Data Structures and Algorithms, Spring 2024
Homework 0

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

Due: 13:00:00, Tuesday, March 5, 2024

Rules and Instructions

Welcome to DSA 2024 from your teaching team with 23 TAs and 2 instructors. Please start the exciting journey by carefully reading the following rules for this homework set. Many rules will also be applied to future homework sets.

- 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.

- 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 (including chatGPT) 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.

- In Homework 0, the problem set contains 3 problems. The set is divided into two parts, the self-graded part (Problem 1) and the programming part (Problems 2 and 3).

- For problems in the programming part, you should write your code in C programming language, and then submit the code via the *DSA Judge*:

  https://dsa2024.csie.org/.

  Each day, you can submit up to 5 times for each problem. The judge system will compile your code with

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

- For debugging in the programming part, we strongly suggest you produce your own testcases with programs and/or use tools like `gdb` or compile with flag `-fsanitize=address` to help you solve issues like illegal memory usage. For `gdb`, you are strongly encouraged to use compile flag `-g` to preserve symbols after compiling.

  Note: For students who use IDE (VScode, DevC++...) you can modify the compile command in the settings/preference section. Below are some examples:

  - `gcc main.c -O2 -std=c11 -fsanitize=address # Works on Unix-like OS`
  - `gcc main.c -O2 -std=c11 -g # use it with gdb`

- The score of the part that is submitted after the deadline will get some penalties according to the following rule:

$$\textit{Late Score} = \max\left(\left(\frac{5 \times 86400 - \textit{Delay Time}(\textit{sec.})}{5 \times 86400}\right) \times \textit{Original Score}, 0\right)$$

- The purpose of Homework 0 is to communicate our *expected background* for this class. If you cannot solve Homework 0 within a reasonable amount of time *by yourself*, it is very likely that you cannot solve those more challenging homework sets in the future. Then, you are *strongly encouraged* to not take this class (though you are always welcome to audit the class to learn at your own pace).

- Following the purpose above, the TAs will *not* hint to you how to solve the problems in Homework 0 via emails or during their TA hours (unlike what they will do for other homework sets in the future). They will only clarify any ambiguity in the problem descriptions. If you need discussions on any issues about Homework 0, please go to the Discord channel and discuss the issues with your classmates.

- If you really need email clarifications on the problem descriptions, please include the following tag in the subject of your email: `"[HW0.Px]"`, specifying the problem where you have questions. For example, `"[HW0.P2] Can a[n] be negative?"`. Adding these tags allows the TAs to track the status of each email and to provide faster responses to you.

Self-Graded Part

Problem 1 (solution)

Consider the following programming task:

> Given a and b, each being a non-negative integer that is no more than 10^{1000}, please output $a \cdot b$.

Student A attempted to write some C code to solve the task above but was trapped with some bugs. So ze came to the DSA TA hour for help. The TA found at least **five** bugs in zir codes. Can you find and fix those bugs too?

The problem is designed to be self-graded. We will release the solution to you after the homework is due. It does not need to be submitted but is meant to hint you some common bugs. If you use chatGPT (yes, we tried), it can identify some but not all of the bugs—that is, your human intelligence will still be critical in your future debugging!

(https://pixabay.com/photos/code-code-debug-learn-sleep-repeat-4918185/)

Please check the buggy code on the next page :-)

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char* multiply(char* num1, char* num2) {
    int len1 = strlen(num1);
    int len2 = strlen(num2);

    char* ret = malloc(sizeof(char) * (len1 + len2 + 1));
    memset(ret, 0, len1 + len2 + 1);

    for (int i = len1 - 1; i >= 0; --i) {
        for (int j = len2 - 1; j >= 0; --j) {
            int mul_ret = (num1[i] - '0') * (num2[j] - '0');
            ret[i + j + 1] += mul_ret % 10;
            ret[i + j] += mul_ret / 10;
        }
    }

    for (int i = 0; i < len1 + len2 + 1; ++i) {
        ret[i] += '0';
    }

    return ret;
}

int main() {
    char num1[1001], num2[1001];

    scanf("%s %s", num1, num2);
    char* ans = multiply(num1, num2);

    while (*ans == '0') ans++;

    printf("%s\n", ans);
    free(ans);

    return 0;
}
```

For each bug you find, please specify where the bug is (the line number) and how to fix it. Once all the bugs are corrected, the code is anticipated to compile and execute correctly. Take the "Hello World" code with a bug for example:

```c
1 #include <stdio.h>
2 #include <string.h>
3 
4 int main(){
5     char str[13] = "Hello World!";
6     printf("%d\n", str);
7     return 0;
8 }
```

Please write your answer in the following format:

```
Line 6: printf("%s\n", str);
```

Solution

- line 15: `ret[i+j+1]` might exceed 10 here, we should take this case into consideration.
- line 20: `ret[len1 + len2]` is `\0` and shouldn't be modified, so we should modified this line to `for(int i = 0; i < len1 + len2; ++i)`
- line 28: The array sizes should be modified to 1002 for the space of `\0`.
- line 33: The code won't output anything if the answer equals zero.
- line 36: The pointer `ans` should be moved to the start of the string before being freed.

Check the next page for the correct version:

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

char* multiply(char* num1, char* num2) {
    int len1 = strlen(num1);
    int len2 = strlen(num2);

    char* ret = malloc(sizeof(char) * (len1 + len2 + 1));
    memset(ret, 0, len1 + len2 + 1);

    for (int i = len1 - 1; i >= 0; --i) {
        for (int j = len2 - 1; j >= 0; --j) {
            int mul_ret = (num1[i] - '0') * (num2[j] - '0');
            mul_ret += ret[i + j + 1];
            ret[i + j + 1] = mul_ret % 10;
            ret[i + j] += mul_ret / 10;
        }
    }

    for (int i = 0; i < len1 + len2; ++i) {
        ret[i] += '0';
    }

    return ret;
}

int main() {
    char num1[1002], num2[1002];

    scanf("%s %s", num1, num2);
    char* tmp = multiply(num1, num2);
    char* ans = tmp;

    while (*ans == '0') ans++;

    printf("%s\n", *ans == '\0' ? "0" : ans);
    free(tmp);

    return 0;
}
```

Problem 2 (solution)

Problem Description

For an array $\mathbf{a}$ with N integers $\mathbf{a}[1], \mathbf{a}[2], \cdots, \mathbf{a}[N]$, we say this array is **beautiful** if it satisfies the following condition:

$$(\mathbf{a}[n+1]-\mathbf{a}[n])(\mathbf{a}[n+2]-\mathbf{a}[n+1]) < 0, \text{ for all } 1 \leq n \leq N-2.$$

Consider the permutations of any given integer array $\mathbf{a}$, please output all the **beautiful** permutations of $\mathbf{a}$.

Input

The first line includes an integer N, representing the length of the array. The second line includes N integers, representing the elements of the array $\mathbf{a}[1], \mathbf{a}[2], \ldots, \mathbf{a}[N]$. All integers are separated by spaces. *Note that* $\mathbf{a}$ *can contain repeated elements and is not guaranteed to be ordered.*

Output

In the first line, output a number M that indicates the number of *different* **beautiful** permutations of $\mathbf{a}$. Note that $M = 0$ if there is no such permutation.

For each of the following M lines, output the *different* **beautiful** permutations in lexicographical order, from the smallest to the largest.[1] The lexicographical order defines a length-N array $\mathbf{p}$ to be smaller than another same-length array $\mathbf{q}$ if and only if $\mathbf{p}[n] < \mathbf{q}[n]$ for the first n in $1, 2, \ldots, N$ such that $\mathbf{p}[n] \neq \mathbf{q}[n]$. The **beautiful** permutation should be outputted with N integers, separated by spaces.

Constraints

- $1 \leq N \leq 20$
- $-10^9 \leq \mathbf{a}[n] \leq 10^9$ for every $n \in \{1, 2, \ldots, N\}$
- The number of **beautiful** permutations (M) is no more than 2×10^5.

[1]https://en.wikipedia.org/wiki/Lexicographic_order

Subtasks

Subtask 1 (10 pts)

- $1 \leq N \leq 10$

Subtask 2 (20 pts)

- $-10^4 \leq \mathbf{a}[n] \leq 10^4$ for every $n \in \{1, 2, \ldots, N\}$

Subtask 3 (20 pts)

- no other constraints

Sample Testcases

Sample Input 1

```
3
3 2 1
```

Sample Output 1

```
4
1 3 2
2 1 3
2 3 1
3 1 2
```

Sample Input 2

```
7
7 7 7 7 7 14 49
```

Sample Output 2

```
0
```

Sample Input 3

```
3
7 7 49
```

Sample Output 3

```
1
7 49 7
```

Sample Input 4

```
2
1 2
```

Sample Output 4

```
2
1 2
2 1
```

Hints

- Any array with length $N \leq 2$ satisfies all the beauty constraints and hence should be considered **beautiful**.
- You may get TLE (Time Limit Exceeded) if you only enumerate every permutation in a brute-force manner without considering any cut-off. That is, you should try to stop spending time on permutations that are **not** beautiful.
- While $\mathbf{a}[n]$ does not exceed the range of 4-byte integers, their difference and multiplication may not stay within 4 bytes. So using some longer integer format such as `long long` can be helpful.

Solution

Please check this github repo link for the solution: `https://github.com/NTU-CSIE-DSA/hw0-ZigZagPermutation`

Problem 3 (solution)

Problem Description

Given N decks of cards, all initially empty, we want to perform four types of operations on them:

1. Add a card numbered u to the top of deck i.
2. Remove a card from the top of deck i.
3. Move all cards from deck i onto the top of deck j. If deck i has cards `3 4` from top to bottom, and deck j has cards `2 5 6 9` from top to bottom. The resulting deck j will have `3 4 2 5 6 9` from top to bottom.
4. Merge deck i into deck j in a shuffling manner. If deck i has cards `3 4` from top to bottom, and deck j has cards `2 5 6 7` from top to bottom. The resulting deck j will have `3 2 4 5 6 7` from top to bottom.

The last operation merges two decks of cards according to the following procedure:

1. Move the top card from deck i to the bottom of the temporary deck if deck i is not empty.
2. Move the top card from deck j to the bottom of the temporary deck if deck j is not empty.

3. If one of the two decks is empty, directly move all the cards in the other deck, from top to bottom, to the bottom of the temporary deck. Otherwise, return to step 1.
4. Move the whole temporary deck to deck j.

After all operations, please output the cards in all the decks.

Input

The first line includes two integers, N and M, representing the number of decks and operations, respectively. The next M lines include 2 or 3 integers. The first integer, t, represents the operation type:

- Type 1 is for the “Add” operation, followed by two integers, i and u, representing the target deck and the card number, respectively.
- Type 2 is for the “Remove” operation, followed by only one integer, i, representing the target deck.
- Type 3 is for the “Move” operation, followed by two integers, i and j, representing the source and destination decks, respectively.
- Type 4 is for the “Merge” operation, followed by two integers, i and j, representing the source and destination decks, respectively.

Output

The output should consist of N lines. For the i-th line, first output c_i, the number of cards in the i-th deck. Then, output c_i numbers representing each card in deck i from top to bottom. All numbers should be separated by a space.

Constraints

- $1 \leq N, M \leq 10^5$
- $1 \leq i, j \leq N$
- $1 \leq t \leq 4$
- $1 \leq u \leq 10^9$
- When executing the “Remove” operation, deck i will not be empty.
- When executing the “Move” or “Merge” operation, $i \neq j$.

Subtasks

Subtask 1 (10 pts)

- $1 \leq N, M \leq 10^3$

Subtask 2 (5 pts)

- only "Add" and "Remove" operations

Subtask 3 (10 pts)

- only "Add", "Remove", and "Move" operations

Subtask 4 (25 pts)

- no other constraints

Sample Testcases

Sample Input 1

```
2 5
1 1 1
1 1 2
1 2 3
1 2 4
2 1
```

Sample Output 1

```
1 1
2 4 3
```

Sample Input 2

```
2 5
1 1 1
1 1 2
1 2 3
1 2 4
3 1 2
```

Sample Output 2

```
0
4 2 1 4 3
```

Sample Input 3

```
2 7
1 1 1
1 1 2
1 2 3
1 2 4
1 2 5
1 2 6
4 1 2
```

Sample Output 3

```
0
6 2 6 1 5 4 3
```

Hints

- You can use *linked lists* to effectively solve this problem. If you are not familiar with what a linked list is, you can review it with this problem on the Judge Girl system:

  `http://140.113.150.142/problem/0/11258`

- When one of the decks is empty, it would be more efficient to directly concatenate the remaining deck to the bottom of the temporary deck.

Solution

Please check this github repo link for the solution: `https://github.com/NTU-CSIE-DSA/hw0-DecksOfCards`