College of Computing and Informatics, Sungkyunkwan University

Problem Solving 2022 Fall
Assignment 2

Instructor: Jaemin Jo (`jmjo@skku.edu`)

October 28, 2022

1 Introduction

In this assignment, you will implement a puzzle solver for a game, `SKKU-Reverse`. Given an array X of N positive integers ($N \leq 100$), the goal of `SKKU-Reverse` is to find a sequence of M "reverse" operations to get a maximum score ($M \leq 100$).

Given an array X, the score of X is defined as the sum of the product of repeating elements and their repeating length. For example, the score of [1, 2, **3, 3**, 4] is 6 (2×3) since two 3s appear consecutively. Similarly, the score of [1, 2, **4, 4, 4**] is 12 (3×4) since there are three 4s in a row. Note that a single non-repeating number, such as 1 or 2 in the above example, does not increase the score.

If there are multiple sub-sequences of repeating elements, you can sum up the score of each sub-sequence. For example, the score of [1, 2, **3, 3**, **5, 5**] is 16 ($2 \times 3 + 2 \times 5$) since there are two 3s and two 5s. Similarly, the score of [1, **5, 5**, 2, **6, 6**] is 22 ($5 \times 2 + 6 \times 2$).

You need to use *reverse* operations to maximize the score. A *reverse*, $R(i, j)$, is defined as placing the elements in the contiguous subarray $X[i{:}j]$ in a reverse order. An array index starts from 0. For example, given X = [1, 2, **3, 1, 2,** 3], $R(2, 4)$ reverses the subarray [3, 1, 2], and after the reverse, X becomes [1, 2, 2, 1, 3, 3]. Note that the score of X increased from 0 (no repeating number) to 10 (two 2s and two 3s) after the reverse.

You need to apply reverse operations exactly M times. No more, no less. For example, in the previous example, if M is 2, you can apply the second reverse $R(1, 3)$ after $R(2, 4)$. The resulting array is [1, 1, 2, 2, 3, 3] whose score is 12, increased from 10.

Given N, X, and M, write a program that finds a sequence of reverses that maximizes the score.

2 Important Tip

It would be computationally infeasible to find the best sequence in the given time. The goal of this assignment is **NOT** to guarantee the best sequence but to find a "good" sequence. Start from straightforward, rule-based, and heuristic solutions and think about how to extend them to cover more general cases. I want you to observe a (very hard) problem space, get hunches and insights, and implement a partial but practical solution, and this is what actually happens in the real world. There is no spherical cow.

As a "perfect" solution would be infeasible, for every testcase, you will be given a partial score that is proportional to (the score earned by the sequence your program produced) / (the maximum score earned by anyone else in the class). **Never focus on finding a perfect solution.** Focus on getting partial scores by writing a program that "moderately" works well.

3 Input

On the first line, N and M are given ($1 \leq N, M \leq 100$). On the second line, the elements of X are given, separated by a single space character ($1 \leq X[i] \leq 100$).

4 Output

Print out a sequence of M reverses to maximize the score, one on a line. Each reverse is described as two integers, i and j. You MUST print out exactly M reverses ($2 \times M$ integers in total) even though the maximum score can be achieved using fewer than M reverses. You will get no points if your program prints 1) fewer than M reverses or 2) an index that is out of range (e.g., less than 0).

Hint: If you can achieve the maximum score using fewer than M reverses, you can print out a reverse that has no effect, e.g., $R(0, 0)$, to make the sequence have M reverses.

5 Examples

Example Input

```
6 2
1 2 3 1 2 3
```

Example Output 1

```
2 4
1 3
```

The final array after the two reverses is [1, 1, 2, 2, 3, 3], whose score is 12. This is the actual maximum score you can get. Therefore, you will get a perfect score for this testcase.

Example Output 2

```
4 5
2 3
```

The final array after two reverses is [1, 2, 1, 3, 3, 2] whose score is 6. This is half of the maximum score (12). Therefore, you will get 50% of the score given to this testcase.

Example Output 3

```
1 1
1 1
```

The two reverses do not change the array at all, so the final array after the reverses is still [1, 2, 3, 1, 2, 3], whose score is 0. You will get no score for this testcase.

6 Rules and Tips (READ THIS CAREFULLY!)

- Your code must be in a single source file, **main.c**. Do not create any separate header or source files. A skeleton code file will be given in iCampus. Feel free to modify it.
- Your code will be compiled by gcc 11.1. You must test your implementation on Goorm.
- You can only use the standard functions of C. See the list of standard header files: `https://en.cppreference.com/w/c/header`
- Your program will be automatically graded by a program. Therefore, be careful about the input and output formats. Do not print extra characters/spaces/newlines that are not specified in this document. such as "Welcome", "Enter a command>", or "Available commands are...".
- The right behavior of your program is **undefined** for cases that are not specified in this document.
- The program must exit with code 0, i.e., returning 0 at the end of the main function.
- Use only `scanf("%s", ...)` to read a string input. Do not use `"%c"` or `getchar()`. Do not use non-standard, non-cross-platform, should-be-avoided, and I-didn't-teach-you statements such as `scanf("%[^\n]*c", ...)`.
- Any array that exceeds 1 KB must be defined globally or allocated dynamically. No `int main() { int a[300][300]; ... }`.
- Do not use variable length arrays. No `int main() { int n; ... int a[n]; ... }`.
- Your program will be given **2** seconds and 256 MB of memory for each testcase.

7 Submission

The deadline is **November 15th 23:55 KST**. You need to submit one C file **main.c** to iCampus.

8 Copyright

You will hold the copyright of your work. I will not copy/distribute/modify your work except for the grading.

9 Plagiarism

Do NOT copy others' source codes. Do NOT transfer any part of your code through email, messenger, text, etc. Both cheater and code-giver will get a penalty if their codes are similar enough.