Todays Content.

1. CountSort
2. Max triplet prod
3. Pair<>;
4. Vector<pair<>>;

CountSort:

Given arr[N] all ele in range [2-6] sort arr[] in Inc.

	0	1	2	3	4	5	6	7	8	9
arr[10] = {	3	2	4	6	4	2	3	4	3	6 }

arr[ ] = { 2 2 3 3 3 4 4 4 6 6 }

Ideal: Apply BS/Is/IS = $O(N^2)$
MS/ Inbuilt = $O(N \log N)$

Idea: Store freq in hashmap & using that sort arr[]

HashMap: Issue: Order not maintained in hashmap

2 : 2 Resolve: Iterate from [2..6]

6 : 2 for every ele get freq from hm.

4 : 3 Insert ele in arr[] those many times.

3 : 3

Dry Run:

i: [2..6] j iterat

HashMap:	i: [2..6]	: Freq:	j iterat
2 : 2	i: 2	: Freq: 2	2 times
6 : 2	i: 3	: Freq: 3	3 times
4 : 3	i: 4	: Freq: 3	3 times
3 : 3	i: 5	: Freq: 0	0 times
	i: 6	: Freq: 2	2 time

	0	1	2	3	4	5	6	7	8	9
	~~3~~	~~2~~	~~4~~	~~6~~	~~4~~	~~2~~	~~3~~	~~4~~	~~3~~	~~6~~
	2	2	3	3	3	4	4	4	6	6

j → j → j → j → j → j → j → j → j → j → j

Total iterations : N+5 = O(N)

Total outer loop = 5 iter

Total inner loop = N iter

#Con: We use freqny of each element to sort

FreqnySort or CountSort

Note: j initialized once at start.

```
vector<int> sort(vector<int> &A){
    #Step1:
    unordered_map<int, int> um;
    for(int i=0; i<A.size(); i++){
        um[A[i]]++;
    }

    #Step2:
    int j=0;
    for(int i=2; i<=6; i++){
        if( um.find(i) != um.end() ){ # i exist
            int f = um[i]; # Element i insert f times in arr[]
            while( f>0 ){
                A[j] = i;
                j++; # goto next index
                f--;
            }
        }
    }

    return A;
}
```

Given an arr[] where all ele in Range [a..b] sort arr[].

```
vector<int> sort(vector<int> &A) {   TC: O(N + N+R) = O(N+R)   -> {b-a+1}
                                     SC: O(N) # Elements in hashmap.
  #Step1:
  unordered_map<int, int> um;        } O(N)
  for(int i=0; i<A.size(); i++){
      um[A[i]]++;
  }

                                     Total = Outerloop   Innerloop
                                             [a..b] = b-a+1   [0..N-1] = N
  #Step2:                            Total = O(b-a+1 + N)  # R Range = b-a+1
    int j=0;                         Total = O( R + N)
    for(int i=a; i<=b; i++){
        if( um.find(i) != um.end()){ # i exist
            int f = um[i]; # Element i insert f times in arr[]
            while(f>0){
                A[j] = i;
                j++; # goto next index
                f--;
            }
        }
    }

    return A;
}
```

```
vector<int> sort(vector<int> &A) {   TC: O(N + N+R) = O(N+R)   → {b-a+1}
                                     SC: O(N) # Elements in hashmap.
  #Step1:
  unordered_map<int, int> um;
  for(int i=0; i<A.size(); i++){
      um[A[i]]++;
  }

  #Step1.5
    Iterate & Calculate min = a
    Iterate & Calculate max = b

  #Step2:
    int j=0;
    for(int i=a; i<=b; i++){
        if( um.find(i) != um.end()){ # i exist
            int f = um[i]. # Element i insert f times in arr[]
            while(f>0){
                A[j] = i;
                j++; # goto next index
                f--;
            }
        }
    }

    return A;
}
```

When to apply CountSort:

Sort ar[N] elements & assume max(ar) - min(ar) + 1 = R

	MergeSort : O(NlogN)	CountSrr O(N+R)
if (R ≈ N)	O(NlogN)	O(N+N) = O(N) ✓
if (R ≈ NlogN)	O(NlogN) * 1 ✓	O(N+NlogN) : O(NlogN) * O(1)
if (R > NlogN)	O(NlogN) ✓	O(N+>NlogN) = > O(NlogN)

Con: Use CountSort only if

Range of arr() ≈ size of arr()

- ↳ Max - Min ✓
- ↳ Constraints ✓

Q Given ar[N] elements calculate min Triplet product

Constraints:

$1 <= N <= 10^6$

$1 <= ar[i] <= 10^9$

	0	1	2	3	4	5	6	
Eg: ar[] = {	3	6	7	4	2	9	8 }	ans = 2*3*4 = 24.

Ideal: Sort & multiply first 3 elements

TC: $O(N \log N + 1) = O(N \log N)$

Idea2: Apply selection sort 3 times, we will get 3 smallest elements at start & multiply first 3 elements

TC: $O(3N) = O(N)$ SC: O(1)

```
for(int i=0; i<3; i++) { # i = 0 1 2
    #Calculate mini
    int mini = i; # iterate from [i.. N-1]
    for( int j=i; j<N; j++){
        if [ ar[j] <. ar[mini]){
            mini = j;
        }
    }
    swap ar[mini] & ar[i]
}
return ar[0] * ar[1] * ar[2]
```

Q: Given ar[N] & M : Mostly skip it

Calculate length of longest subset, without a pair (i, j) such that (ar[i] + ar[j]) % M = 0.

Pair: When we want to combine more than 1 value into single variable

#library:

```
#include <utility>
```

#Declaration:

```
pair<Type1, Type2> p1;
    pair<int, float> p1;    p1 <0, 0.0>

pair<Type1, Type2> p2(v1, v2);
    pair<int, string> p2(23, "Sagar");    p2 <23, "Sagar">

pair<Type1, Type2> p3 = make_pair(v1, v2);
    pair<int, int> p3 = make_pair(10, 20);
                        <10, 20>

pair<Type1, Type2> p4 = {v1, v2}
    pair<int, int> p4 = {10, 20}
                        <10, 20>
```

Note: To create a pair:

```
make_pair(v1, v2);
{v1, v2}
```

#Auss:

p {v_1, v_2} v_1 = P.first v_2 = p.second

pair<int, int> p = {10, 20} # p = <10, ~~20~~ 30>

print(p.first); #10

p.second = p.second+10; # 20+10 = 30

Comparism.

When we compare 2 pairs, program follow below steps

Step1: 1st element in both are compared first

Step2: If 1st element is same 2nd element in both are compared.

pair<type1, type2> p1 (10, 20);

pair<type1, type2> p2 (20, 10);

pair<type1, type2> p3 (20, 30);

print(p1 == p2) #false

print(p2 > p1) #true

print(p1 > p3) #false

print(p3 > p2) #true;

pair<Type1, Type2> p

pair<pair<int, string>, pair<float, char>> p;

p.first.first p.first.second

p.second.first

Pair in vector:

vector<type> v; → type can be int/long/string/<Pair>/Object/...

fn:

```
vector<pair<int,int>> v;
v.push_back({10,20})
v.push_back({5,20})
v.push_back({10,5});
v.push_back({5,10});
```

v

0	{10,20}
1	{5,20}
2	{10,5}
3	{5,10}

Iteration in vector:

#Ways:

```
for(int i=0; i<v.size(); i++){
    #v[i] is a pair.
    print(v[i].first);
    print(v[i].second);
}
```

3

v

0	{10,20}
1	{5,20}
2	{10,5}
3	{5,10}

Sort vector of pairs:

```
sort(v.begin(), v.end()); # Sort of the order.
#Sort pairs in inc order of 1st element
if 2 elements have same 1st element:
        sort on 2nd element
```

v

0	{5,10}
1	{5,20}
2	{10,5}
3	{10,20}

#way2:

```
for(auto it : vec){
    print(it.first, it.second)
}
```

auto it :

<1, 2>
<3, 4>
<10, 20>
<5, 6>

#way3

```
for(auto &it : vec){
    print(it.first, it.second)
}
```

auto it :

<1, 2>
<3, 4>
<10, 20>
<5, 6>