3. Given 2 arrays A and B, each of size n.

a) Design an algorithm to test whether there is at least one common element between 2 arrays

- Sort the two arrays

- Iterate the two arrays until finding the first common element, set count=1 and break the loop.

Pseudocode:

**COMMON-SEARCH(A,B)**

1. // Sort 2 arrays
2. A. sort and B.sort
3. **while** i<=A. sort.length and j<=B. sort.length
4. **if** (A.sort[i]=B.sort[j])
5. i=i+1
6. j=j+1
7. count= 1
8. **break**
9. **else if** (A.sort[i]>B.sort[j])
10. j=j+1
11. **else** (A.sort[i]<B.sort[j])
12. i=i+1
13. **if** count==0
14. **Print** “no common elements”
15. **else**
16. **Print** “there is at least one common element”

b) Prove its correctness

The **loop variant** : at the start of each iteration of the while loop line 3-12, the value of *count* is 0

* Initialization: before the first loop iteration, *count*=0
* Maintenance: the only way to terminate the loop is if A.sort[i]=B.sort[j], the loop invariant is preserved for the next iteration.
* Termination: the loop terminates because of only two cases

+ first case: *count* different than 0 (in this code we choose *count*=1)

+ second case: i>A.sort.length or j>B.sort.length meaning *count* still be 0

The output of the first case is “*there is at least one common element*”, and the output of second case is “*no common element*”. Thus the algorithm is correct.

**Test by Matlab Code**:

|  |
| --- |
| A=[1 5 3 -8 9 4];  B=[0 -1 6 2 3 10];  Asort=sort(A);  Bsort=sort(B);  i=1;j=1;  count=0;  while(i<=length(Asort)&& j<=length(Bsort))  if(Asort(i)==Bsort(j))  i=i+1;  j=j+1;  count= 1;  break  else if (Asort(i)>Bsort(j))  j=j+1;  else  i=i+1;  end  end  end  if count==0  fprintf('there is no common elements');  else  fprintf('there exists at least one common element');  end |

Result:

>> there exists at least one common element

c) Estimate speed

- Sort the two arrays, O(n logn) time complexity (indeed, its value is twice that but we can remove the constant)

- Iterate the two arrays at the same time and compare two elements of arrays. If they are same, then we notify the match and break the loop. Otherwise, increasing the pointer of the array, which has the smaller value at that time. The worse time is O(n).

Thus, the worse time for the algorithm is O(n logn)