Find the minimum distance between two numbers

Given an unsorted array arr[] and two numbers x and y, find the minimum distance between x and y in arr[]. The array might also contain duplicates. You may assume that both x and y are different and present in arr[].

Examples:

Input: arr[] = {1, 2}, x = 1, y = 2

Output: Minimum distance between 1 and 2 is 1.

Input: arr[] = {3, 4, 5}, x = 3, y = 5

Output: Minimum distance between 3 and 5 is 2.

Input: arr[] = {3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3}, x = 3, y = 6

Output: Minimum distance between 3 and 6 is 4.

Input: arr[] = {2, 5, 3, 5, 4, 4, 2, 3}, x = 3, y = 2

Output: Minimum distance between 3 and 2 is 1.

Method 1 (Simple)

Use two loops: The outer loop picks all the elements of arr[] one by one. The inner loop picks all the elements after the element picked by outer loop. If the elements picked by outer and inner loops have same values as x or y then if needed update the minimum distance calculated so far.

```
#include <stdio.h>
#include <stdlib.h> // for abs()
#include <limits.h> // for INT_MAX

int minDist(int arr[], int n, int x, int y)
{
   int i, j;
   int min_dist = INT_MAX;
   for (i = 0; i < n; i++)
   {
     for (j = i+1; j < n; j++)
     {
         if( (x == arr[i] && y == arr[j] ||
              y == arr[i] && x == arr[j]) && min_dist > abs(i-j))
         {
              min_dist = abs(i-j);
         }
     }
   }
```

```
    return min_dist;
}

/* Driver program to test above fnction */
int main()
{
    int arr[] = {3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3};
    int n = sizeof(arr)/sizeof(arr[0]);
    int x = 3;
    int y = 6;

    printf("Minimum distance between %d and %d is %d\n", x, y,
              minDist(arr, n, x, y));
    return 0;
}
```

Run on IDE

Output: *Minimum distance between 3 and 6 is 4*

Time Complexity: O(n^2)

Method 2 (Tricky)

1) Traverse array from left side and stop if either *x* or *y* is found. Store index of this first occurrence in a variable say *prev*

2) Now traverse *arr[]* after the index *prev*. If the element at current index *i* matches with either x or y then check if it is different from *arr[prev]*. If it is different then update the minimum distance if needed. If it is same then update *prev* i.e., make *prev* = *i*.

Thanks to wgpshashank for suggesting this approach.

```
#include <stdio.h>
#include <limits.h>  // For INT_MAX

int minDist(int arr[], int n, int x, int y)
{
   int i = 0;
   int min_dist = INT_MAX;
   int prev;

   // Find the first occurence of any of the two numbers (x or y)
   // and store the index of this occurence in prev
   for (i = 0; i < n; i++)
   {
     if (arr[i] == x || arr[i] == y)
     {
       prev = i;
       break;
     }
   }

   // Traverse after the first occurence
   for ( ; i < n; i++)
   {
      if (arr[i] == x || arr[i] == y)
      {
          // If the current element matches with any of the two then
          // check if current element and prev element are different
          // Also check if this value is smaller than minimm distance so far
          if ( arr[prev] != arr[i] && (i - prev) < min_dist )
```

```
            {
               min_dist = i - prev;
               prev = i;
            }
            else
               prev = i;
         }
      }

      return min_dist;
}

/* Driver program to test above fnction */
int main()
{
    int arr[] ={3, 5, 4, 2, 6, 3, 0, 0, 5, 4, 8, 3};
    int n = sizeof(arr)/sizeof(arr[0]);
    int x = 3;
    int y = 6;

    printf("Minimum distance between %d and %d is %d\n", x, y,
              minDist(arr, n, x, y));
    return 0;
}
```

Run on IDE

Output: *Minimum distance between 3 and 6 is 1*

Time Complexity: O(n)

Please write comments if you find the above codes/algorithms incorrect, or find other ways to solve the same problem.

136 Comments Category: Arrays

Related Posts:

- Longest Span with same Sum in two Binary arrays
- Count Inversions of size three in a give array
- Find the subarray with least average
- Count triplets with sum smaller than a given value
- Find zeroes to be flipped so that number of consecutive 1's is maximized
- Reorder an array according to given indexes
- Find maximum value of Sum(i*arr[i]) with only rotations on given array allowed
- Find maximum average subarray of k length

(Login to Rate and Mark)

2 Average Difficulty : **2/5.0**
Based on **10** vote(s)

Add to TODO List

Mark as DONE

Writing code in comment? Please use code.geeksforgeeks.org, generate link and share the link here.