Day-20

**Problem1**- [**424. Longest Repeating Character Replacement**](https://leetcode.com/problems/longest-repeating-character-replacement/)

Solution-

class Solution {

public:

int characterReplacement(string s, int k) {

int n=s.size();

map<int ,int>m;

int i=0;

int j=0;

int maxi=0;

int ans=0;

while(j<n)

{

m[s[j]]++;

maxi=max(maxi,m[s[j]]);

int len=j-i+1;

if(len-maxi>k)

{

m[s[i]]--;

i++;

}

ans=max(ans,j-i+1);

j++;

}

return ans;

}

};

**Problem2**- **Check if reversing a sub array make the array sorted**

Solution-

#include <bits/stdc++.h>

using namespace std;

bool sortArr(int a[], int n)

{

int x = -1;

int y = -1;

for (int i = 0; i < n - 1; i++) {

if (a[i] > a[i + 1]) {

if (x == -1) {

x = i;

}

y = i + 1;

}

}

if (x != -1) {

reverse(a + x, a + y + 1);

for (int i = 0; i < n - 1; i++) {

if (a[i] > a[i + 1]) {

return false;

return 0;

}

}

}

return true;

}

// Driver Program

int main()

{

int arr[] = { 1, 2, 5, 4, 3 };

int n = sizeof(arr) / sizeof(arr[0]);

sortArr(arr, n) ? (cout << "Yes" << endl)

: (cout << "No" << endl);

return 0;

}

**Problem 3**- **Radix Sort – Data Structures and Algorithms Tutorials**

// C++ implementation of Radix Sort

#include <iostream>

using namespace std;

// A utility function to get maximum

// value in arr[]

int getMax(int arr[], int n)

{

int mx = arr[0];

for (int i = 1; i < n; i++)

if (arr[i] > mx)

mx = arr[i];

return mx;

}

// A function to do counting sort of arr[]

// according to the digit

// represented by exp.

void countSort(int arr[], int n, int exp)

{

// Output array

int output[n];

int i, count[10] = { 0 };

// Store count of occurrences

// in count[]

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

count[(arr[i] / exp) % 10]++;

// Change count[i] so that count[i]

// now contains actual position

// of this digit in output[]

for (i = 1; i < 10; i++)

count[i] += count[i - 1];

// Build the output array

for (i = n - 1; i >= 0; i--) {

output[count[(arr[i] / exp) % 10] - 1] = arr[i];

count[(arr[i] / exp) % 10]--;

}

// Copy the output array to arr[],

// so that arr[] now contains sorted

// numbers according to current digit

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

arr[i] = output[i];

}

// The main function to that sorts arr[]

// of size n using Radix Sort

void radixsort(int arr[], int n)

{

// Find the maximum number to

// know number of digits

int m = getMax(arr, n);

// Do counting sort for every digit.

// Note that instead of passing digit

// number, exp is passed. exp is 10^i

// where i is current digit number

for (int exp = 1; m / exp > 0; exp \*= 10)

countSort(arr, n, exp);

}

// A utility function to print an array

void print(int arr[], int n)

{

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

cout << arr[i] << " ";

}

// Driver Code

int main()

{

int arr[] = { 170, 45, 75, 90, 802, 24, 2, 66 };

int n = sizeof(arr) / sizeof(arr[0]);

// Function Call

radixsort(arr, n);

print(arr, n);

return 0;

}

**Problem4**- **Make all array elements equal with minimum cost**

#include <bits/stdc++.h>

using namespace std;

// This function assumes that a[] is

// sorted. If a[] is not sorted, we need

// to sort it first.

int minCostToMakeElementEqual(int a[], int n)

{

// If there are odd elements, we choose

// middle element

int y;

if (n % 2 == 1)

y = a[n / 2];

// If there are even elements, then we choose

// the average of middle two.

else

y = (a[n / 2] + a[(n - 2) / 2]) / 2;

// After deciding the final value, find the

// result.

int s = 0;

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

s += abs(a[i] - y);

return s;

}

// Driver code

int main()

{

int a[] = { 1, 100, 101 };

int n = sizeof(a) / sizeof(a[0]);

sort(a, a + n);

cout << (minCostToMakeElementEqual(a, n));

}

// This code is contributed by chitranayal