CALLICODER

Find the pair with the smallest difference in two unsorted arrays

by RAJEEV SINGH · ALGORITHMS · NOVEMBER 03, 2019 · 1 MINS READ

#algorithm
Find the pair with the smallest difference in two unsorted arrays

Given two non-empty arrays of integers, find the pair of values (one value from each array) with the smallest (non-negative) difference.

Example

```
Input: [1, 3, 15, 11, 2], [23, 127, 235, 19, 8]

Output: [11, 8]; this pair has the smallest difference.
```

Solution 1. Brute Force approach: Use two for loops

The naive way to solve this problem is to use two for loops and compare the difference of every pair to find the pair with the smallest difference:

Time complexity: O(n^2)

```
import java.util.Arrays;

class SmallestDifference {

  public static int[] findSmallestDifferencePair_Naive(int[] a1, int[] a2) {
    double smallestDiff = Double.MAX_VALUE;
    int[] smallestDiffPair = new int[2];

    for(int i = 0; i < a1.length; i++) {
      for(int j = 0; j < a2.length; j++) {
        int currentDiff = Math.abs(a1[i] - a2[j]);
        if(currentDiff < smallestDiff) {
          smallestDiff = currentDiff;
          smallestDiffPair[0] = a1[i];
          smallestDiffPair[1] = a2[j];
        }
      }
    }
    return smallestDiffPair;
  }

  public static void main(String[] args) {
    int[] a1 = new int[] {-1, 5, 10, 20, 28, 3};
    int[] a2 = new int[] {26, 134, 135, 15, 17};

    int[] pair = findSmallestDifferencePair_Naive(a1, a2);
```

```
      System.out.println(pair[0] + " " + pair[1]);
    }
  }
```

Solution 2. Use Sorting along with the two-pointer approach

You can improve upon the brute force solution by first sorting the array and then using the two-pointer pattern.

Here is how it will work in this case:

- Initialize a variable to keep track of the smallest difference found so far (`smallestDiff`).
- Sort both the arrays
- Initialize two pointers/indexes (one for each array): `i = 0` and `j = 0`.
- Loop until we reach the end of any of the arrays.
- For every iteration:
    - Compare the `smallestDiff` with `abs(a1[i] - a2[j])` and reset it if the new difference is smaller.

- If `a1[i] < a2[j]`, increment `i`.
- Otherwise, increment `j`

Time complexity: O(mlog(m) + nlog(n))

```
import java.util.Arrays;

class SmallestDifference {
  public static int[] findSmallestDifferencePair(int[] a1, int[] a2) {
    Arrays.sort(a1);
    Arrays.sort(a2);

    double smallestDiff = Double.MAX_VALUE;
    int[] smallestDiffPair = new int[2];
    int i = 0, j = 0;

    while(i < a1.length && j < a2.length) {
      double currentDiff = Math.abs(a1[i] - a2[j]);
      if(currentDiff < smallestDiff) {
        smallestDiff = currentDiff;
        smallestDiffPair[0] = a1[i];
        smallestDiffPair[1] = a2[j];
      }
      if(a1[i] < a2[j]) {
        i++;
      } else {
        j++;
```

```
            }
        }
        return smallestDiffPair;
    }

    public static void main(String[] args) {
        int[] a1 = new int[] {-1, 5, 10, 20, 28, 3};
        int[] a2 = new int[] {26, 134, 135, 15, 17};

        int[] pair = findSmallestDifferencePair(a1, a2);
        System.out.println(pair[0] + " " + pair[1]);
    }
}
```

Share on social media