## Решење и анализа

За сваки интервал, проласком од левог до десног краја, можемо наћи број различитих бројева које садржи тако што ћемо бројати само првo појављивањe сваког броја. Како ћемо знати да је неко појављивање прво? Знамо да је низ сортиран, па важи да је појављивање броја прво ако је тај број или први број унутар интервала, или је број различит од свог претходника (броја лево од њега). Временска сложеност по упиту је *О( N )*, па је сложеност овог решења *О( N ∙ Q )*. Меморијска сложеност је *О( N ).* Ово решење доноси 30 поена.

Главни део кода у *C++* за ово решење је:

*for ( int i = 0; i < q; i++ )*

*{*

*scanf( "%d %d", &leva\_granica, &desna\_granica );*

*int resenje = 1;*

*for ( int x = leva\_granica + 1; x <= desna\_granica; x++ )*

*if ( niz[ x ] != niz[ x - 1 ] ) resenje++;*

*printf( "%d\n", resenje );*

*}*

Нека је *cnt[ K ][ I ]* број појављивања броја *K* у низу на позицијама од 1 до *I*. Важи следеће:

*cnt[ K ][ I ] = cnt[ K ][ I – 1 ], ако је niz[ I ] ≠ K*

*= cnt[ K ][ I – 1 ] + 1, ако је niz[ I ] = K*

Користећи ово, *cnt* можемо одредити у сложености *О( max\_num ∙ N )* где је *max\_num* највећи број у низу. За сваки интервал можемо у сложености *О( 1 )* да одредимо да ли се број *K* појављује унутар интервала тако што је број појављивања броја *K* унутар *[ L, R ]* једнак *cnt[ K ][ R ] – cnt[ K ][ L – 1 ].* Сада на сваки упит можемо одговорити проласком од 1 до *max\_num* и провером за сваки број да ли се појављује унутар интервала из упита. Временска сложеност овог решења је *О( Q ∙ max\_num + N ∙ max\_num )*, а меморијска *О( N ∙ max\_num ).* Ово решење такође вреди 30 поена, али може се применити у комбинацији са претходним решењем тако што уколико је *max\_num* ≤ 500 радимо у сложености *О( Q ∙ max\_num + N ∙ max\_num )*, а иначе у сложености *О( N ∙ Q ),* и оваква комбинација вреди 60 поена.

Kод у *C++* овог решења је:

*for ( int num = 1; num <= 500; num++ )*

*{*

*cnt[ num ][ 0 ] = ( niz[ 0 ] == num );*

*for ( int i = 1; i < n; i++ ) cnt[ num ][ i ] = cnt[ num ][ i - 1 ] + ( niz[ i ] == num );*

*}*

*scanf( "%d", &q );*

*for ( int i = 0; i < q; i++ )*

*{*

*scanf( "%d %d", &leva\_granica, &desna\_granica );*

*int resenje = 0, found;*

*for ( int num = 1; num <= 500; num++ )*

*{*

*found = cnt[ num ][ desna\_granica ];*

*if ( leva\_granica ) found -= cnt[ num ][ leva\_granica - 1 ];*

*resenje += ( found > 0 );*

*}*

*printf( "%d\n", resenje );*

*}*

Анализирајмо сада решење за 100 поена. Посматрајмо само прво појављивање сваког броја у низу.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 2 | 5 | 5 | 5 | 7 | 8 | 9 | 10 | 10 | 11 | 12 |

Пример

Нека је *dp[ I ]* број ових првих појављивања у низу на позицијама од 1 до *I*.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 2 | 2 | 2 | 3 | 4 | 5 | 6 | 6 | 7 | 8 |

Низ *dp* за претходни пример

Број различитих бројева на интервалу једнак је броју првих појављивања на интервалу. Већ смо рекли да је појављивање броја прво ако је тај број или први број унутар интервала, или је број различит од свог претходника. Број оних који су различити од свог претходника на интервалу *[ L, R ]* једнак је *dp[ R ] – dp[ L – 1 ].* Међутим, у случају да је *niz[ L ] = niz[ L – 1 ]*, тада решење треба увећати за 1 јер нисмо рачунали број са индексом *L* јер је он једнак свом претходнику, а треба га рачунати зато што је први број унутар интервала. Дакле, решење за интервал *[ L, R ]* је:

*dp[ R ] – dp[ L – 1 ], ако је niz[ L ] ≠ niz[ L – 1 ]*

*dp[ R ] – dp[ L – 1 ] + 1, ако је niz[ L ] = niz[ L – 1 ]*

Низ dp креирамо у сложености *О( N ),* и када имамо тај низ, на упите одговарамо у *О( 1 ).* Временска сложеност овог решења је *О( N + Q ),* а меморијска *О( N ).*

Код у *C++* за ово решење:

*for ( int i = 1; i < n; i++ )*

*{*

*if ( niz[ i ] != niz[ i - 1 ] ) broj\_razlicitih[ i ] = broj\_razlicitih[ i - 1 ] + 1;*

*else broj\_razlicitih[ i ] = broj\_razlicitih[ i - 1 ];*

*}*

*scanf( "%d", &q );*

*for ( int i = 0; i < q; i++ )*

*{*

*scanf( "%d %d", &leva\_granica, &desna\_granica );*

*int resenje = broj\_razlicitih[ desna\_granica ];*

*if ( leva\_granica )*

*{*

*resenje -= broj\_razlicitih[ leva\_granica - 1 ];*

*if ( niz[ leva\_granica ] == niz[ leva\_granica - 1 ] ) resenje++;*

*}*

*printf( "%d\n", resenje );*

*}*