M24-CS1.304 Data Structures and Algorithms for Problem Solving

Assignment 2

Deadline: 11.59PM, 13 October 2024

Important Points

1. The assignment has 3 problems. The first two problems are mandatory and weigh 50 marks each. The last problem is ungraded but students are encouraged to attempt it.
2. Only C++ is allowed. Allowed versions: 11, 14, 17, and 20.
3. **Only the headers, <iostream>, <string>, and <vector> are allowed.** Use of any other headers in the submissions will lead to 0 marks being awarded for the particular problem.
4. Directory structure to be followed for submission:
   2023201045_A2
   – 2023201045_A2_Q1a.cpp
   – 2023201045_A2_Q1b.cpp
   – 2023201045_A2_Q2a.cpp
   – 2023201045_A2_Q2b.cpp
   – 2023201045_A2_Q2c.cpp
   Replace 2023201045 with your roll number.
   Submissions not following the above format will not be evaluated.
5. Submission Format: Follow the above mentioned directory structure and zip the *RollNo_A2* folder and submit *RollNo_A2.zip* on the courses portal.
6. You can ask queries over the courses portal.
7. **Late Submission Rule**: Deadlines for the assignment are final and will not be extended. Late submissions cost 5% loss of marks for each late day up to 6 days. (i.e, 5%, 10%, …). Submissions beyond 6 days from the deadline shall receive 0 marks.
8. You are encouraged to discuss approaches among yourselves but any instance of code plagiarism will not be tolerated.

NOTE: In case of plagiarism in any of the questions, the students involved will be awarded -10 as the final marks for the given assignment.

Sample Tests: DSAPS-M24-A2-Sample-Tests

Q1 Photographic Memory [50 Marks]
Storage and retrieval of information is a common task. Data may not always be ordered, but efficient retrieval is oftentimes of significant importance.

1A Offline Processing [10 Marks]
For this problem, you shall be given n integer key-value pairs $< k,\ v >$ to store and q queries to retrieve values using their keys.
Note - If a key has been stored prior, it gets overwritten by successive $< k,\ v >$ pairs.

Input
The first line of input shall contain a single integer, t, denoting the number of test cases.
In each test, the first line shall contain a single integer n, the number of key-value pairs.
Each of the next n lines shall contain two integers each, k, v, the key and value respectively.
The next line shall contain a single integer q, denoting the number of queries.
Each of the next q lines contain a single integer each, k_q, the query key.

Constraints

$1 \le t \le 10^4$

$1 \le \sum n, \sum q \le 2 \cdot 10^5$

$1 \le k,\ k_q,\ v \le 10^{18}$

Memory Limit
512MB

Time Limit
2s

Output
For each test, print q lines, responses to each of the queries. If k_q is stored already in that test, print the value v associated, otherwise print 0.

1B Online Processing [10 Marks]

For this problem, you shall be given n queries. Process them as operations on an associative array and respond accordingly. Queries can be of two types, store ($type$: 0), and retrieve ($type$: 1).

For ($type$: 0) queries, you shall be given integer key-value pairs $< k,\ v >$. Store them and respond 1 if the key is already stored, 0 otherwise. Overwrite if present.

For ($type$: 1) queries, you shall be given an integer k_q, respond with the value associated if this key is already stored, respond with 0 otherwise. For reference, you can incorporate the ideas of hashing - open-addressing and separate-chaining.

Input

The first line of input shall contain a single integer, t, denoting the number of tests.

The first line of each test shall contain n, the number of queries.

Each of the next n lines will contain a single query.

$type$: 0 queries follow the format: 0 k v

$type$: 1 queries follow the format: 1 k

Constraints

$1 \leq t \leq 10^4$

$1 \leq \sum n \leq 2 \cdot 10^5$

$1 \leq k,\ v \leq 10^{18}$

Memory Limit

512MB

Time Limit

2s

Output

For each test, print n lines, responses to each of the queries.

1C Retrieval Premium [30 Marks]

For this problem, you shall be given n queries. Process them and respond accordingly. Queries can be of three types, store ($type$: 0), retrieve ($type$: 1), and delete ($type$: 2). Consider an initially empty storage S.

For ($type$: 0) queries, report the presence of the string s_i in S, and store the string s_i in S.

For ($type$: 1) queries, you only need to report the presence of the string in your store.

For ($type$: 2) queries, you need to report the presence of the string s_i in S, and remove the string if it is present in S. For reference, you can incorporate the ideas of cuckoo-hashing in this problem.

Input

The first line of input contains a single integer t, denoting the number of test cases.

The first line of each test contains a single integer, n, the number of queries.

Each of the next n lines contains an integer $type$, (0, 1, or 2) the type of the query, and s_i the query string.

Constraints

$1 \leq t \leq 10^4$

$1 \leq \sum n \leq 2.1 \cdot 10^6$

$1 \leq \sum_i s_i \leq 2 \cdot 10^8$ $(summed\ across\ all\ tests)$

$1 \leq |s_i| \leq 10^5$

s_i is composed of lower-case English alphabet characters.

It is guaranteed that the number of ($type$: 1) queries are at least 20 times as many as queries of the other two types.

Memory Limit

1GB

Time Limit

2s

Output

For each query, output a single integer denoting the presence of the string in the storage S for that test. Print 1 if s_i was present in S, print 0 otherwise.

Q2 Strings Galore [50 Marks]
String processing is a common problem across domains. With greater amounts of data being generated with time, the efficiency of algorithms for storage/retrieval has become increasingly important.

2A Never Again MLE [30 Marks]
Comparing two strings has a complexity of $O(min(n_1, n_2))$, where n_1, and n_2 are the sizes of the two strings. In practice, when processing large amounts of data, a variety of techniques are applied to optimize for memory and time.

For this problem, you are tasked with designing a system to efficiently check for the presence of a set of strings while minimizing memory usage. For reference, you can incorporate the ideas of string hashing and bloom filters in this problem.

You will be given n strings. For each string s_i, report whether or not s_i has been encountered before.

Input
The first line of input contains a single integer, t, denoting the number of tests.
In each test, the first line will contain a single integer n, the number of strings.
Each of the next n lines will contain a single string, s_i.

Constraints
$1 \leq t \leq 10^4$
$1 \leq n \leq 10^5$
$1 \leq \sum_i s_i \leq 2 \cdot 10^8$ $(summed\ across\ all\ tests)$
$1 \leq |s_i| \leq 10^5$
s_i is composed of lower-case English alphabet characters.

Memory Limit
64MB [30 Marks]
512MB [5 Marks]
Suboptimal submissions failing under 64MB but passing under 512MB, may be awarded up to 5 marks.

Time Limit
6s

Output
For each string, report 1 if it was encountered before in that test, 0 otherwise.

2B Pooling Resources [20 Marks]

For this problem, you shall efficiently store strings in an associative array. You will be given n queries. Queries can be of two types, put ($type$: 0), and, get ($type$: 1). Consider an initially empty associative array, A.

For ($type$: 0) queries, you will be given an integer i, and a string s_i. This operation maps s_i to i in A.

For ($type$: 1) queries, you will be given an integer q_i, you must respond with its corresponding value in A if the key, q_i is present in A, and with 0 otherwise.

For reference, you can incorporate the ideas of string-pooling in this problem.

Input

The first line of input contains a single integer, t, denoting the number of tests.

The first line of each test contains a single integer, n, the number of queries.

Each of the next n lines will contain a single query.

($type$: 0) queries follow the format: 0 i s_i

($type$: 1) queries follow the format: 1 i

Constraints

$1 \leq t \leq 10^4$

$1 \leq \sum n \leq 10^6$

$1 \leq i \leq 10^{18}$

$1 \leq \sum_i s_i \leq 2 \cdot 10^8$ $(summed\ across\ all\ tests)$

$1 \leq |s_i| \leq 10^5$

s_i is composed of lower-case English alphabet characters.

Let S denote the set of **unique** strings in the input.

$1 \leq \sum_{s_i \in S} |s_i| \leq 10^6$ $(summed\ across\ all\ tests)$

Memory Limit

64MB [20 Marks]

512MB [5 Marks]

Suboptimal submissions failing under 64MB but passing under 512MB, may be awarded up to 5 marks.

Time Limit

5s

Output

For each query of $type$: 1, report the response - the associated string s_{q_i} or 0 if there is no such key.

3 Red Team [0 Marks]

Hashing is known to be the magical $O(1)$ solution to computation (to be fair, it is **not** - simply hashing a string str, can take $O(len(str))$ time). It so happens that this fairy tale is not always true and bad actors can exploit that. A few smart hashmap implementations avoid this - for instance, Java has an interesting modification, SipHash is another popular implementation. Similarly, there exist a large number of pathological cases where programs may fail.

In this problem, you will be given buggy $C++$ programs and corresponding problem descriptions. You are expected to generate adversarial inputs to force the programs to err ($RTE/WA/TLE/MLE$).

C++ programs/problems to be added on a rolling basis here.

For this problem, you can reference PRNGs. A decent implementation in C++ is found here.

NOTE: In case of plagiarism in any of the questions, the students involved will be awarded -10 as the final marks for the given assignment.