Todays Content

1. Overriding Comperatr
2. Given ar[N] sort elements based on freq
3. largest number in array
4. k closest Points in Origin
5. Sort in Decreasing Order

Idea: When we want our own sorting order, we use concept of compare function

Compare function:
If we define order for 2 elements, using that we can fin order for entire array.

#Syntax:

```
bool f_name(Type s1, Type s2){
    if we want s1 before s2: return True
     else if s2 before s1: return False;
}

int arr[];
sort(arr, arr+n, f_name); #Sort arr based on f_name

vector<int> v;
sort(v.begin(), v.end(), f_name); #Sort v based on f_name
```

TC: O(Nlog N * Time taken for cmp function);

Q: Given ar[N] elements sort elements in decreasing Order.

Ex: ar[] : { 6 4 3 2 10 14 12 }

```
bool dec(int s1, int s2){
    if(s1 > s2){ # s1 come first
    }   return true;
    else{
        return false;
    }
}
```

return $s_1 > s_2$;

```
vector<int> sortDec(vector<int> & ar){

    sort(ar.begin(), ar.end(), dec);
    return ar;
}
```

Given ar[N] elements sort elements in Increasing order of frequency.
Note: If 2 elements have same freq, smaller element should come first

Ex:

```
          0  1  2  3  4  5  6  7
ar[] : {  7  2  9  2  3  3  2  5 }

ar[] = {  5  7  9  3  3  2  2  2 }
```

```
unordered_map<int, int> um;

bool freq(int s1, int s2){
    # We need to get freq of s1, s2
    if( um[s1] < um[s2] ){ return true;}          # s1 comes first;
    else if( um[s1] == um[s2] ){
        if( s1 < s2 ){ return true; # s1 comes first; }
        else { return false; # s2 comes first
    }

    else{ # um[s1] > um[s2]
    }  return false;                              # s2 comes first
}

vector<int> sortFreq( vector<int> & ar){

    for(int i = 0; i < ar.size(); i++){
        um[ar[i]]++;
    }
                                          Add Inbuilt
                                          Notes for sort in C++;
    sort( ar.begin(), ar.end(), freq)
}
```

38 Given vector of pairs, each pair is representing 2D point.
Sort points based on their distance to origin. #Imp

Note: If 2 points have same distance point with smaller x should come first

Note2: Distance between 2 points $(x_1\ y_1)$ & $(x_2\ y_2) = \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$

Distance between origin $(0,0)$ & $(x\ y) = \sqrt{(x_2-0)^2 + (y_2-0)^2} = \sqrt{x^2+y^2}$

Instead of comparing d, compare d^2
square won't effect comparison for the order

Ex: V: x y distance d^2:

	V (x, y)	distance	d^2
0	<2, 3>	$\sqrt{13}$	~~13~~
1	<1, 4>	$\sqrt{17}$	~~17~~
2	<5, 2>	$\sqrt{29}$	29
3	<3, 3>	$\sqrt{18}$	~~18~~
4	<4, -2>	$\sqrt{20}$	~~20~~
5	<3, 2>	$\sqrt{13}$	~~13~~

	x y
0	<2, 3>
1	<3, 2>
2	<1, 4>
3	<3, 3>
4	<4, -2>
5	<5, 2>

```
bool dist( pair<int,int> s1, pair<int,int> s2){

    int d1 = s1.first * s1.first + s1.second * s1.second   # x1^2 + y1^2
    int d2 = s2.first * s2.first + s2.second * s2.second   # x2^2 + y2^2
    if( d1 < d2) { return true; # s1 comes first}
    else if( d1 == d2){                        ↗ if s1.first < s2.first = True
        return s1.first < s2.first;   #
    }                                          ↘ else: return False;
    else{ # d1 > d2 : s2 comes first
        return false;
    }
}

vector<pair<int,int>> sortDist (vector<pair<int,int>> & arr){

    sort(arr.begin(), arr.end(), dist);
    return arr;
}
```

48 Largest Number:

Given an ar[N], arrange them in such a way that by concenating all of them from left to right it should from largest number.

Note: Result may be very large, so return a string.

Ex1: ar[] = {2 3 9 0} (indices 0 1 2 3)

$\rightarrow$ {2 3 9 0} = 2390

$\rightarrow$ {9 3 2 0} = 9320 ✓

Ex2: ar[] = {99 90 98} $\longrightarrow$ {99 98 90} = 999890

Ex3: ar[] = {998 9} $\longrightarrow$ {9 998} = 9998

Ex4: ar[] = {30 3} $\longrightarrow$ {3 30} = 330

Idea

1. Sort arr[] In decreasing & concatenate *

2. Hint: When ever we need order for arr[], fin order for 2 elements

Take 2 elements:

ele1	ele2	ele1+ele2		ele2+ele1	order:
89	8	898	>	889	ele1 comes first
90	9	909	<	990	ele2 comes first
98	9	989	<	998	ele2 comes first

Note: When we compare strings we get dictionary order

```
                                   #S1: 6 5 4    cout << S2 > S1; #true;
bool desc(int s1, int s2){         #S2: 6 9 4
    string c1 = to_string(s1) + to_string(s2)
    string c2 = to_string(s2) + to_string(s1)
    if(c1 > c2){ return true; }
    else{ return false; }
}

string sortDist(vector<int> & arr){
    sort(arr.begin(), arr.end(), desc);
    string ans = "";
    for(int i=0; i< arr.size(); i++){
        ans = ans + to_string(arr[i]);
    }
    return ans;
}
```