Q. Count distinct element in every window

Given an array of integers and a number k. Find count of distinct element in every window of size k.

Code -

```
vector<int> count_distinct(int a[], int n, int k){
    vector<int> v;
    unordered_map<int,int> m;
    for(int i=0;i<k;i++)
        m[a[i]]++;

    v.push_back(m.size());

    for(int i=k; i<n;i++){
        m[a[i-k]]--;
// decreasing frequency of those no, who
// are out of window and frequency is
// zero removing them.
        if(m[a[i-k]]==0)
            m.erase(a[i-k]);

        v.push_back(m.size());
    }
```

return V;
}

3. Sorting elements of array by frequency -

make a hash map to store frequency of elements in array.

make a vector of pair and store unodered map in vectors.

Sort the vector according to compairsion function (if frequely is higher put that first, if frequency is same put smaller first)

After that print the vectors.

bool myCmp(pair<int, int> a, pair<int, int> b){
    if(a.second == b.second){
        return a.first < b.first;
    else
        return a.second > b.second;
}

```
vector<int> sortbyfreq(int arr[], int n){

    ~~vector<pair<int,int>> v;~~
    unordered_map<int, int> m;
    vector<int> out;
    for(int i=0; i<n; i++)
        m[arr[i]]++;

    vector<pair<int,int>> v(m.begin(), m.end());

    sort(v.begin(), v.end(), mycmp);

    for(auto x: v){
        while(x.second!=0){
            out.push_back(x.first);
            x.second--;
        }
    }
    return out;
}
```

sorted -

Q. Median in a row wise matrix. -

We have given an matrix of size (r x c) which is row wise sorted.

we have to find median of all these element.

For the simplicity sake here no of total elements in matrix is odd. so that there will be only one media

median in this case will be that element, before which (r x c)/2 elements are present and after which (r x c)/2 elements are present.

So we have to find (r x c + 1)/2 th element in the matrix.

approach -

- we first find minimum and maximu element of matrix
for minimum element search first element of every row matrix and for maximum elements search last element of every row matrix

- Find median position as
  int median_pos = (r*c+1)/2;

- now run a loop until min<max;

  - int mid = mint + (max-min)/2;

now make a variable as cur_pos which store sum of position before mid element in each row matrix.

- int cur_pos = 0

- now run a loop over each row matrix and calculate no of elements in each row matrix less than or equal to mid, using upper_bound.

```
for(int i=0; i<r; i++){
    auto x = upper_bound(mat[i].begin(), mat[i].end(),
                         mid);
    int a = x - mat[i].begin();
    pos += a;
}
```

median

now if(cw_pos < [redacted]_pos)
min = mid+1;

else
max = mid;

after while loop

return min.

Q. Pattern searching in String.

- We have given a String and a substring and we have to print all index at which it appears.

ex-

Str = "0geeks for 10geeks"

Substring = "geeks"

o/p 0 10

approach - Initially we will use find to search substring in substring and again search for it in string but after the index first occurance is found.

```
void pattern(string str, string word){
    int pos = str.find(word);
    while(pos != string::npos){
        cout<< pos;
        pos = str.find(word, pos+1);
    }
}
```

- We have given a string storing floating point numbers. We have to print all digits after point/decimal.

approach - we find decimal index if present return substring after decimal not present return empty string.

```
string pattern(string num){
    int index = num.find('.');
    if(index == string::npos)
        return "";
    else
        return num.substr(index+1);
}
```

Q. We have given two strings. first one is of size 'n' and second one is of size 'n+1'. second string contains all characters of first string but one extra string. We have to print that character.

→ approach 1 - O(n log n)

We sort both string, and match all char. of both string one by one and the char which don't mach we print second string char at that index. and if all char of string one matched with string 2. we print last char of string 2.

```
void findextra(string s1, string s2){

    sort(s1.begin(), s1.end());
    sort(s2.begin(), s2.end());
    int n = s1.length();
    for(int i=0; i<n; i++){
        if(s1[i] != s2[i])
            cout<<s2[i]<<"\n";
            break;
    }
}
```

cout << s2[i];
}

approach 2 :- O(n)

We make an unordered_map<char, int>; we count occurance of each character in string 2. we run another loop on string 1 and decrease occurance of character in string 1. At last occurance of all character will be zero except that extra character.

```
char findextra(string s1, string s2){
    unordered_map<char, int> m;
    for(int i=0; i<s2.length(); i++){
        m[s2[i]]++;
    }
    for(int i=0; i<s1.length(); i++){
        m[s1[i]]--;
    }
    for(auto x: m){
        if(x.second == 1)
            return x.first;
    }
}
```

Q. String pangram checking.

A string is called pangram if it contains all 26 alphabet of english, lower or upper ~~don't~~ matter. They can be more characters along side with these 26 characters in string.

Approach 1 - O(n)

We will make a boolean array of size `26` and assign it false. now run a loop over string and for every alphabet (lower or upper) change respecting index to true. (for 'a' or 'A' index is '0' and for 'z' or 'Z' index is `25`)

now again run a loop over this array and check if any value is false, if found return false if not found any false value return true.

```
bool pangram(String s){
    bool arr[26]={0};
}
```

```
for(int i=0; i<s.length(); i++){
    char x = s[i];
    if(x<='z' && x>='a')
        arr[x-a] = true;
    else if(x<='Z' && x>='A')
        arr[x-A] = true;
}

for(int i=0; i<26; i++)
    if(arr[i] == false)
        return false;
return true;
}
```

approach 2 - O(n)

we use hash_map to store occurance of each character in hash_map. Then we run a loop i=0 to i<26, and check if m[a+i] or m[A+i] is present in the map or not. If not return false. If yes return true.

```
bool pangram(String s){
    unordered_map<char, int> m;
    for(auto x: s)
        m[x]++;
    for(int i=0; i<26; i++){
        if(m[a+i]==0 && m[A+i]==0)
            return false;
    }
```

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

return true;
}

Q. Check whether given two strings are anagram or not.

For 2 string to be anagram both must have same characters as other, but order may be different.

approach 1 - O(n log n)

we first match length of both string if they are different return false, if not then sort them.

If both are anagram after sorting both becomes same. now, match each character one by one, if differs returns false if not return true.

approach 2 - O(n) -

Check for length of both string if same then make a hashmap and count frequency of each char of one string and for other string run another loop and decrease frequency of each

char from that string and the check if any frequency is not zero. If found return false if not found return true.

→ __builtin_popcount() in C++ -

--builtin_popcount() function takes unsigned integer as input and count no of set bits in its ~~sign~~ binary representation.

ex-

n=5 (101) o/p-2

n=8 (1000) o/p=1

for (-ve) integers, first they converted in unsigned int (2's complement) then function applies.

there are 2 more versions of this function

--builtin_popcountl() for long unsigned int

--builtin_popcountll() for long long unsigned int.