Data Structure and Algorithm, Spring 2018

Homework 3

Release: Tuesday, May 8, 2018

Due: 23:59:59, Sunday, May 20, 2018

TA email: dsa1@csie.ntu.edu.tw

Rules and Instructions

- In homework 3, the problem set contains 5 problems and is divided into two parts, non-programming part (problem 1, 2, 3) and programming part (problem 4, 5).
- Please go to *DSA Judge* (`https://dsa.csie.org`) to complete this homework. If you have any problem, check out the uploading tutorial in problem 0.
- For problems in non-programming part, you should combine your solutions in ONE pdf file, and then hand it in via the judge system. You can either type them or write on papers and scan them. If you choose the latter, please make sure that your writing is recognizable and clear enough, otherwise you might receive some penalties. The pdf file should contain your STUDENT ID and your NAME. If not, you would get minor penalties. Your solution must be as simple as possible. At the TAs' discretion, too complicated solutions will be counted as wrong.
- For problems in programming part, you should write your code in C programming language, and then hand them in via the judge system. You have 5 chances per day for each problem. The compilation command of the judge system will be `gcc main.c -static -std=c11 -O2 -lm`.
- Discussions with others are encouraged. However, you should write down your solutions in your own words. In addition, for each problem, you have to specify the references (the Internet URL you consulted with or the people you discussed with) on the first page of your solution to that problem.
- Do NOT lend your judge system account to others or borrow other students' accounts. Both cases would be regarded as plagiarism.
- The score of the part that is submitted after the deadline will get some penalties according to the following rule:

$$LateScore = (\frac{86400 - DelayTime(sec.)}{86400})OriginalScore$$

Non-Programming Part

Problem 1. Pusheens with Binary Tree (20%)

Pusheen Kingdom again! You're now still the teacher of Pusheens. Today, they want to learn about **binary tree**. Please answer the following problems. (Note that in Problem 1, if you're not confident with your English ability, please write in Chinese.)

1. (5%) Given **preorder** and **inorder** traversals of a binary tree, please draw the original binary tree, then write the **postorder** traversal of this tree.
   preorder: DEONCJIGKFLMQBHAP
   inorder: NOIJGCKELFMDBQAHP

2. (3%) Given **preorder** and **postorder** traversals of a binary tree, please give one simple (number of nodes<5) example that more than one binary trees can be generated. You should write down the **preorder** traversals, **postorder** traversals, and draw the trees.

3. (6%) Please use the online resource to learn some topics about binary tree that not covered in class, including **successor**, **predecessor**, and **threaded binary tree**. You have to explain them by your words (in this problem, you can not write more than **100** words). You can only write down their definitions. Don't directly copy the resource. Also, you have to give the reference. (Your classmates or friends can't be your reference in this problem.)

4. (6%) Given the **preorder** traversal sequence, please draw a **threaded binary search tree**.
   preorder: 17, 9, 5, 2, 6, 10, 25, 20, 19, 21, 23, 39, 28, 34, 41
   Note that you should draw ordinary tree edges in solid lines, thread tree edges in dashed or dotted lines.

Problem 2. Naughty Rabbits Playing Poker (20%)

Naughty Rabbits again! This time, your rabbits are playing Poker and they need your help. Each rabbit has one Poker card in hand. Poker cards include 13 ranks and 4 suits: clubs(♣), diamonds(♢), hearts(♡), and spades(♠). Each card consists of one kind of suit and one rank. The order of four suits would be: clubs(♣) < diamonds(♢) < hearts(♡) < spades(♠). For the questions below, we would assume that rank 'A' equals 1, 'J' equals 11, 'Q' equals 12, and 'K' equals 13.

1. (4%) Assuming that ♣=1, ♢=2, ♡=3, ♠=4, we define a primary hash function of a given card like this: $h1(card) = suit \times rank(mod\ m)$ and a secondary hash function of a given card like this: $h2(card) = suit \times 7 + rank(mod\ m)$. We learned about double hashing in class, so let's use it to insert the cards of 8 rabbits: {♠7, ♡K, ♠10, ♢Q, ♣6, ♡J, ♢5, ♣3} into a hash table of length m = 12. Please fill the table below to simulate the process of insertions. The row of the table indicates a single insertion, and the column represents the corresponding card of that hash value.(You can remain it to be a blank if there doesn't exist a corresponding card.)

   The result of the first iteration for inserting ♠7 is given in the table as an example. The cards which causes a collision with ♠7 will need to utilize the secondary hash function.

	0	1	2	3	4	5	6	7	8	9	10	11
♠7					♠7							
♡K					♠7							
♠10					♠7							
♢Q					♠7							
♣6					♠7							
♡J					♠7							
♢5					♠7							
♣3					♠7							

2. (3%) Next, let's assume that ♣=17, ♢=22, ♡=53, ♠=42, and then we define a hash function of a given card like this: $h(card) = 30 + suit \times rank(mod\ 23)$. Please list all the cards which share the same hash value with ♡Q. Also, you have to write down the hash value.

3. (3%) In this question, we would regard rank 'A' as 01, '2' as 02, ... 'J' as 11, 'Q' as 12, and 'K' as 13. That is, card ♡A would be regarded as ♡01. Therefore, all cards can be mapped to a representation of length 3, containing 1 symbol and 2 digits. Suppose here is a list of cards of 10 rabbits: {♣J, ♢5, ♡Q, ♢10, ♡9, ♠A, ♣K, ♠3, ♢2, ♡8}. Please mapped them to the representation described above and then apply **LSD Radix Sort**. The result should be in increasing order. Define the base of Radix Sort to be a single symbol/digit of the representation. Write down the sequence of the list of each iteration.

4. (4%) A card array looks like this:

1	2	3	4	5	6	7	8	9	10
♣A	♢9	♡7	♣8	♡Q	♠K	♣4	♠2	♡A	♢7

Please determine how the array would look like after calling Max-Heapify(Card_Array, 1). Write down the result of each pass. (Note that the result after calling Max-Heapify is not necessarily sorted.)

5. (6%) Next, you would like to sort a list of cards of N rabbits. However, you don't have direct access to the cards. Recall the Naughty Rabbits problem in HW2, you know that you can ask questions to sort the rabbits! In this problem, you can ask **any** Yes/No question like "Is Rabbit A's card larger than ♠10?" or "Is MUMU handsome?" Note that same as HW2, some rabbits may lie but there are strictly more than $\frac{N}{2}$ honest rabbits. Write down a procedure that asks $O(NlogN)$ questions to **determine the sorted list** of N cards in increasing order.(In other words, you don't need to actually sort the rabbits. Being able to output the sorted list of N cards will earn you full credit.) Extra space complexity is restricted to $O(1)$. Also, briefly explain the time complexity. (Note that N may be very large since there may be more than one deck of cards.)

Problem 3. Meldable Priority Queue (20%)

A *meldable priority queue* stores a set of keys from a totally ordered universe (e.g. Integers) and supports following operations:

- `makeQueue()`: Return a priority queue over an empty set.
- `maximum(Q)`: Return the biggest element of Q (if any).
- `extractMax(Q)`: Remove and return the biggest element of Q (if any).
- `insert(Q, x)`: Insert element x into Q.
- `meld(Q1, Q2)`: Return a new priority queue over elements of Q_1 and Q_2. Both Q_1 and Q_2 will not be referenced in the future.

In the following problemset, you will design a meldable priority queue data structure. You are also required to briefly explain the correctness and time complexity

1. (6%) Meldable Binary Heap
   Describe an algorithm of `meld(Q1, Q2)` for MaxHeap in amortized $O(\log^2 n)$ time.

   That is, given any sequence of operations of length m with n `insert` operations with $m = O(n)$. Total time for performing these operations is $O(m \log^2 n)$ or $O(n \log^2 n)$ for $m = O(n)$.

2. (9%) Meldable Binary Max Tree
   Here we consider binary max tree for priority queue:

   Definition 0.1. A *max tree* is a tree in which the key value in each node is no smaller (larger) than the key values in its children(if any).

   A *binary max tree* is a binary tree that is also a max tree.

```
1 struct Node {
2     int key; // key of the element
3     int size; // size of the subtree rooted at this node
4     struct Node *lc; // pointer to left child
5     struct Node *rc; // pointer to right child
6 };
```

   Given 2 trees `Q1` and `Q2`, describe a algorithm for `meld(Q1, Q2)` in $O(\log n)$ time where n is the total size of $Q1$ and $Q2$.

```
1 struct Node* meld(struct Node *Q1, struct Node *Q2) {
2     if (Q1 == NULL) return Q2;
3     if (Q2 == NULL) return Q1;
4     // TODO
5 }
```

3. (5%) Meldable Binary Max Tree

Implement rest of the operations with $O(1)$ call to `meld()` and $O(1)$ additional time.

```
1  int maximum(struct Node *Q) {
2      // return the maximum element of Q.
3  }
4
5  struct Node* extractMax(struct Node *Q) {
6      // return the priority queue of Q with max element removed.
7  }
8
9  struct Node* insert(struct Node *Q, int x) {
10     // return the priority queue with element x inserted.
11 }
```

Programming Part

Problem 4. (20%)

The deadline of homework 3 of DSA(Digital Security Assurance) is around the corner, and more and more students are attending the TA hour. Cerkiv the TA is quite impatient, and think that answering questions for student one by one is very inefficient.

Thus, he encourages students that comes to his TA hours to gather into group if they have similar questions, so that he can answer several students at once. To encourage, every time Cerkiv becomes available (just finished answering a group of student), he would pick the currently largest group to answer. If there are more than one group with the largest size, he would answer the group that has the **smallest minimum student ID**. For example, if there are currently two groups {1,5,6} and {2,3,4}, Cerkiv will answer the group {1,5,6} first, since its minimum student id 1 is smaller.

During Cerkiv's TA hour tonight, all the events are recorded by timestamp order, where every event is of either of the two types:

- Two groups are joined together.
- Cerkiv becomes available and picks a new group of students.

Initially each student is an individual group. Your are given the list of attending students and the list of events. For each event of *type-2*, you're required to find out a member in the group picked by Cerkiv.

Input Format

The first line is a single integer N, indicating the number of students.

The following N lines are N students' name. The i^{th} line is the name of the student with student ID i.

Then there is a line containing an integer M, indicating the number of events.

The following M lines represent M events in timestamp order. Each line starts with an integer that indicates the type of the event. After that

- If the event is of *type-2*, the line is over.
- Otherwise, there will be two student IDs x and y, indicating that the group of student x and the group of student y are joined together.

The followings are guaranteed for all inputs:

- $1 \leq N, M \leq 2 \times 10^5$
- $1 \leq x, y \leq N$
- The length of every student's name is no more than 10.
- Every *type-1* event joins two **different** groups where all members have not been answered yet.
- A *type-2* event takes place only when there are at least one group left.

Output Format

For each *type-2* event, output the name of the member with smallest student ID in the picked group.

Subtasks

10%	there is no *type-1* event.
15%	$N, M \leq 2000$.
25%	there is no *type-2* event before the last *type-1* event takes place.
50%	no additional constraints.

Sample Input

```
5
pusheen
justin
plagia
rhythm
caesar
4
2
1 3 4
1 2 5
2
```

Sample Output

```
pusheen
justin
```

Problem 5. (20%)

Time travel has always been a dream to researchers. One hypothesis is that time traveling is merely a switching between different world lines.

In one accidental occasion, mad scientist Kyōma Hōōin (a.k.a Okabe Rintarō) realized that he has gained ability to jump between different world lines. Further investigation leaded Kyōma to believe that each world line can be represented as a unique integer. Moreover, one can simply switch between world lines by triggering specific events(which can also be represented as integers).

However, his power is not without limits. Events that can be successfully triggered are scarce, and each event can trigger exactly one certain world line(the rule is given below). Given those constraints, could Kyōma fully utilize his power and fulfill his prolonged wish? Or, would he have no choice but to be left in despair? The future of Kyōma and the world lies in your hand now.

Though possible, Kyōma is nice enough not to challenge you with a world or event which has representation smaller than 0.

Formula

The next world line = Current world line $\oplus$ the event Kyōma choose. ($\oplus$ means the steins;gates choice, more commonly referred to as XOR.)

Input Description

The first line is an integer N, indicating the number of initial events. Following N lines are the binary numbers representing these events. The next line contains an integer M, indicating the number of instructions. The last M lines contain an operation code followed by a number representing some event/worldline.

Details about operation codes as below:

- **Q**uery: Given the number of current world line, please help Kyōma decide which event should he choose from his list of choices to achieve the numerically largest world line next time.
- **U**pdate: A brand new event discovered by Kyōma, which may possibly be useful in the future. Please add this event to Kyōma's list of choices.
- **D**elete: Some events are traps which might lead to miserable futures (like WWIII or even worse things) . Kyōma is a good fellow at heart, and would like to avoid such events even if it meant to compromise his wish to reality. Thus, please delete this event from Kyōma's list of choices.
- hint: trie

Input Constraint and Details

- $1 \leq N, M \leq 10^5$
- $0 \leq worldline\&event \leq 2^{64}$
- The worldline & event given will be in binary format.
- If the event to Update is already in list of choices, please ignore this request.
- If the event Kyōma wishes to delete is not in list of choices, please ignore this request.

Output Format

For each query operation, please output one line containing the chosen event to trigger for each Query. **Note that output should be in binary expression with no leading zeros.**

Score Distribution

- for 20% testdata:
  - $1 \leq N, M \leq 10^3$
  - 5%: operation ∈ "Query"
  - 5%: operation ∈ "Query","Update"
  - 10%: operation ∈ "Query","Update","Delete"
- for 80% testdata:
  - 15%: operation ∈ "Query"
  - 5%: operation ∈ "Update","Delete"
  - 20%: operation ∈ "Query","Update"
  - 40%: operation ∈ "Query","Update","Delete"

Sample Input

```
5
1001000
1000011
101111
1100
11001
5
Q 101000
U 1010111
Q 100111
D 101111
Q 100111
```

Sample Output

```
1000011
1010111
1010111
```