Data Structure and Algorithm, Spring 2024

Mini-Homework J

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

Due: 13:00:00, Tuesday, June 11, 2024

PLEASE NOTE THE DIFFERENT LATENESS POLICY IN RED BELOW

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.
- 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 each mini-homework, we will have one programming problem that is closely connected to what we have taught in the class. The purpose of the problem is for you to *practice* what you have learned, before conquering the more challenging problems in other homework sets.
- Like other programming problems, 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 test-cases 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:

$$Late\ Score = \max\left(\left(\frac{1 \times 86400 - Delay\ Time(sec.)}{1 \times 86400}\right) \times Original\ Score, 0\right)$$

- If you have questions about this mini-homework, please go to the the Discord channel and discuss the issues with your classmates. This is *strongly preferred* because it will provide everyone with a more interactive learning experience. If you really need an email answer, please follow the rules outlined below to get a fast response:

  - The subject should contain one tag, `"[hwe]"`, specifying the problem where you have questions. For example, `"[hwe] Will there be a repeated number?"`. 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 are discouraged and may not be reviewed.

Programming Part

Disjoint Sets (20 pts)

Problem Description

In this problem, we ask you to implement the standard disjoint-set forest data structure and algorithm presented in the lecture and in the textbook. The pseudo codes of the three main operations of the disjoint set data structure, MAKE-SET, UNION, and FIND-SET, are as follows. The pseudo code implements the heuristics to improve the running time: *union by size* and *path compression*. You also need to implement a PRINT-PATH function, which prints all nodes of the path from the given node to the root of the disjoint-set tree that has the given node, following the parent link of each visited node.

Note that you need to follow exactly how to union two sets (by their sizes) and how to perform path compression, in order to have PRINT-PATH generate the right answer.

Assume the object x which forms the disjoint sets are represented by distinct postive integers, $1 \leq x \leq N$. Printing an object x would output that integer. In addition, assume that MAKE-SET is executed for each of the N objects (i.e., for these N integers) as part of the initialization process.

```
MAKE-SET(x)
1  x.p = x
2  x.rank = 0

UNION(x, y)
1  v = FIND-SET(x)
2  w = FIND-SET(y)
3  if v ≠ w
4        LINK(v, w)

FIND-SET(x)
1  if x ≠ x.p
2        x.p = FIND-SET(x.p)
3  return x.p
```

LINK(x, y)

```
1  if x.rank > y.rank
2        y.p = x
3  else
4        x.p = y
5        if x.rank == y.rank
6              y.rank = y.rank + 1
```

PRINT-PATH(x)

```
1  k = x
2  while k.p ≠ k
3        print k
4        k = k.p
5  print k
```

Input

The first line includes N. The second line includes M, which specifies the number of operations in the input. In each of the following M lines, it will be in one of these three formats:

1. `F x`. This means you should execute FIND-SET(x).
2. `U x y`. This means you should execute UNION(x, y).
3. `P x`. This means you should execute PRINT-PATH(x).

The x and y above are integers representing the sets. A single white space is used to separated numbers and after the characters `F`, `U`, and `P`.

You should execute all operations according to the given order.

Output

For each `P` operation (i.e., PRINT-PATH) given, output in one line, the output of PRINT-PATH, which include all the nodes in the path from the given nodes x to the root of the disjoint-set tree that x belongs to, in this format: $x\ x_1\ x_2\ \ldots x_n$ Consecutive numbers are separated by a white space.

Constraint

- $1 \leq N \leq 1000000$
- $1 \leq M \leq 1000000$

Sample Testcases

Sample Input 1

```
3
6
F 1
U 3 1
U 3 3
U 2 2
F 2
P 3
```

Sample Output 1

```
3 1
```

Sample Input 2

```
5
10
U 4 5
F 5
P 4
P 5
P 3
U 5 4
U 2 4
F 2
U 2 5
P 2
```

Sample Output 2

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

Sample Input 3

```
5
10
U 3 4
U 1 4
F 4
P 1
P 4
P 3
F 1
P 2
P 1
P 4
```

Sample Output 3

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