L27 : Tries

1-Tut : Maximum XOR Subarray

Send Feedback

Given an array of N integers, find the subarray whose XOR is maximum.

Input Format:

First line of input contains an integer N, representing number of elements in array.
Next line contains N space-separated integers.

Constraints:

1 <= N <= 10^6
1 <= A[i] <=10^5

Output Format:

Print XOR of the subarray whose XOR of all elements in subarray is maximum over all subarrays.

Sample Input 1:

4
1 2 3 4

Sample Output 1:

7

```
1.  #include<bits/stdc++.h>
2.  using namespace std;
3.
4.  // Assumed int size
5.  #define INT_SIZE 32
6.
7.  // A Trie Node
8.  struct TrieNode
9.  {
10.     int value;  // Only used in leaf nodes
11.     TrieNode *arr[2];
12. };
13.
14. // Utility function tp create a Trie node
15. TrieNode *newNode()
16. {
17.     TrieNode *temp = new TrieNode;
18.     temp->value = 0;
19.     temp->arr[0] = temp->arr[1] = NULL;
20.     return temp;
21. }
22.
23. // Inserts pre_xor to trie with given root
```

```
void insert(TrieNode *root, int pre_xor)
{
    TrieNode *temp = root;

    // Start from the msb, insert all bits of
    // pre_xor into Trie
    for (int i=INT_SIZE-1; i>=0; i--)
    {
        // Find current bit in given prefix
        bool val = pre_xor & (1<<i);

        // Create a new node if needed
        if (temp->arr[val] == NULL)
            temp->arr[val] = newNode();

        temp = temp->arr[val];
    }

    // Store value at leaf node
    temp->value = pre_xor;
}

// Finds the maximum XOR ending with last number in
// prefix XOR 'pre_xor' and returns the XOR of this maximum
// with pre_xor which is maximum XOR ending with last element
// of pre_xor.
int query(TrieNode *root, int pre_xor)
{
    TrieNode *temp = root;
    for (int i=INT_SIZE-1; i>=0; i--)
    {
        // Find current bit in given prefix
        bool val = pre_xor & (1<<i);

        // Traverse Trie, first look for a
        // prefix that has opposite bit
        if (temp->arr[1-val]!=NULL)
            temp = temp->arr[1-val];

        // If there is no prefix with opposite
        // bit, then look for same bit.
        else if (temp->arr[val] != NULL)
            temp = temp->arr[val];
    }
```

```
68.     return pre_xor^(temp->value);
69. }
70.
71. // Returns maximum XOR value of a subarray in arr[0..n-1]
72. int maxSubarrayXOR(int *arr, int n)
73. {
74.     // Create a Trie and insert 0 into it
75.     TrieNode *root = newNode();
76.     insert(root, 0);
77.
78.     // Initialize answer and xor of current prefix
79.     int result = INT_MIN, pre_xor =0;
80.
81.     // Traverse all input array element
82.     for (int i=0; i<n; i++)
83.     {
84.         // update current prefix xor and insert it into Trie
85.         pre_xor = pre_xor^arr[i];
86.         insert(root, pre_xor);
87.
88.         // Query for current prefix xor in Trie and update
89.         // result if required
90.         result = max(result, query(root, pre_xor));
91.     }
92.     return result;
93. }
94.
95. int main( int argc , char ** argv )
96. {
97.         ios_base::sync_with_stdio(false) ;
98.         cin.tie(NULL) ;
99.
100.        int n;
101.        cin>>n;
102.        int* arr = new int[n];
103.        for (int i = 0; i < n; ++i)
104.        {
105.                cin>>arr[i];
106.        }
107.
108.        cout << maxSubarrayXOR(arr, n) << '\n';
109.
110.        return 0 ;
111.    }
```

2-Tut : Sub - XOR

Send Feedback

Given an array of positive integers, you have to print the number of subarrays whose XOR is less than K.

Subarrays are defined as a sequence of continuous elements A_i, A_{i+1}, ..., A_j. XOR of a subarray is defined as A_i ^ A_{i+1} ^ ... ^ A_j. Symbol ^ is Exclusive Or.

Input Format:

First line of input contains N and K, space separated.
Second line of input contains N space separated.integers.

Constraints:

1 ≤ N ≤ 10^5
1 ≤ A[i] ≤ 10^5
1 ≤ K ≤ 10^6

Output Format:

For each test case, print the required answer.

Sample Input 1:

5 2
4 1 3 2 7

Sample Output 1:

3

Explanation to Sample Input 1

Only subarrays satisfying the conditions are [1],[1,3,2] and [3,2].

```
#include <iostream>
using namespace std;
class trienode
{
public:
    int left_count, right_count;
    trienode *left;
    trienode *right;
    trienode()
    {
        left_count = 0;
        right_count = 0;
        left = NULL;  //left denotes 0
        right = NULL; //right denotes 1
    }
};
void insert(trienode *root, int element)
{
    for (int i = 31; i >= 0; i--)
    {
        int x = (element >> i) & 1;
```

```
22.
23.       if (x) //if the current bit is 1
24.       {
25.          root->right_count++;
26.          if (root->right == NULL)
27.             root->right = new trienode();
28.          root = root->right;
29.       }
30.       else
31.       {
32.          root->left_count++;
33.          if (root->left == NULL)
34.             root->left = new trienode();
35.          root = root->left;
36.       }
37.    }
38. }
39. int query(trienode *root, int element, int k)
40. {
41.    if (root == NULL)
42.       return 0;
43.    int res = 0;
44.    for (int i = 31; i >= 0; i--)
45.    {
46.       bool current_bit_of_k = (k >> i) & 1;
47.       bool current_bit_of_element = (element >> i) & 1;
48.       if (current_bit_of_k)//if the current bit of k is 1
49.       {
50.          if (current_bit_of_element)//if current bit of element is 1
51.          {
52.             res += root->right_count;
53.             if (root->left == NULL)
54.                return res;
55.             root = root->left;
56.          }
57.
58.          else//if current bit of element is 0
59.          {
60.             res += root->left_count;
61.             if (root->right == NULL)
62.                return res;
63.             root = root->right;
64.          }
65.       }
```

```
66.         else//if the current bit of k is zero
67.         {
68.            if (current_bit_of_element)//if current bit of element is 1
69.            {
70.               if (root->right == NULL)
71.                  return res;
72.               root = root->right;
73.            }
74.            else//if current bit of element is 0
75.            {
76.               if (root->left == NULL)
77.                  return res;
78.               root = root->left;
79.            }
80.         }
81.      }
82.      return res;
83. }
84. int main()
85. {
86.
87.         int n, k;
88.         cin >> n >> k;
89.         int temp, temp1, temp2 = 0; // will be using these three for storing current
    XOR
90.         // we will not be storing the inputs in the array
91.         trienode *root = new trienode();
92.         insert(root, 0);
93.         long long total = 0;
94.         for (int i = 0; i < n; i++)
95.         {
96.            cin >> temp;
97.            temp1 = temp2 ^ temp;
98.            total += query(root, temp1, k);
99.            insert(root, temp1);
100.               temp2 = temp1;
101.            }
102.            cout << total << endl;
103.        return 0;
104.     }
```

3-Tut : Search Engine

Send Feedback

Let us see how search engines work. Consider the following simple auto complete feature. When you type some characters in the text bar, the engine automatically gives best matching options among it's database. Your job is simple. Given an incomplete search text, output the best search result.

Each entry in engine's database has a priority factor attached to it. We consider a result / search suggestion best if it has maximum weight and completes the given incomplete search query. For each query in the input, print the maximum weight of the string in the database, that completes the given incomplete search string. In case no such string exists, print -1.

Note: All the strings in database will be unique.

Input Format:

First line contains two integers n and q, which represent number of database entries and number of search queries need to be completed.

Next n lines contain a string s and an integer weight, which are the database entry and it's corresponding priority.

Next q lines follow, each line having a string t, which needs to be completed.

Constraints:

1 ≤ n, weight, len(s), len(t) ≤ 10^6

1 ≤ q ≤ 10^5

Total length of all strings in database entries ≤ 10^6

Total length of all query strings ≤ 10^6

Output Format:

Output q lines, each line containing the maximum possible weight of the match for given query, else -1, in case no valid result is obtained.

Sample Input 1:

```
2 1
potential 10
potent 8
pot
```

Sample Output 1:

```
10
```

```
1.  typedef struct Node
2.  {
3.          Node *next[26];
4.          int maxSubtree;
5.          Node()
6.          {
7.                  maxSubtree = 0;
8.                  for (int i = 0; i < 26; i++)
9.                          next[i] = NULL;
10.         }~Node()
11.         {
```

```
12.                     for (int i = 0; i < 26; i++)
13.                     {
14.                             if (next[i] != NULL)
15.                             {
16.                                     delete next[i];
17.                             }
18.                     }
19.             }
20. }
21.
22. Node;
23. void insert(Node *curr, pair<string, int> &databaseEntry, int index)
24. {
25.             // Setting the Priority of the current text.
26.             if (index == databaseEntry.first.length())
27.             {
28.                     curr->maxSubtree = databaseEntry.second;
29.                     return;
30.             }
31.
32.             curr->maxSubtree = max(curr->maxSubtree, databaseEntry.second);
33.             // Getting the next character.
34.             int nextIndex = (int)(databaseEntry.first[index] - 'a');
35.             // Creating a new node in Trie.
36.             if (curr->next[nextIndex] == NULL)
37.             {
38.                     curr->next[nextIndex] = new Node();
39.             }
40.
41.             insert(curr->next[nextIndex], databaseEntry, index + 1);
42. }
43.
44. int query(Node *curr, string &txt, int index)
45. {
46.             // String not found.
47.             if (curr == NULL)
48.                     return -1;
49.             // String found.
50.             if (index == txt.length())
51.             {
52.                     return curr->maxSubtree;
53.             }
54.
55.             // The next character that we need.
```

```
56.         int nextIndex = (int)(txt[index] - 'a');
57.         return query(curr->next[nextIndex], txt, index + 1);
58. }
59.
60. vector<int> searchEngine(vector<pair<string, int> > database, vector< string >
    text)
61. {
62.         // Creating a Trie data-structure.
63.         Node *root = new Node();
64.         int n = database.size();
65.
66.         for (int i = 0; i < n; i++)
67.         {
68.                 insert(root, database[i], 0);
69.         }
70.
71.         vector<int> ans;
72.         // Iterating in each query.
73.         for (auto &i: text)
74.         {
75.                 // Appending the answer in 'ans'.
76.                 ans.push_back(query(root, i, 0));
77.         }
78.
79.         delete root;
80.         return ans;
81. }
82.
```

4-Tut : Help Me Pradyumana!

Send Feedback

Pradyumn is tired of using auto - correct feature in his smartphone. He needs to correct his auto - correct more times than the auto - correct corrects him. Pradyumn is thinking to make his own app for mobile which will restrict auto - correct feature, instead it will provide auto - completion. Whenever Pradyumn types factorial, auto - correct changes it to fact. Pradyumn's App will show options such as fact, factorial, factory. Now, he can chose from any of these options. As Pradyumn is busy with his front - end work of his App. He asks you to help him. He said "You will be given set of words(words may repeat too but counted as one only). Now, as user starts the app, he will make queries(in lowercase letters only). So, you have to print all the words of dictionary which have the prefix same as given by user as input. And if no such words are available, add this word to your dictionary." As you know, Pradyumn want this app to be as smart as him :P So, implement a program for him such that he can release it on Fun Store.

Input Format:

Single integer N which denotes the size of words which are input as dictionary
N lines of input, where each line is a string of consisting lowercase letter
Single integer Q which denotes the number of queries.
Q number of lines describing the query string on each line given by user

Constraints:

1 ≤ N ≤ 30000
sum(len(string[i])) ≤ 2∗10^5
1 ≤ Q ≤ 10^3

Output Format:

If auto - completions exists, output all of them in lexicographical order else output "No suggestions" without quotes

Sample Input 1:

```
3
fact
factorial
factory
2
fact
pradyumn
```

Sample Output 1:

```
fact
factorial
factory

No suggestions
```

```
#include <bits/stdc++.h>
using namespace std;

class Trie{
    public:
```

```
6.          Trie** children;
7.          bool word_end; //to store wether node has any word ending there ex: if words fact,facto present we need to print both fact and facto (if we dont use word_end we only print "facto")
8.      Trie(){
9.          children = new Trie*[26];
10.         for(int i=0;i<26;i++){
11.             children[i] = NULL;
12.         }
13.         word_end = false;
14.     }
15. };
16.
17. void insert(Trie* root,string str){
18.     if(str.empty()){
19.         root->word_end = true;   //indicates node has the character which is the last character in word
20.         return;
21.     }
22.     Trie* child;
23.     int index = str[0] - 'a';
24.         if(root->children[index]){
25.         child = root->children[index];
26.     }else{
27.         child = new Trie();
28.         root->children[index] = child;
29.     }
30.     insert(child,str.substr(1));
31.     return;
32. }
33.
34. //Printing all the strings thar are possible from given prefix using DFS ex: if given string is "do" start DFS after "do" to get all words in trie
35. void printDFS(Trie* current,string prefix){
36.     if(current->word_end){  //if word end is reached print word ans start exploring again
37.         cout << prefix << endl;
38.     }
39.
40.     for(int i=0;i<26;i++){  //iterating to all possible nodes
41.         Trie* temp = current;
42.         if(temp->children[i]){
43.             char suffix = (int)i + (int)'a'; //concatenating character to original string
44.             printDFS(temp->children[i],prefix+suffix);
```

```
45.         }
46.     }
47. }
48.
49. void query(Trie* root,string prefix){
50.     Trie* current = root;
51.     for(int i=0;i<prefix.length();i++){
52.         int index = prefix[i] - 'a';
53.         if(current->children[index]){
54.             current = current->children[index];
55.         }else{
56.             cout << "No suggestions" << endl;
57.             insert(root,prefix);  //given in question if "No suggestions" insert word in
    words dictionary
58.             return;
59.         }
60.     }
61.
62.     printDFS(current,prefix);//printing words which all have prefix same as given
    string
63.     return;
64. }
65.
66.
67. int main()
68. {
69.     Trie* root = new Trie();
70.     int n;
71.     cin >> n;
72.     while(n--){
73.         string str;
74.         cin >> str;
75.         insert(root,str);
76.     }
77.     int q;
78.     cin >> q;
79.     while(q--){
80.         string str;
81.         cin >> str;
82.         query(root,str);
83.         cout << endl;
84.     }
85.     return 0;
86. }
```

5-Ass : Code Breaker

Send Feedback

Jack Ryan is one of the world's most famous cryptographers. He has been recently tasked with breaking a code with which our country's enemies are communicating. He has thought of a possible break in the code, using a very complex system of strings, which thankfully, you have nothing to do with. You are tasked with a little problem. Jack will give you n strings, labelled S1, S2, …. Sn, along with q queries. In each query, you have an integer X and a string CODE. You will take into consideration strings S1 to SX. Among these selected strings, consider all the strings such that their longest common prefix with CODE is the maximum possible. Now, from these strings, print the lexicographically smallest one. This would give Jack tremendous insight into further breaking the code. Can you help him?

Input Format:

The first line of the input contains a single integer N.
N lines follow. For each valid i, the i-th of these lines contains Chef's string Si.
The next line contains a single integer Q.
The following Q lines describe queries. Each of these lines contains an interger R, followed by a space and a string P

Constraints:

1 ≤ n ≤ 100,000
1 ≤ |Si| ≤ 10 for each valid i
1 ≤ q ≤ 100,000
1 ≤ X ≤ n
1 ≤ |CODE| ≤ 10

Output Format:

For each query, print a single line containing the string that satisfies the required conditions — the answer to that query.

Sample Input 1:

```
4
abcd
abce
abcdex
abcde
3
3 abcy
3 abcde
4 abcde
```

Sample Output 1:

```
abcd
abcdex

abcde
```

1. **#include<bits/stdc++.h>**
2. **using namespace std;**
3. **struct trie**

```
4.  {
5.      int cnt;
6.      trie *arr[26] = {NULL};
7.      trie() : cnt(0) {}
8.  };
9.
10. void insert(trie *root, string &s)
11. {
12.     for(int i=0;i<s.length();i++)
13.     {
14.         int val = (s[i] - 'a');
15.         if(root->arr[val] == NULL)
16.         {
17.             root->arr[val] = new trie();
18.         }
19.         root = root->arr[val];
20.     }
21.     root->cnt +=1;
22. }
23.
24. string search(trie *root, string &s)
25. {
26.     string ans = "";
27.     for(int i=0;i<s.length();i++)
28.     {
29.         if(root->arr[s[i]-'a']!=NULL)
30.         {
31.             root = root->arr[s[i]-'a'];
32.             ans += s[i];
33.         }
34.         else
35.         {
36.             break;
37.         }
38.     }
39.     if(root->cnt)
40.     {
41.         return ans;
42.     }
43.     for(int i=0;i<26;i++)
44.     {
45.         if(root->arr[i]!=NULL)
46.         {
47.             root = root->arr[i];
```

```
            char c = 'a';
            c += i;
            i = -1;
            ans += c;
            if(root->cnt)
               return ans;

            continue;
        }
    }
    return ans;
}
int main()
{
    int n, q;
    cin>>n;
    string s[n+1];
    trie *root = new trie;
    for(int i=1;i<=n;i++)
    {
        cin>>s[i];
    }
    cin>>q;
    vector<pair<string, int>> v[n+1];

    for(int i=1;i<=q;i++)
    {
        int r;
        string s;
        cin>>r>>s;
        v[r].push_back({s, i});
    }
    string ans[q+1];
    for(int i=1;i<=n;i++)
    {
        int sz = v[i].size();
        insert(root, s[i]);
        for(int j=0;j<sz;j++)
        {
            string temp = search(root, v[i][j].first);
            ans[v[i][j].second] = temp;
        }
    }
    for(int i=1;i<=q;i++)
```

```
    {
        cout<<ans[i]<<endl;
    }
    return 0;

}
```

6-Ass : Ninja and Multiset

Send Feedback

Our problem setter, Ninja, is fascinated by different types of sets. Quite recently, he came across a new term- Multiset. Multiset is a set, where equal elements are allowed. He started experimenting with it and has managed to frame a pretty interesting problem for you to solve.

You are given a multiset G, initially containing only 0, and a few queries q. Queries are actually of 3 types:

"+ x" — This query adds an integer x to the given multiset.
"- x" — This query erases one occurrence of x from the given multiset. Before this query is encountered, it is guaranteed that the multiset will contain integer x at least once.
"? x" — In this query, you are given integer x and you need to compute the maximum value of bitwise exclusive OR (also known as XOR) of integer x and some integer y from the given multiset.

Input Format:

First line on input contain Q (number of queries).
Next Q line contain Q queries as defined above.

Output Format:

For each query of type '?' print the answer in new line

Constraints:

1 <= Q <= 10^5
1 <= x <= 10^9

Sample Input 1:

```
8
? 8
? 5
+ 10
? 5
? 4
+ 4
- 4
? 10
```

Sample Output 1:

```
8
5
15
14
10
```

Explanation:

Initially the multiset contains only the integer 0. So for the first query, answer is 8 XOR 0 = 8.
Similarily, the answer for second query is 5 XOR 0 = 5.

After the third query, 10 is added to the multiset.
For the fourth query, 5 will be XOR'ed with 0 and 10. Out of these, 5 XOR 10 = 15 is max.
Similarily, for the fifth query, 4 will be XOR'ed with 0 and 10. Out of these, 4 XOR 10 = 14 is max.
After the sixth query, 4 is added to the multiset.
After the seventh query, 4 is erased from the multiset.

For the last query, 10 is XOR'ed with 0 and 10. Out of these, 10 XOR 0 = 10 is max.

```
#include<bits/stdc++.h>
using namespace std;

struct trie
{
    trie *left = NULL, *right = NULL;
    int count = 0;
    void insert(int index, int val)
    {
        if(index < 0)
        {
            return ;
        }
        int a = (1ll << index);
        int b = (a & val);
        if(b == 0)
        {
            if(left == NULL)
                left = new trie();
            left->count++;
            left->insert(index-1,val);
        }
        else
        {
            if(right == NULL)
              right = new trie();
            right->count++;
            right->insert(index-1,val);
        }
    }

    void delete_(int index, int val)
    {
        if(index<0)
          return;
        int a = (1ll << index), b = (a & val);
        if(b == 0)
```

```
38.       {
39.          if(left == NULL)
40.            left = new trie();
41.          left->count--;
42.          left->delete_(index-1,val);
43.       }
44.       else
45.       {
46.          if(right == NULL)
47.             right = new trie();
48.          right->count--;
49.          right->delete_(index-1,val);
50.       }
51.    }
52.
53.    int max_(int index, int val)
54.    {
55.       int ans = 0;
56.       if(index < 0)
57.          return 0;
58.       int a = (1ll << index), b = (a & val);
59.       if(b == 0)
60.       {
61.          if(right && right->count > 0)
62.          {
63.             ans += (a);
64.             ans += (right->max_(index-1,val));
65.          }
66.          else if(left && left->count > 0)
67.          {
68.             ans += (left->max_(index-1,val));
69.          }
70.          else
71.             return ans;
72.       }
73.       else
74.       {
75.          if(left && left->count > 0)
76.          {
77.             ans += (a);
78.             ans += (left->max_(index-1,val));
79.          }
80.          else if(right && right->count > 0)
81.          {
```

```
82.                 ans += (right->max_(index-1,val));
83.             }
84.             else
85.                 return ans;
86.         }
87.         return ans;
88.     }
89. };
90. int main()
91. {
92.     trie *head = new trie();
93.     int q;
94.     cin>>q;
95.     while(q--)
96.     {
97.         char x,y;
98.         int a,b,c,d;
99.         cin>>x;
100.            cin>>a;
101.            if(x == '+')
102.            {
103.                head->insert(31,a);
104.            }
105.            else if(x == '-')
106.            {
107.                head->delete_(31,a);
108.            }
109.            else
110.            {
111.                b = max(a, head->max_(31,a));
112.                cout<<b<<endl;
113.            }
114.        }
115.    }
```