Map 1:

$$\begin{matrix} a & b & c & d & e \\ q & p & o & i & s \\ q & q & p & r & r \end{matrix}$$

Map 2:

$$\begin{matrix} q & i & s & q & p \\ h & r & r & q & q \\ a & b & c & d & e \end{matrix}$$

and the following 3 maps are examples that do not contain the pattern:

Map 1:

$$\begin{matrix} a & b & c & d & e \\ q & p & o & i & s \\ q & j & p & r & r \end{matrix}$$

Map 2:

$$\begin{matrix} q & i & s & q & p \\ h & r & c & y & e \\ a & b & c & d & e \end{matrix}$$

Map 3:

$$\begin{matrix} q & h & r & r & q \\ a & b & v & d & e \\ p & b & i & s & q \end{matrix}$$

4. (10 pts) Please prove that your algorithm in the previous subproblem has an **worst case running time of** $O(K \times M \times N + g \times h)$.

Finally, How-How's team has successfully found the target map that is most likely to contain valuable mineral deposits. However, a team member accidentally deleted all data related to the planets. Fortunately, How-How is also a distinguished hardware engineer and is able to restore the target map from the hard drive, but they still don't know which planet is associated with the map. To find the planet, they reconstructed all maps of the K planets. Since the planets rotate, the measured map would be different from the old map. In other words, the new map of the target planet would be a rotation of the old map. This problem has been troubling How-How's team for days and the team members still cannot find an efficient solution to it.

5. (20 pts) Given your spectacular performance in previous tasks, How-How desperately hopes that you can find a solution to this devastating situation. Please design an algorithm that finds which of the K newly measured maps is the rotation of the target map with **worst case running time of** $O(K \times M \times N)$. Note that only the longitude value (horizontal index) changes for each point and the latitude value (vertical index) remain

the same. For example, given the maps

Map 1:

a	b	c	d	e
q	p	o	i	s
q	q	p	r	r

Map 2:

q	p	o	i	s
q	q	p	r	r
a	b	c	d	e

and target map

e	a	b	c	d
s	q	p	o	i
r	q	q	p	r

your algorithm should output "Map 1".

6. (10 pts) Please prove that your algorithm in the previous subproblem has an **worst case running time of** $O(K \times M \times N)$.

Problem 3 - DSA Founder How-How (100 pts)

How-How is not only the current head scientist of the Der-tian space agency but also the founder of **DinoSa Award(DSA)**.

The **DinoSa Award** (stylized as **DINO**, originally called **DinoSaur Award**), or just **DinoSaurs**, is an award presented by the Sorting Academy to recognize **"Outstanding Achievement in Sorting Algorithms"** of the DSA kingdom. The trophy depicts a gilded Dinosaur Egg.

- Each Dinosaur egg has a magic label saved inside the egg, which denotes the starting order. Each magic label l is an integer with $0 < l < 10^7$.
- There are N eggs in a row.

In this year's awards ceremony, since the manufacturer of the trophies ran a little behind, the delivery was postponed to the day before the ceremony.

Can you design the fastest algorithm in the world to sort the eggs for catching up on the start ceremony?

1. (10 pts) Now the sequence is $(4, 73, 184, 76, 299, 47, 9, 504)$, please illustrate the process of radix sort for ascending order step by step.

2. (20 pts) To simplify the problem, How-How offers a magical scanner that tells you the label of each egg in $O(1)$ time. Please design an $O(N)$-time-complexity algorithm that sorts the given sequence in ascending order, and specify its space complexity. (You can use any linear-sorting algorithm taught in-class.)

3. (20 pts) Unfortunately, the magical scanner crashed, so How-How tells you there is a magical libra. You can put two eggs on it, and the libra will tilt to the egg with the larger label in $O(1)$ time, but the libra is inside a tiny room.
   Please design an $O(N)$ time complexity and $O(N)$ space complexity algorithm to solve the type 1 sequence.

   - Type 1. For sequence $(s_i)_{i=1}^N$, there won't be any subsequence where $s_{i+k} < s_i < s_{i+j}$ with $k > j > 0$
   - For example: 31524 is not type 1, since $2 < 3 < 5$

   Hint. There is a deep hole inside the room so that you can put the egg inside it. There is also a magical fishing rod that can get the top egg from the hole in the blink of an eye.

4. (20 pts) Please design an $O(N)$ time complexity and $O(N)$ space complexity algorithm to solve the type 2 sequence.

   - Type 2. For sequence $(s_i)_{i=1}^N$, the length of the longest increasing subsequence is less than 3.
   - For example: 31524 is not type 2, since $1 < 2 < 4$, the length of this increasing subsequence is 3.

   Hint. Now the room is a little bigger so that the space is enough for two deep holes inside the room.

5. (30 pts) Please design an $O(N)$ time complexity and $O(N)$ space complexity algorithm to solve the type 3 sequence.

   - Type 3. For sequence $(s_i)_{i=1}^N$ and given K, the length of the longest increasing subsequence is less than K.
   - Type 2 is a special case of the type 3 with $K = 3$

   Hint. Now the room is a little bigger so that the space is enough for K deep holes inside the room.

Programming Part

Problem 4 - Magic Certification (100 pts)

Problem Description

Erina was a 19-year-old fairy who lived in the angustifolia forest. In the angustifolia forest, residents of the forest needed to obtain a magic certification before 20-year-old, or they would be expelled. Fairies invented innovative magic and perform it on the angustifolia magic assembly. Elder fairies award a magic certification if they felt satisfied with the magic invented by fairies.

For a magic certification, Erina racked her brains to create high-quality magic. However, she got a secret before the angustifolia magic assembly - Most elder fairies preferred symmetrical magic. Erina wanted to modify her magic with the lowest effort. Can you help her to convert the original magic to the shortest symmetrical magic by adding some characters in front or back of the original magic?

Input

The input contains a string s.

Output

You should output several lines. The first line outputs how many characters you need. Output the shortest palindrome you convert by adding characters in front or back of the original string s. If there is more than one converting method to satisfy the shortest palindrome, please print out all of them (from the least extra characters in front of the original string s to the most extra characters in front of the original string s.) You can refer the Sample Input2.

Constraints

- $0 \leq$ the length of the string $s \leq 10^7$
- s consists of ASCII codes from $0x21$ to $0x7E$ only

Subtask 1 (25 pts)

- $0 \leq$ the length of the string $s \leq 10^4$

Subtask 2 (25 pts)

- You only need to add characters **in front of** the string s, then you will convert string s to the shortest palindrome.

Subtask 3 (50 pts)

- No other constraints.

Sample Cases

Sample Input 1

```
aabc
```

Sample Output 1

```
2
cbaabc
```

Sample Input 2

```
baaaac
```

Sample Output 2

```
5
caaaabaaaac
baaaacaaaab
```

Sample Input 3

```
ababa
```

Sample Output 3

```
0
ababa
```

Problem 5 - Magic Certification II (100 pts)

(The setting follows the previous question.)
During the test, some elder fairies found that some magic are too similar, and they suspect there might be a plagiarism. Since any form of cheating, lying, or plagiarism will not be tolerated, they decided to investigate it.
For two strings with the same length, we call them "similar" if it at most one character is different. (Formally, two strings s, s' are similar if there exists at most one $i \in \mathbb{N}$ s.t. $s[i] \neq s'[i]$)
However, there are too many fairies pursuing the magic certification. The elder fairies can barely remember all the magic, but couldn't figure out whether there are similar magic. Can you determine for them?

Input Format

The first line contains three integers `k, l, flag`, indicating the number of magic and the length of the magic.
Each of the following `k` lines contain a string `s` with length `l`, indicating the magic performed.
The use of `flag` will be explained in the output section.

Output Format

If there are no similar magic, output a line "No" (without quotes).
Otherwise, output two lines: "Yes", following a line, depending on `flag`:

- If `flag = 0`, containing two integers i j, where the i-th magic is similar to j-th. (0-based) If there are multiple pairs of magic that are similar, output either of them.
- If `flag = 1`, print out one integer, indicating the number of similar pairs.

Constraint

- $1 \leq k \cdot l \leq 10^6$
- s consists of ASCII codes from `0x21` to `0x7E` only

Subtasks

Subtask 1 (20 pts)

- `flag = 0`
- $1 \leq k \cdot l \leq 10^4$

Subtask 2 (20 pts)

- `flag = 0`
- If two magic are similar, they are guaranteed to be exactly the same. That is, you only need to determine whether there exists same magic.

Subtask 3 (30 pts)

- `flag = 0`

Subtask 4 (30 pts)

- No other constraints

Sample Testcases

Sample Input 1

```
3 3 0
yee
yay
yee
```

Sample Output 1

```
Yes
0 2
```

Sample Input 2

```
7 4 0
qwer
qwxx
qxex
qxxr
xwex
xwxr
xxer
```

Sample Output 2

```
No
```

Sample Input 3

```
1 2 0
><
```

Sample Output 3

```
No
```

Sample Input 4

```
8 4 0
qwer
qwxx
qxex
qxxr
xwex
xwxr
xxer
xxxr
```

Sample Output 4

```
Yes
7 6
```

Sample Input 5

```
9 4 1
qwer
qwxx
qxex
qxxr
xwex
xwxr
xxer
xxer
xxxr
```

Sample Output 5

```
Yes
5
```