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;
}
```

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* are found. Store index of this first occurrrence 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
```


Popular Posts

```
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
          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;
}
```

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.

Related Tpoics:

- Remove minimum elements from either side such that 2*min becomes more than max
- Divide and Conquer | Set 6 (Search in a Row-wise and Column-wise Sorted 2D Array)
- Bucket Sort
- Kth smallest element in a row-wise and column-wise sorted 2D array | Set 1
- Find the number of zeroes
- Find if there is a subarray with 0 sum
- Divide and Conquer | Set 5 (Strassen's Matrix Multiplication)
- Count all possible groups of size 2 or 3 that have sum as multiple of 3

Recent Comments

Writing code in comment? Please use **ideone.com** and share the link here.

84 Comments GeeksforGeeks

Sort by Newest

Join the discussion…

Lohith Ravi • 17 days ago

Step 1: add X+y

Step 2
now for all elements in the array, {
check if (sum - element) is there in the array and what is its index
and if difference in the index is less than the previous difference (initially keep t
also store them in the hashmap, the key is the element and the value is the cu
position incase of duplicates]
}
the minimum difference that is store print it

amazon_it_is • 3 months ago

```
#include <cstdlib>
#include <iostream>

using namespace std;

int main(int argc, char *argv[])
{
int a[]={3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3};
int s=12;
```

int n1=3;
int n2=6;
int index1=-1;
int index2=-1;
int q=10000;
for(int i=0;i<s;i++){ if(a[i]="=n1)" {="" index1="i;" }="" else="" if(a[i]="=n2){" inde
index2!="-1){" q="q<abs(index1-index2)?q:abs(index1-index2);" }="" }="" cou
exit_success;="" }="">

• Reply • Share ›

Code_Addict • 4 months ago

Nice solution. Thanks

• Reply • Share ›

anonymous • 5 months ago

What if the array is :
{3, 5, 4, 6, 5, 6, 3}, and elements are 3 and 6.
Will the minimum be 1 or 3. My question is that does the relative order of the t
while calculating, minimum comes out as 1. and 3 otherwise.

• Reply • Share ›

guru • 7 months ago

int minimum(int _arr_size, int* _arr, int _a, int _b) {
int i=0,j=0;
int n=_arr_size,min=n,temp;
while(i<n) {="" if(_arr[i]="=_a)" {="" j="i+1;" while(j<n)="" {="" if(_arr[j]="=_b)" {=
min="temp;" break;="" }="" else="" j++;="" }="" }="" else="" if(_arr[i]="=_b)" {="
if(_arr[j]="=_a)" {="" temp="j-i-1;" if(temp<min)="" min="temp;" break;="" }="" e
return="" min;="" }="">

• Reply • Share ›

Neha Garg • 7 months ago

your solution will not work for dup elements... If array is (1, 5, 3, 7, 2, 8, 3, 4, 5, min_distance(9,9) should be zero

Reply • Share ›

jinzhi chen ➦ Neha Garg • 5 months ago

in that case, no point find min_distance, every min_distance is 0

1 Reply • Share ›

jackSparrow ➦ Neha Garg • 7 months ago

author has clearly specified "You may assume that both x and y are dif

1 Reply • Share ›

Guest • 8 months ago

```
#include<stdio.h>
#include<limits.h>
#include<stdlib.h>

int min_num(int a,int b)
{
    if(a<b) return="" a;="" else="" return="" b;="" }="" int="" min_d
```

Reply • Share ›

Guest • 8 months ago

could some1 pls validate the following code

#include<stdio.h>
#include<limits.h>
#include<stdlib.h>

int min_num(int a,int b)
{
if(a<b) return="" a;="" else="" return="" b;="" }="" int="" min_dist(int="" arr[],int=

t1,i,j,ind1,mind="INT_MAX;" for(i="0;i<n;i++)" {="" if(arr[i]="=a)" {="" t1="a;" in
if(arr[i]="=b)" {="" t1="b;" ind1="i;" break;="" }="" }="" for(j="i+1;j<n;j++)" {="" i
{="" t1="a;" ind1="j;" }="" else="" if((t1="=b)&&(arr[j]==b))" {="" t1="b
if((t1="=a)&&(arr[j]==b))" {="" mind="min_num(mind,abs(ind1-j));" t1
if((t1="=b)&&(arr[j]==a))" {="" mind="min_num(mind,abs(ind1-j));" t1
if(mind!="INT_MAX)" return="" mind;="" else="" return="" -1;="" }="" void="" ma
arr[]="{2," 5,="" 3,="" 5,="" 4,="" 4,="" 2,="" 3};="" int="" n="sizeof(arr)/sizeof(ar
1)" printf("mindist="" is="" %d",res);="" else="" printf("either="" of="" 2="" nums
}="">

• Reply • Share ›

SudhanshuAnand • 9 months ago

Below is another solution in linear time

```
int min_dist(int a[], int size, int x, int y){
       int minDist;
       int ix, iy, i;
       for(i=0;i<size;i++)
              if(a[i] == x){
                     ix = i;
                     break;
              }
       for(i=0;i<size;i++)
              if(a[i] == y){
                     iy = i;
                     break;
              }
       minDist = abs(ix - iy);
       int start = max(ix, iy);
       for(i=start;i<size;i++){
```

see more

rajat6875 • 10 months ago

```
class ArrayDis{

        public static int minDis(int arr[],int x,int y){
                int start=-2,end=-1,minDis=Integer.MAX_VALUE;
                        boolean flag=true;
                        for(int i=0;i<arr.length;i++){
                                if((arr[i]==x||arr[i]==y)&&flag)
                                {
                                        start=i;
                                        flag=false;
                                }
                                else if((arr[i]==x||arr[i]==y)&&!flag
                                {
                                        end=i;
                                        flag=true;
                                }
                                if(start>-1&&end>-1)
                                        minDis=Math.min(minDis,Math.al
```

see more

rajat6875 • 10 months ago

i tried to implement it in java if somebody finds some error or wrong output for

```
class ArrayDis{

        public static int minDis(int arr[],int x,int y){
                int start=-2,end=-1,minDis=Integer.MAX_VALUE;
                        boolean flag=true;
```

```
boolean flag=true;
for(int i=0;i<arr.length;i++){
        if((arr[i]==x||arr[i]==y)&&flag)
        {
                start=i;
                flag=false;
        }
        else if((arr[i]==x||arr[i]==y)&&!flag)
        {
                end=i;
                flag=true;
        }
```

see more

• Reply • Share ›

akshaykumar • 10 months ago

How to modify this program if x and y can be equal ?

• Reply • Share ›

Akhil • 11 months ago

The code given above is quite complicated!
Using LastX for last X found, and LastY for Last Y found:
(assuming X and Y are the numbers to be found)
We can keep the code quite simple and clear.

```
 #include<stdio.h>
#include<stdlib.h>
int findMinimum(int a[], int size,int x, int y)
{
    int min = size;
    int LastX = 4*size;
    int LastY = 2*size;
```

```
    // So that min doesnt change till both numbers have been       fou
    for(i=0;i<size;i++)
    {
        if(a[i]==x)
            LastX = i;
        else if(a[i]==y)
```

see more

• Reply • Share ›

raghson • 11 months ago

```
 #include<stdio.h>
#include<stdbool.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++)
   {
       if(arr[i]!=x)
         break;

     for (j = 0; j < n; j++)
     {
         if( y == arr[j] && min_dist > abs(i-j))
         {
```

see more

• Reply • Share ›

Ganesh • a year ago

You can find java code here:

[sourcecode language="JAVA"]
/**
* Given an unsorted array arr[] and two numbers x and y, find the minimum dis
* The array might also contain duplicates. You may assume that both x and y
* Example:
* Input: arr[] = {1, 2}, x = 1, y = 2
* Output: Minimum distance between 1 and 2 is 1.
* @author GAPIITD
*
*/
public class MinimumDistanceBetweenTwoNumbers {

public static void main(String[] args) {
int arr[] = {3, 5, 4, 2, 6, 3, 0, 0, 5, 4, 8, 3};
System.out.println(MinDistance(arr, 3, 6));
}

see more

• Reply • Share ›

Niks • a year ago

```
 #include<stdio.h>
#include<stdlib.h>
#include<algorithm>
#include<string.h>
using namespace std;


int minDist(int arr[], int n, int x, int y)
```

```
{
        int st[2] = {-1,-1};
        int ans = INT_MAX;

        for(int i=0; i<n; i++)
        {
                if(arr[i] == x && st[1]!=-1)
                {
                        ans = min(ans,i-st[1]);

                }
```

see more

Reply • Share ›

nikoo28 • a year ago

Please correct the typo in this post...

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

The minimum distance is 1 and the code gives a correct answer...

Reply • Share ›

GeeksforGeeks nikoo28 • a year ago

@nikoo28: Thanks for pointing this out. We have updated the post. Ke

Reply • Share ›

akshaykumar GeeksforGeeks • 10 months ago

How to modify this program if x and y can be equal ?

Reply • Share ›

J • 2 years ago

I am getting the answer as one the distance between 6 and 3

I am getting the answer as one the distance between 6 and 3 ....

```
/* Paste your code here (You may delete these lines if not writing co
```

Reply • Share ›

deep J • 2 years ago

ya both of code will give minimum distance as 1 between 3 & 6. if array
5, 4, 8, 3};
becoz none of the above code distinguisgh between 3,6 and 6,3

```
/* Paste your code here (You may delete these lines if not wri
```

Reply • Share ›

Tiger • 2 years ago

Why don't we use something like

finding the last occurance of x and first occurance of y and subtract the index

Finding the occurance can be done in O(logn)..

Reply • Share ›

Apoorv Tiger • 2 years ago

Array is unsorted, searching can't be done in O(log n) time

Reply • Share ›

bartender Apoorv • a year ago

Searching in an unodered array takes O(n) time complexity whi
algorithms in the post, os there is no harm in finding the last oc
time followed by first occurence of larger number in 0(n) time, r
O(n), which is equivalent to time complexity of method 2 of the

Reply • Share ›

Venki • 2 years ago

I missed small part of code, here is correct version

```
 void MinSampleAndHold(int &min, int exp) { if( min > exp ) min = exp

int GetMinimumDistance(int A[], int size, int x, int y)
{
    int minDiff = INT_MAX;
    int sum = x + y;
    int i = 0;
    int p; // previously traced element

    // X and Y are present in the array
    while( A[i] != x && A[i] != y )
        ++i;

    for( p = i, ++i; i < size; i++ )
    {
        if( sum - A[p] == A[i] )
```

see more

• Reply • Share ›

Venki • 2 years ago

The code can be improved.

1. We need to increment the i value before entering into second loop. Otherwis unnecessarily.
2. We don't need two comparisons inside if of second loop.

Idea is same. Here is relatively simple code,

```
 inline
// usually second parameter is an expression
void MinSampleAndHold(int &min, int exp) { if( min > exp ) min = exp;

int GetMinimumDistance(int A[], int size, int x, int y)
{
    int minDiff = INT_MAX;
    int sum = x + y;
    int i = 0;
    int p; // previously traced element
```

see more

• Reply • Share ›

kartikaditya • 2 years ago

[sourcecode language="C++"]
#include <iostream>
#include <stdio.h>

using namespace std;

int getMinDistance(int a[], int n, int x, int y) {
int posX = -1, posY = -1;
int mini = (int)(((unsigned) 1 << 31) - 1);
for (int i = 0; i < n; ++i) {
if (a[i] == x) {
posX = i;
if (posY != -1) {
if (posX - posY < mini) {
mini = posX - posY;
}
}

```
}
if (a[i] == x) {
```

see more

Reply • Share ›

prabhatlamba • 2 years ago

```
 #include<conio.h>
#include<stdio.h>
#define max 100
void main ()
{
        clrscr();
        int n,c=0,k=0,arr[max],x,y;
        scanf("%d",&n);
        int min=n;
        for (int i=1;i<=n;i++)
        {
        scanf("%d",&arr[i]);
        }

        printf("\nenter the values of x and y ");
        scanf("%d%d",&x,&y);
        for(int m=1;m<=n;m++)
        {
```

see more

1 • Reply • Share ›

Nagarjuna • 2 years ago

I think there is a bug in the actual (main) post - Test your code with the below array: 1 2 3 4 5 6 7 1 2 4, and no's: 1 & 4.
correct ans = 2; [distance between last pair of 1 and 4]

I guess your code would give 3.

To fix it: I guess you have to update prev always -

So in your for loop

```
for ( ; i < n; i++)
{
.....
.....
if ( arr[prev] != arr[i] && (i - prev) < min_dist )
{
min_dist = i - prev;
}
prev = i;
// removing the else part and reset prev always.
/* }
else
prev = i;
*/
}
}
```

 • Reply • Share ›

kartik ➚ Nagarjuna • 2 years ago

Dude, the porgam works fine for your input. The following driver progra

```
 /* Driver program to test above fnction */
int main()
{
    int arr[] ={1, 2, 3, 4, 5, 6, 7, 1, 2, 4};
    int n = sizeof(arr)/sizeof(arr[0]);
    int x = 1;
```

```
    int y = 4;

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

• Reply • Share ›

Nagarjuna kartik • 2 years ago

My bad Karthik, your logic works fine..

Just a readability suggestion - I guess you don't need an else h

```
if ( arr[prev] != arr[i]&&(i - prev) < min_dist )
{
min_dist = i - prev;
prev = i;
}
else
prev = i;
```

Instead -- you can write

```
if ( arr[prev] != arr[i]&&(i - prev) < min_dist ) {
min_dist = i - prev;
}
prev = i; // update prev always.
```

• Reply • Share ›

Anuj Bansal • 2 years ago

Here is a O(n) algorithm to solve the above problem

```
#include<stdio.h>
```

```
#include<math.h>
#define MAX 12

int MinDis(int a[MAX],int x, int y) {
        int i,min,posX,posY;
        min = MAX, posX = -1, posY = -1;
        for(i=0;i < MAX;i++) {
                if(a[i] == x) {
                        if(posX == -1)
                                posX = i;
                        else {
                                if(posY != -1)
                                        min = min < abs(posX-posY) ? r
                                else
                                        posX = i;
```

see more

• Reply • Share ›

Anuj Bansal ➦ Anuj Bansal • 2 years ago

The above code gives incorrect result for some input. here is the corre

```
 #include<stdio.h>
#include<math.h>
#define MAX 12

int MinDis(int a[MAX],int x, int y) {
        int i, min, posX,posY;
        min = MAX; posX = -1, posY = -1;
        for(i=0 ; i < MAX;i++) {
                if(a[i] == x) {
                        if(posX != -1 && posY != -1 && min > ab
```

```
                        min = abs(i-posY);
                posX = i;
        }
        else if(a[i] == y) {
                if(posX != -1 && posY != -1 && min > ab
                        min = abs(i-posX);
```

see more

• Reply • Share ›

anujbansal Anuj Bansal • 2 years ago

```
 #include<stdio.h>
#include<math.h>
#define MAX 12

int MinDis(int a[MAX],int x, int y) {
        int i, min, posX,posY;
        min = MAX; posX = -1, posY = -1;
        for(i=0 ; i < MAX;i++) {
                if(a[i] == x) {
                        if(posY != -1 && min > abs(i-pos
                                min = abs(i-posY);
                        posX = i;
                }
                else if(a[i] == y) {
                        if(posX != -1 && min > abs(i-pos
                                min = abs(i-posX);
                        posY = i;
                }
```

see more

• Reply • Share ›

amit • 3 years ago

@wgpshashank :
The solution assumes that the minimum distance between elements x & y wo
element spotted in the input array & some other element.

Check for input {3, 5, 4, 2, 6, 5, 6, 6, 5, 4, 8, 3}
and let x=5 and y=6

According to the code, the solution would come out as 3 {x_pos=1 and y_pos=
y_pos=4}

 • Reply • Share ›

kartik ➚ amit • 3 years ago

@amit: take a cloaser look at the program. It works fine for your input.

 • Reply • Share ›

radhakrishna • 3 years ago

```
xpos = -1
ypos = -1
mindist = +infinity
for i : 0 to n-1 do
if a[i] == x || a[i] == y then
if a[i] == x then
xpos = i;
else
ypos = i;
//valid positions found for x and y
if xpos != -1 and ypos != -1 then
diff = abs(xpos-ypos);
if(mindist > diff) then
mindist = diff;
```

sekhar • 3 years ago

```
public class MinDistanceBetweenNumbers {

/**
* @param args
*/
public static void main(String[] args) {
// TODO Auto-generated method stub

int a[] = {2,3,3,3,3,3,6,8,9,0,5,3,5,4,4,2,7,7,8,9,3};

int minDistance = 0;
int prev = -1;
int x = 3;
int y = 2;

for(int i = 0; i<a.length;i++){
/* Index element is equal to any of the element in X and Y*/
if(a[i] == x || a[i] == y ){
/*If the first element index is not at found */
```

see more

Reply • Share ›

radhakrishna • 3 years ago

```
public static int mindist(int[] a, int x, int y) {

int i =0 ;
int xpos = Integer.MAX_VALUE;
int ypos = Integer.MAX_VALUE;
int mindist= Integer.MAX_VALUE;
while(i Math.abs(xpos-ypos)) {
```

```
mindist = Math.abs(xpos-ypos);
}

i++;

}

return mindist;

}
```


Pandharinath • 3 years ago

```
int MinDist(int arr[], int len, int x,int y)
{
int vals[2]={-1,-1};
int mindist=len+1;;
for(int i=0; i< len ; i++)
{
if(arr[i]== x)
{
if(vals[1] != -1 && (i-vals[1]) < mindist )
{
mindist= i-vals[1];

}
vals[0]=i;
}
else if(arr[i] == y)
{
if(vals[0] != -1 && (i-vals[0]) < mindist )
```

see more

• Reply • Share ›

Ankur • 3 years ago

```
int minDistance(int a[],int n,int x,int y){
int xindex=INT_MAX,yindex=0;
int d = INT_MAX;
for(int i=0;i<n;i++){
if(a[i]==x || a[i]==y){
if(a[i]==x)
xindex=i;
if(a[i]==y)
yindex=i;

d = min(d,abs(xindex-yindex));
}
}
return d;
}
```

• Reply • Share ›

Sravan Vurapalli • 3 years ago

```
int FindMinimumDistance(int a[], int size,int key1, int key2)
{
int previous = -1;
int minimumDistance = INT_MAX;

for(int i = 0; i < size; i++)
{
if(a[i] == key1 || a[i] == key2)
{
if(previous < 0 || a[i] == a[previous])
```

```
{
previous = i;
continue;
}

if(i - previous < minimumDistance)
{
minimumDistance = i - previous;
}

previous = i;
}
}

return minimumDistance;
}
```


Sravan Vurapalli • 3 years ago

```
int FindMinimumDistance(int a[], int size,int key1, int key2)
{
int previous = -1;
int minimumDistance = INT_MAX

for(int i = 0; i < size; i++)
{
if(a[i] == key1 || a[i] == key2)
{
if(previous < 0 || a[i] == a[previous])
{
previous = i;
continue;
}
```

```
if(i - previous < minimumDistance)
{
minimumDistance = i - previous;
previous = i;
}
}
}

return minimumDistance;
}
```

• Reply • Share ›

gauravkohli • 3 years ago

```
#include
void main()
{
int arr[6]={1,2,3,4,4,1},i,pos1=0,pos4=0,diff=6,d=6;
for(i=0;i<6;i++)
printf("array is %d \n",arr[i]);
for(i=0;ipos4 && pos1!=0 && pos4!=0)
d=pos1-pos4 ;
else
if(pos4>1 && pos4!=0 && pos1!=0)
d=pos4-pos1;
if(d<diff)
diff=d;
}
printf("\ndiffrence is : %d\n",diff);
getch();
}
[sourcecode language="c"]
```

gauravkohli gauravkohli • 3 years ago

```
#include
void main()
{
int arr[6]={1,2,3,4,4,1},i,pos1=0,pos4=0,diff=6,d=6;
for(i=0;i<6;i++)
printf("array is %d \n",arr[i]);
for(i=0;ipos4 && pos1!=0 && pos4!=0)
d=pos1-pos4 ;
else
if(pos4>1 && pos4!=0 && pos1!=0)
d=pos4-pos1;
if(d<diff)
diff=d;
}
printf("\ndiffrence is : %d\n",diff);
getch();
}
```


gauravkohli gauravkohli • 3 years ago

```
#include
void main()
{
int arr[6]={1,2,3,4,4,1},i,pos1=0,pos4=0,diff=6,d=6;
for(i=0;i<6;i++)
printf("array is %d \n",arr[i]);
for(i=0;ipos4 && pos1!=0 && pos4!=0)
d=pos1-pos4 ;
else
```

```
if(pos4>1 && pos4!=0 && pos1!=0)
d=pos4-pos1;
if(d<diff)
diff=d;
}
printf("\ndiffrence is : %d\n",diff);
getch();
}
<del></del><del>
```


Nitesh • 3 years ago

```
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;
}
}
```

> `i++;`

Bala • 3 years ago

1) scan the array to store the first occurrences of both the elements
2) once both elements are found, take the absolute diff of indices and store the
3) continue the same for rest of the array

/* Code snippet */

```
int x_index = -1;
int y_index = -1;
int min_dis = INT_MAX;

for (i = 0; i < N; i++) {

if (x == arr[i]) {
x_index = i;
} else if (y == arr[i]) {
y_index = i;
}
```

see more