Insert at position

Your are given a sorted array and a target value.

You have to find the position of the given element and if not found , print where it would be if inserted in sorted in the same sorted order.

Input consists of an integer N the number of elements and target value T

followed by sorted array elements.

Position of the element if found , else the expected position.

4 3

1 3 4 8

1

1<=N<=10^5

0<=arr[i]<=10^7

Here position of 3 is 1.

#include<bits/stdc++.h>

using namespace std;

int searchInsert(vector<int> &A, int B) {

int high = A.size() - 1;

int low = 0;

int mid;

while(low < high) {

mid = low + (high - low)/2;

if(A[mid]==B) {

return mid;

}

else if(A[mid] > B) {

high = mid - 1;

}

else {

low = mid + 1;

}

}

if(B > A[low])

return low + 1;

else

return low;

}

int main() {

int n, a, b;

vector<int>v;

cin >> n >> b;

for(int i = 0; i < n; i++) {

cin >> a;

v.push\_back(a);

}

cout<<searchInsert(v, b);

}

#include<bits/stdc++.h>

using namespace std;

int searchInsert(vector<int> &V, int B) {

}

int main() {

int n, a, b;

vector<int>v;

cin >> n >> b;

for(int i = 0; i < n; i++) {

cin >> a;

v.push\_back(a);

}

cout<<searchInsert(v, b);

}

- Here the idea used is simply Binary Search but a little twist.

- We need to find the expected position if element is not found.

- Our search stops when Low = High ,

- Now, There can be two cases :

- Element is between low and high - in this case answer is low itself.

- Second case is Target element is greater than low and high - therefore the answer is low +1 ;

<br>

Code:

<pre><code class="lang-cpp">

#include&lt;bits/stdc++.h&gt;

#define ll long long int

using namespace std;<br>

int searchInsert(vector<int> &A, int B) {

int high = A.size() - 1;

int low = 0;

int mid;

while(low < high) {

mid = low + (high - low)/2;

if(A[mid]==B) {

return mid;

}

else if(A[mid] > B) {

high = mid - 1;

}

else {

low = mid + 1;

}

}

if(B > A[low])

return low + 1;

else

return low;

}

int main() {

int n, a, b;

vector<int>v;

cin >> n >> b;

for(int i = 0; i < n; i++) {

cin >> a;

v.push\_back(a);

}

cout<<searchInsert(v, b);

}