

## DQUERY - D-query

[#sorting](#) [#tree](#)

[English](#)

[Vietnamese](#)

Given a sequence of  $n$  numbers  $a_1, a_2, \dots, a_n$  and a number of d-queries. A d-query is a pair  $(i, j)$  ( $1 \leq i \leq j \leq n$ ). For each d-query  $(i, j)$ , you have to return the number of distinct elements in the subsequence  $a_i, a_{i+1}, \dots, a_j$ .

### Input

- Line 1:  $n$  ( $1 \leq n \leq 30000$ ).
- Line 2:  $n$  numbers  $a_1, a_2, \dots, a_n$  ( $1 \leq a_i \leq 10^6$ ).
- Line 3:  $q$  ( $1 \leq q \leq 200000$ ), the number of d-queries.

- In the next  $q$  lines, each line contains 2 numbers  $i, j$  representing a d-query ( $1 \leq i \leq j \leq n$ ).

## Output

- For each d-query  $(i, j)$ , print the number of distinct elements in the subsequence  $a_i, a_{i+1}, \dots, a_j$  in a single line.

## Example

### Input

```
5
1 1 2 1 3
3
1 5
2 4
3 5
```

### Output

```
3
2
3
```

 Submit solution!

[hide comments](#)



[adityaghosh96](#): 2017-01-03 04:50:00  
O( $n \cdot \sqrt{n}$ ) with scanf,printf.



[harisagar](#): 2016-12-28 09:57:28  
use faster io(not cin,cout).....

**Last edit: 2016-12-28 09:58:05**



[rihaz\\_zahir](#): 2016-12-28 09:21:22  
std::ios\_base::sync\_with\_stdio(false);  
use above code in main for fast I/O in c++,  
this will run cin/cout faster than scanf/printf.  
google search for details.



[hamjosh1](#): 2016-12-27 21:41:18  
fast io to use in cpp



[vengatesh15](#): 2016-12-22 17:08:02  
my first MO type Problem AC after 3WA



[testing java](#): 2016-12-13 12:58:03  
Nice problem, not so nice time limit. Wasted a lot of time on making java solution fast enough.



[rihaz\\_zahir](#): 2016-12-05 09:32:01  
where can i find my previous submission for this question?



[davidgalehouse](#): 2016-11-15 05:44:54  
0.17s with C++, TLE with C# despite very similar benchmarking on my local

machine for a 30k/200k test case... I don't get how, given other ACs with much higher times like .5 or .7... Also the source array is malformed, you'll have to trim before splitting or remove empty entries if trying C#.

**Last edit: 2016-11-15 05:47:16**



[oakszyjnrndy](#): 2016-10-26 15:59:26

It's very easy to solve this problem using a data structure called 'Zhuxi Tree' in Chinese(sorry~, i don't know how to call it in English, maybe Chair Tree?).

Time:  $O(n \cdot \log n + q \cdot \log n)$ .

Space:  $O(n \cdot \log n)$ .

**Last edit: 2016-10-26 16:00:56**



[vicennial](#): 2016-10-24 19:20:21

How are people solving it with 0.3+ running time when the time limit itself is 0.227 seconds??

 Submit solution!

Added by: [Duc](#)

Date: 2008-10-26

Time limit: 0.227s  
Source limit: 50000B  
Memory limit: 1536MB  
Cluster: [Cube \(Intel G860\)](#)  
Languages: All except: ERL JS NODEJS  
PERL 6 VB.net  
Resource: © [VNOI](#)

Con  
imp  
you

[About](#) | [Tutorial](#) | [Tools](#) | [Clusters](#) | [Credits](#) | [Jobs](#) | [API](#) | [Terms](#)

 [RSS](#)

© Spoj.com. All Rights Reserved. Spoj uses Sphere Engine™ © by Sphere Research Labs.