

CS3334 - Data Structures

Lab 3

Outline

- Queue and Hash Exercises 5
- Assignment 746: Fast Food Restaurant Queue
- Assignment 823: Word Review Hash table

Verdict Information

- ❑ **Presentation Error (PE)**: Your program outputs are correct but are not presented in the correct way.
Check for spaces line breaks, ‘’, ‘\n’, ‘comma’...
e.g., 819 Josephus Problem
- ❑ **Wrong Answer (WA)**: output is not correct.
\$input format,
\$special case, any tricky test case?
- ❑ **Runtime Error (RE)**: Your program failed during the execution.
\$ index overflow, $a[-1]$, $a[n+1]$.
\$ Stack/memory overflow, endless recursions, self call
- ❑ **Time Limit Exceeded (TLE)**: Your program is not finished in required time.
\$ time complexity, infinite loop, waiting for input
- ❑ **Memory Limit Exceeded (MLE)**: tried to use more memory than allowed.
\$ static memory (array size) + memory for stack/queue during execution
- ❑ **Output Limit Exceeded (OLE)**: Your program output too much than expected.
\$ infinite loop

Exercise 1

Determine whether each of the following characteristics apply to **a stack, a queue, both, or none**.

- a. An element is inserted at a special place called the top. *stack*
- b. An element is inserted at a special place called the rear. *queue*
- c. The structure can hold only one type of data element. *both*
- d. An element is deleted at the front. *queue*
- e. The i -th position may be deleted. *none*
- f. An element is deleted at the top. *stack*
- g. The structure is a LIFO structure. *stack*
- h. The structure is a FIFO structure. *queue*

Exercise 2

Given an integer k and a queue of integers, the task is to reverse the order of the first k elements of the queue, leaving the other elements in the same relative order.

Only following standard operations are allowed on queue.

- API
- enqueue(x) : Add an item x to rear of queue
 - dequeue() : Remove an item from front of queue
 - size() : Returns number of elements in queue.
 - front() : Finds front item but not remove it.

Example:


Input: $k = 5$, queue = {10, 20, 30, 40, 50, 60, 70, 80, 90, 100} (10 is the front)


Output: {50, 40, 30, 20, 10, 60, 70, 80, 90, 100}


void **reverseFirstK**(int k ⁵, queue<int>& Queue) { . . . }

1. Create an empty stack.
2. One by one dequeue first K items from queue and push the dequeued items to stack.
3. Enqueue the contents of stack at the back of the queue
4. Dequeue (size-k) elements from the front and enqueue them one by one to the queue.

```
void reverseFirstK(int k, queue<int>& Queue)
{
    if (Queue.empty() == true || k > Queue.size())
        return;
    if (k <= 0)
        return;

    stack<int> Stack;           // step 1  $O(1)$ 
    for (int i = 0; i < k; i++) { // step 2
        Stack.push(Queue.front()); //  $O(k)$ 
        Queue.pop();
    }
    // 

    while (!Stack.empty()) { // step 3  $O(k)$ 
        Queue.push(Stack.top());
        Stack.pop();
    }
    // 

    for (int i = 0; i < Queue.size() - k; i++) { // step 4
        Queue.push(Queue.front()); //  $O(N-k)$ 
        Queue.pop();
    }
    // 

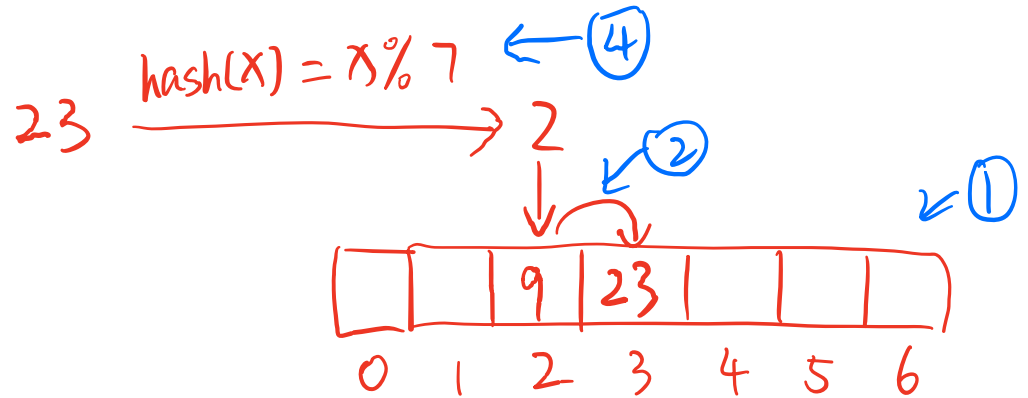
    Complexity?  $O(N)$ 
}
```

Exercise 3

According to lecture, a hash table consists of several components.

Which of these was **NOT** one of those components?

1. An array ✓
2. Collision handling ✓
3. Encryption ✗
4. A hash function ✓



Exercise 4

Given the following input

(4322, 1334, 1471, 9679, 1989, 6171, 6173, 4199)
hash(k) = 2 4 1 9 9 1 3 9
and the hash function $x \bmod 10$,

which of the following statement(s) are true?

1. 9679, 1989, 4199 hash to the same value ✓
2. 1471, 6171 hash to the same value ✓
3. All elements hash to the same value ✗
4. Each element hashes to a different value ✗

Exercise 5

~ $\boxed{1/3}$ *

when the hash table is full,

⇒ Inefficient!

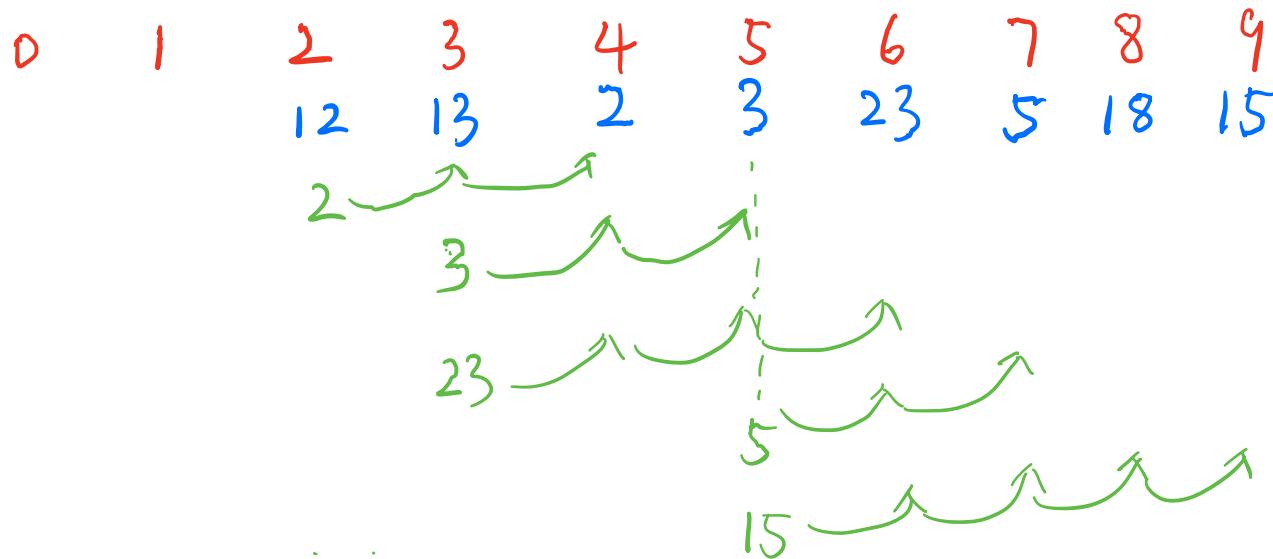
hash(k) = 2 8 3

(2) (3) (3) (5)

(5)

The keys 12, 18, 13, 2, 3, 23, 5 and 15 are inserted into an initially empty hash table of length 10 using open addressing with hash function $h(k) = k \bmod 10$ and linear probing.

What is the resultant hash table?



0	
1	
2	12
3	13
4	2
5	3
6	23
7	5
8	18
9	15

746: Fast Food Restaurant

Description

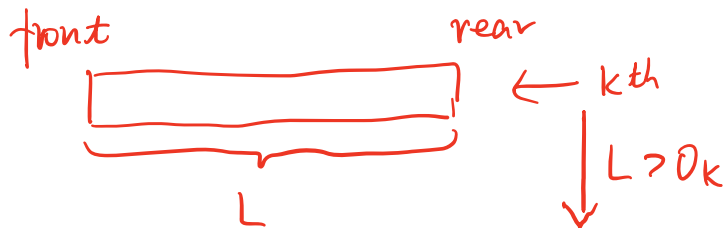
What a nice holiday! After travelling to a small beautiful town, Tom plans to get something to eat, finally he finds a small fast food restaurant with a long queue for ordering. He wonders **how long he has to wait before ordering his food**.

Giving you N customers, for **each** customer you will be given

the arrival time A ,

the service time O needed for making order

and the maximum number L of the people representing that if there are more than L people ahead of him (when he/she arrives) in the queue, the customer will give up.



746: Fast Food Restaurant

Input

The input contains multiple test cases.

For each test case,

the first line contains a number N indicating the number of customers.

Each of the following N lines contains three numbers, A_i , O_i and L_i corresponding to the i-th customer.

It is guaranteed that the sequence of arrival times is non-decreasing (if two customers have the same arrival time, the customer who is earlier in the input is considered to have arrived earlier).

$1 \leq N \leq 100000$, $0 \leq A, L \leq 100000$, $0 \leq O \leq 100$.

Output

For each test case,

print the result of the last customer, that is print -1 if he/she gives up, otherwise print the time when he/she begins to be served.

746: Fast Food Restaurant

Sample Input	Sample Output
<div> <div>2</div> <div>N</div> <div>1 1 1 A₁, D₁, L₁ arrive, service, limit</div> <div>1 0 0 A₂, D₂, L₂</div> <div>3</div> <div>0 1 1</div> <div>0 1 2</div> <div>0 1 1</div> <div>3</div> <div>0 1 1</div> <div>0 0 1</div> <div>0 1 2</div> <div>4</div> <div>0 9 0</div> <div>7 4 1</div> <div>8 3 1</div> <div>12 2 2</div> </div>	<div> <div>-1</div> <div>-1</div> <div>1</div> <div>13</div> </div> <div> <p>queue q; // store the finish time of each customer</p> <p>foreach arrive, service, limit</p> <p>while (q.front() <= arrive) q.pop() // these are finished</p> <p>if (q.size() <= limit) finish = max(q.rear(), arrive) + service // this customer join the queue</p> <p>check, when the last customer arrive</p> </div> <div> <p>Which data structure seems a good choice? Why?</p> <p>Queue, FIFO</p> <p>1. [] ← (0, 9, 0)</p> <p>2. [9] ← (7, 4, 1)</p> <p>3. [9, 13] ← (8, 3, 1)</p> <p>4. [9, 13] ← (12, 2, 2)</p> <p>[13] ← (12, 2, 2)</p> <p>finish time of the 3rd customer</p> <p>give up</p> </div>

823: Word Review

Description

Kyaru has just finished her argumentative essay, the most significant part of the course work of GE1401 University English.

In the class, teacher gave her a list of academic words.

Before submitting, she would like to check quality of her essay by find a consecutive part in her essay where most academic words are used. ☆

Also, the number of words in the part she finds needs to be as small as possible. ☆

823: Word Review

Input

The input **contains several cases**. (here we don't know the detailed#)

For each case, there holds:

The first line of the input contains a positive integer n , indicating the number of words in the academic word list.

In the following n lines, each line contains a string with a length not exceeding 10, representing an academic word.

The next line contains a positive integer m , indicating the word count of Kyaru's essay.

In the following m lines, each line contains a string with a length not exceeding 10, together denoting the content of the essay.

The input of test cases terminates by end of file.

$1 \leq n \leq 1000$, $1 \leq m \leq 100000$

It is guaranteed that all the words consist of only English alphabet in lowercase.

823: Word Review

Output

For each test case, output two lines.

The first line contains an integer indicating the number of academic words appearing in the part found (the same word appearing multiple times counts only once),

followed by another line containing an integer indicating the length of this part.

?

$O(nm)$ algorithm or any other better algorithm can pass through all test cases.

823: Word Review

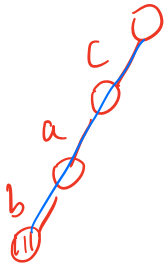
Sample Input	Sample Output
<p> <u>3</u> <u>n</u> analyze <u>initiates</u> <u>split</u> </p> <p> <u>14</u> <u>m</u> (check zero) when a <u>initiates</u> nuclear reaction <u>initiates</u> the old <u>uranium</u> <u>nucleus</u> will <u>split</u> into two <u>split</u> nuclei </p> <p> "this part" </p>	<p> Idea: Hash table var F [(str, position)] </p> <p> Solution: var D (std::unordered_map) for 1 ---- n var str D.insert(str) </p> <p> Add these n key words in the hash table </p> <p> for i in 1 ---- m var str if str in D // is in the key words F.push_back((str, i)) if str appear first time ++res1 </p> <p> same words count once </p> <p> F[(initiates, 3), (initiates, 6), ..., ..., (split, 14), (split, 17)] Then find the length of "this part" </p>

Solution 1: Hash table

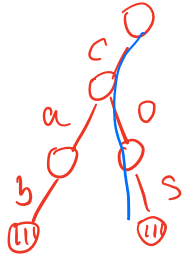
Solution 2: Trie (a prefix digital tree), faster than hash table, more memory

E.g.

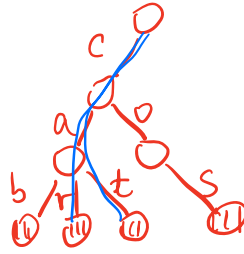
{cab, cos, car, cat, cate, rain}



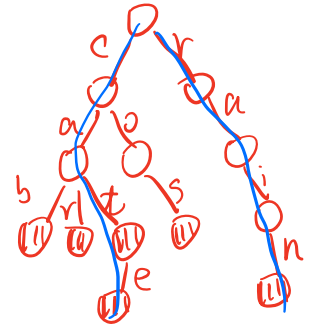
{cab}



{cab, cos}



{cab, cos, car, cat}



{... , cate, rain}