SECOND APPROACH USING MAPS

🔷 Problem Statement (same hi hai)

Hume array ke elements ko unke **sorted order ke rank** (0-based) me convert karna hai.

- Smallest → `0`
- Next smallest → `1`
- ... and so on.

🔷 Code ka Step by Step Logic

```
int arr[] = {19,12,23,8,16};
int n = sizeof(arr) / sizeof(arr[0]);
```

➡️ Array banaya aur size nikala.

```
unordered_map<int,int> mp;
vector<int> v;
for(int i=0;i<n;i++){
    v.push_back(arr[i]);
}
```

➡️ Ek vector `v` banaya jisme array ke sab elements copy kar diye.

➡️ `mp` ek map hai jo **element → uska rank** store karega.

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

➡️ Vector ko sort kar diya → ab elements ascending order me aa gaye.

➡️ `v` = `{8, 12, 16, 19, 23}`

```
for(int i=0;i<v.size();i++){
    mp[v[i]] = i;
}
```

➡️ Har element ko uske **rank (index in sorted array)** ke saath map kar diya.

- `mp[8] = 0`
- `mp[12] = 1`
- `mp[16] = 2`
- `mp[19] = 3`
- `mp[23] = 4`

```
for(int i=0;i<n;i++){
    arr[i] = mp[arr[i]];
}
```

➡️ Ab original array ke elements ko unke rank se replace kar diya.

- `arr[0] = mp[19] = 3`
- `arr[1] = mp[12] = 1`
- `arr[2] = mp[23] = 4`
- `arr[3] = mp[8] = 0`
- `arr[4] = mp[16] = 2`

➡️ Final `arr = {3, 1, 4, 0, 2}`

```
for(int i=0;i<n;i++){
    cout<<arr[i]<<" ";
}
```

➡️ Output print ho jaayega.

🔷 Dry Run

Initial:

```
arr = {19,12,23,8,16}
v   = {}
```

Step 1: Copy to vector

```
v = {19,12,23,8,16}
```

Step 2: Sort vector

```
v = {8,12,16,19,23}
```

Step 3: Fill map (element → rank)

```
mp[8]  = 0
mp[12] = 1
mp[16] = 2
mp[19] = 3
mp[23] = 4
```

Step 4: Replace arr elements

```
arr[0] = mp[19] = 3
arr[1] = mp[12] = 1
arr[2] = mp[23] = 4
arr[3] = mp[8]  = 0
arr[4] = mp[16] = 2
```

Final:

```
arr = {3, 1, 4, 0, 2}
```

🔷 Key Difference from First Approach

- **Pehle wala**: Har baar minimum nikalne ke liye loop chal raha tha → `O(n^2)` time.
- **Ye wala**: Sirf ek sort (`O(n log n)`) aur map fill → jyada **efficient** hai.

👉 Output dono approach me same hai:

```
3 1 4 0 2
```

```
arr = {19,12,23,8,16}
```

🔷 Step 1: Copy arr → v

Index	arr[i]	v (after push_back)
0	19	{19}
1	12	{19,12}
2	23	{19,12,23}
3	8	{19,12,23,8}
4	16	{19,12,23,8,16}

Final:

```
v = {19,12,23,8,16}
```

🔷 Step 2: Sort v

```
Before sort: {19,12,23,8,16}
After sort : {8,12,16,19,23}
```

🔷 Step 3: Fill map (mp[v[i]] = i)

i (index in v)	v[i]	mp[v[i]]
0	8	0
1	12	1
2	16	2
3	19	3
4	23	4

Final map:

```
mp = { 8→0, 12→1, 16→2, 19→3, 23→4 }
```

🔷 Step 4: Replace arr[i] with mp[arr[i]]

i	Old arr[i]	mp[arr[i]]	New arr[i]
0	19	3	3
1	12	1	1
2	23	4	4
3	8	0	0
4	16	2	2

🔷 Final arr

```
{3, 1, 4, 0, 2}
```

⚡ Difference clear hai bhai:

- Pehle wale approach me har baar minimum dhoondhna pada ($O(n^2)$).
- Ye wala sort + map approach direct rank assign karta hai ($O(n \log n)$).