Data Structure and Algorithm, Spring 2022

Homework 4

Due: 13:00:00, Wednesday, June 8, 2022

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

Blue Update on Problem 1 at 2022/05/20 16:00

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 4, the problem set contains a special Problem 0 (see below) and 4 other problems. The set is divided into two parts, the non-programming part (Problems 0, 1, 2) and the programming part (Problems 3, 4).
- 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 (2022/05/17-2022/05/24). 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, `"[hw4]"` and `"[Px]"`, specifying the problem where you have questions. For example, `"[hw4][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.*

Non-Programming Part

Problem 1 - Hash Table and Disjoint Set (100 pts)

Hash Table

While Alex works hard in the NTU Main Library, he found that there are many reservation-based self-study rooms. Right after burning the midnight oil, when Alex has the most inspiration, he found that these self-study rooms look very much like a table from the data structure perspective. Alex starts to dream about the infinite possibilities beyond these rooms.

1. (20pts) Imagine that now the reservation strategy is based on the hash functions we taught in the class. Students come up with a lucky number. A machine, when given this number, will tell them which self-study room he or she should use, in a manner similar to a hash table. More formally, insert into the hash table the following keys in the given order: [42, 11, 25, 1, 56, 70, 19], with the number of rooms $N = 7$. Let

$$h_1(k) = 10k \mod N$$
$$h_2(k) = 1 + (3k \mod (N-1))$$

be the two hash functions.

Please specify in a step-by-step manner how the keys are inserted into the hash table using open addressing with

(a) linear probing with hash function $h_1(k)$,

(b) double hashing with primary hash function $h_1(k)$ and secondary hash function $h_2(k)$.

Please fill out the two tables below, showing the content of the hash table after each insertion. No explanation is needed.

(a) Open addressing with linear probing

keys to be inserted \ index	0	1	2	3	4	5	6
42	42						
11							
25							
1							
56							
70							
19							

(b) Open addressing with double hashing

keys to be inserted \ index	0	1	2	3	4	5	6
42	42						
11							
25							
1							
56							
70							
19							

2. (15pts) Suppose now there are $N = 17$ self-study rooms. We now look at the interesting ON/OFF patterns of the light of the rooms. For example, at 9:00, only room 1, 3, 5, 7, and 17 have the lights ON, and at 9:05, only room 1, 3, and 4 have the lights ON.

   More formally, a *light pattern* is the set of the numbers of the rooms with the light ON, e.g., $\{1, 3, 5, 7, 17\}$. Alex would like to store *the number of occurrences* of all possible light patterns in an array. The only question is how to translate a particular light pattern into the index of the array (which is an non-negative integer). In addition, the size of the array should not be larger than 133333.

   Please help Alex to design this translation function, and explain why your method fits the requirements.

3. (15pts) Studying consumes lots of energy (formally, ATP). Now, imagine that every room has an "energy level" indicating the remaining energy of the student studying inside. Assume that there are N self-study rooms. Alex wants to know the maximum number of **consecutive** rooms such that the sum of the energy levels of these rooms is E.

   Formally, the energy levels are given in an array, where the index is the room number and the value is an integer indicating the energy level. For example, given the energy level

array $[1, -3, 2, 5, 4]$, the maximum number of consecutive rooms with energy sum $E = 5$ is 4, since $1 + (-3) + 2 + 5 = 5$ and this represents the case with the maximum number of consecutive rooms.

Please design an $O(n)$-time and $O(n)$-extra-space algorithm to find the maximum number of consecutive rooms as described. You can directly use `Get(k)` and `Insert(k, v)` (`k` is key, `v` is value) operations of a hash table and assume that no collision occurs.

4. (15pts) Next, assume that the reservation machine employs a **uniform** hash function $h(k)$ that maps input k into a hash table of size N. Suppose there are a total of M students devising their lucky numbers as the keys to be inserted into the hash table independently. What is the expected number of collisions in terms of N and M?

Disjoint Set

5. (20pts) Suppose we want to use the following pseudo code to perform a depth-first traversal on a binary tree with N nodes. Each node u has a lucky number denoted by $u.val$. Specially, for each node, we will merge the heaps obtaining from the children and insert the node's lucky number as well. We start the `DFS` from the `root` node.

```
def DFS(node u):
    if u is NULL:
        return new heap()
    heap_ptr = MERGE_HEAP(DFS(u->left), DFS(u->right))
    heap_ptr->insert(u->val)
    return heap_ptr

final_heap = DFS(root) # A heap with N elements.
```

In the above pesudo code, `MERGE_HEAP` will take the pointers of two heaps as input and return the pointer to the merged heap. Please write a function `MERGE_HEAP(heap a, heap b)` to merge the heaps, so that the overall time complexity of `DFS` is $O(N \log^2 N)$. Note that you are not allowed to build extra heaps inside `MERGE_HEAP`. Briefly explain why your `MERGE_HEAP` meets the requirements.

In subproblem 6, you can use a disjoint set data structure with union by size and path compression technique. You may use the following function without any detailed explanation:

- `MAKE_SET(x)`: Create a set with one element `x`.

- `UNION(x, y)`: Merge the set that contains `x` and the set that contains `y` into a new set, and delete the original two sets that contains `x` and `y`, respectively.
- `FIND_SET(x)`: Return the index of the set that contains `x`.

6. (15pts) Many classes need the students to form groups to work on reports. However, forming groups is not a trivial task. People have preferences. Let us use an array of pairs, with size M to indicate the mutually-dislike relationship. For example, the array $[(1, 3), (2, 4)]$ means students 1 and 3 dislike each other, and students 2 and 4 dislike each other, too. Now, suppose you are required to partition all N students into exactly two groups, such that no two students in the same group dislike each other.

   Please design an $O((M + N)\alpha(N))$-time algorithm to determine whether it is possible to form the two groups as described. Your algorithm should also output the final group formation as well. If there are multiple possible formation, output any one of them. Please provide the pseudo code of your algorithm.

   (Hint: Can the people you dislike, be in a group?)

Problem 2 - Red-Black Tree (100 pts)

You, unbeknownst to your classmates, are actually a gardening hobbyist who owns a small garden located at a secret corner in the NTU campus. Every day, after long hours of classes and coding sessions, you would go to the garden to take care of your plants—especially the mulberry trees.

Note. In subproblem 1 and 2, we are talking about binary trees in general. That is, there's no distinction of color on the nodes. (Which makes a lot of sense: the mulberries are not ripe yet, so they are all green.)

1. (10pts) Consider a *complete* binary tree of n nodes, where $n \geq 1$ (here, the term *complete binary tree* is used in the same way as you've learned in class). How many possible left rotations and right rotations are there, respectively? Briefly explain your reasoning.

2. (25pts) Given an *arbitrary* binary tree of n nodes, we know that it can be transformed into a right-going chain using at most $n - 1$ right rotations. Design an $O(n)$-time algorithm to do so; you can call the function RIGHT-ROTATE directly inside your algorithm. To make life easier, you may take for granted that RIGHT-ROTATE runs in $O(1)$-time, and you are allowed to call it for more than $n - 1$ times, as long as your algorithm is able to yield the correct result in $O(n)$-time. After writing down the algorithm, you need to explain why it meets the time complexity requirement.

With so many chores and so many responsibilities occupying your everyday schedule, you barely notice how fast time passes: in the meantime you're striving to finish DSA homework 1, 2, and 3, your mulberry trees are also growing...

One day you find out that the mulberries have ripened, as some of them are starting to turn red, and others are turning black. You have also discovered that, whenever a tree is leaning towards one side too heavily, it will adjust the direction in which the branches should grow.

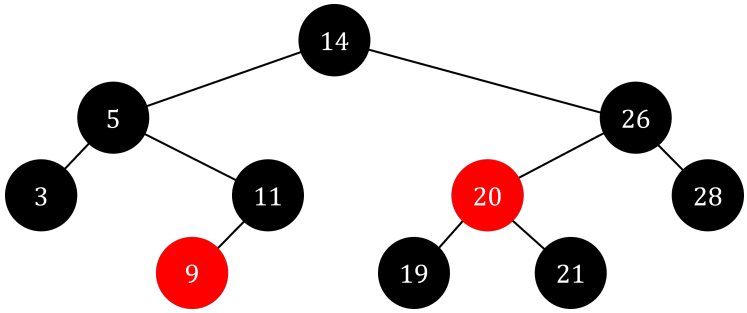
Note. In the subproblems below, the implementation of red-black trees follows exactly as how the algorithms are described in our textbook. For example, in the case where you are deleting a node z from a tree and z has two children, you should pick z's successor (i.e. the smallest node in z's right subtree), instead of its predecessor, as the one to be put into z's original position, even though the latter implementation may also give you a valid red-black tree.

3. (20pts) Prove or disprove the statement: For all red-black trees, it's impossible to do exactly one rotation on any of its nodes (without modifying the colors), while maintaining

its red-black properties. More formally, we cannot find any red-black tree T and any node y in T, such that the resulting tree T' after doing either one of LEFT-ROTATE(T, y) or RIGHT-ROTATE(T, y) can result in T' that can satisfy all the red-black properties and is still a valid red-black tree.

Looking at the flourishing trees that you have put your heart into, you begin to imagine: what shapes will the trees possibly form? What if you harvest some mulberries now?

4. (10pts) "Your favorite tree" looks like this:



   Insert a node with value 10 into "your favorite tree" and draw the resulting tree. No additional explanation is needed.

5. (10pts) Still starting with "your favorite tree" (that is, assume node 10 has never been inserted), please delete node 3, and then delete node 26. Draw the resulting tree; no explanation is required.

You are really delighted to witness the liveliness and diversity of your trees. After some pondering, you realize that you can create even more patterns of the branches by carefully pruning the trees!

6. (25pts) Give an example of a red-black tree T such that it satisfies the following statements. It is not possible to construct the tree T using only insertions, yet it can be constructed correctly (i.e., with correct shape and node colors) if both insertion and deletion can be used.

   Please come up with such a red-black tree T, draw it down, write a sequence of operations consisting of INSERT and DELETE operations, with which T can be constructed. Finally, briefly describe why it is not possible to build T with only insertions.

Problem 3 - Purple Cow Revenge (100 pts)

Problem Description

`One day, Purple Cow will be back.`

Time passes, the 118 alley no longer looks like before, but becomes a "MERGE" mode. If a shop goes bankrupt, it would merge with another one. The boss of Purple Cow, Mr. Pao, noticed this, and want to record it.

Mr. Pao also noticed that some merges could result in an unstable state. For instance, when steak meets Vanilla Ice, it would explode, causing some trouble in the 118 alley. But the phenomenon is too complicated, please help Mr. Pao record the whole process.

`One and only one` situation happens in one day. The day starts from day_1.

Input Format

The first line contains two integers, N, representing the numbers of shops, and, M, the number of days for data collection. The shops are denoted by $(shop_1, shop_2, ..., shop_N)$. Each of the next M lines represent what each day happens and is given in one of the following formats:

1. `merge` is followed by two integers i, j separated by spaces. This means that $shop_i$ merge into $shop_j$. Note that if $shop_m$ was merged by $shop_i$ before the current `merge` operation takes place, $shop_m$ would also become part of $shop_j$ after the current operation.

2. `query` For each query your program needs to output a line containing a single number representing the current number of shops in the 118 alley.

3. `boom` followed by a integer k. This means a Steak + Vanilla Ice event; there will be an instant explosion, which the status of all shops in the 118 alley is reverted back to day k (after the operation on that day happens).

Output Format

For each *query* you need to output one line, representing the number of shops in the 118 alley at that time.

Constraint

1. $1 \leq N, M \leq 1 \cdot 10^6$
2. for `merge` i j , $1 \leq i, j \leq N$, $i \neq j$
3. for `boom` in day d, $0 \leq k < d$, $k = 0$ represents the initial state (all shop are pairwise independent)

Subtasks

Subtask 1 (15 pts)

- $N, M \leq 1000$
- no `boom` situation

Subtask 2 (25 pts)

- no `boom` situation

Subtask 3 (25 pts)

- $N, M \leq 1000$

Subtask 4 (35 pts)

- No ⊕ther constraints!

Sample Cases

Sample Input 1

```
4 7
query
merge 1 2
query
merge 3 4
query
merge 1 3
query
```

Sample Output 1

```
4
3
2
1
```

Sample Input 2

```
4 11
merge 1 2
merge 1 3
query
boom 1
query
merge 1 3
merge 2 4
boom 4
query
boom 0
query
```

Sample Output 2

```
2
3
3
4
```

Hints

- If you want to know why steak + Vanilla Ice would cause an explosion. Please refer to .
- Let's Merge it together!

Problem 4 - COOL Server (100 pts)

Problem Description

Serving thousands of students, NTU COOL has N machines to execute programs. In order to check whether the machines run normally, the administrators may choose a block of machines (with consecutive ID) and reboot them **one by one**. At the beginning, assume that the i-th machine needs a_i second(s) to reboot. Long down time causes inconvenience, so the administrators must know exactly how much time the machines need to reboot.

It seems like an easy thing. However, during the semester, the NTU COOL team may change the positions of the machines, buy new ones and (or) discard a broken one. Moreover, they may update a block of machines (with consecutive IDs) to improve the user experience. With these operation, estimating the required time to reboot machines becomes difficult. Please help them to compute the required time to reboot based on the machine maintenance records.

Input Format

The first line contains two integers N, Q, representing the number of machines, and the number of records, respectively. Let the 1-st machine be the leftmost one and the N-th machine be the rightmost one.

The next line contains N positive integers $a_1, a_2, \ldots, a_N$, representing the required time for the i-th machine to reboot, in second.

Each of the next Q lines is given in one of the following formats:

- `1 p k`: A new machine which needs k seconds to reboot is placed between the p-th machine and the $(p+1)$-th machine. If $p = 0$, the new machine is placed to the left of the originally 1-st machine. If $p =$ the number of machines, the new machine is placed to the right of the originally rightmost machine.
- `2 p`: The p-th machine will be retired and discarded.
- `3 l r`: The order of the machines from the l-th to the r-th is reversed, which would swap the l-th machine and the r-th machine, swap the $(l+1)$-th machine and the $(r-1)$-th machine, and so on.
- `4 l r x y`: Swap the block of machines between the l-th and the r-th and the block of machines between the x-th and the y-th. Note that the number of machines in these two specified blocks are **NOT necessarily the same**.

- `5 l r k`: For any machine between the l-th and the r-th, if more than k second(s) is required to reboot, update it so that only k second(s) is needed to reboot.
- `6 l r`: Output a line with an integer representing the amount of time required to reboot the machines from the l-th to the r-th one by one, in second.

Output Format

For each query, output a line with an integer representing the amount of time required to reboot the machines from the l-th to the r-th one by one, in second.

Constraint

- $1 \leq N, Q \leq 10^5$
- all the indices in the operation (p, l, r, x, y) are legal and form a proper range
- for operation `3,4,5,6`, $l \leq r, x \leq y$
- for operation `4`, $[l, r]$ and $[x, y]$ do not overlap
- $1 \leq a_i, k \leq 10^9$

Subtasks

Subtask 1 (5 pts)

- $N, Q \leq 1000$
- Only operation `6`

Subtask 2 (5 pts)

- Only operation `6`

Subtask 3 (5 pts)

- Only operation `1,2,3,4,5`

Subtask 4 (5 pts)

- Only operation `1,6`

Subtask 5 (5 pts)

- Only operation `2,6`

Subtask 6 (5 pts)

- Only operation `1,2,6`

Subtask 7 (10 pts)

- Only operation `3,6`

Subtask 8 (10 pts)

- Only operation `4,6`

Subtask 9 (10 pts)

- Only operation `5,6`

Subtask 10 (20 pts)

- Only operation `3,4,5,6`

Subtask 11 (20 pts)

- No other constraints!

Sample Cases

Sample Input 1

```
5 7
4 8 7 6 3
6 3 5
1 0 2
6 1 2
1 6 1
6 6 7
2 3
6 1 6
```

Sample Output 1

```
16
6
4
23
```

Sample Input 2

```
5 10
4 8 7 6 3
3 2 4
6 3 5
1 0 2
5 2 4 1
6 1 2
1 6 1
4 6 7 1 3
6 6 7
2 3
6 1 6
```

Sample Output 2

```
18
3
2
16
```

Hints

- The Problem Description is actually a noise. Getting rid of it may be useful.
- One good strategy is to split the subtasks and use different ways to solve them, and then merge all the method together to solve the whole problem.