Subarray with 0 sum

```
bool subArrayExists(int arr[], int n)
   {  int sum=0;
      unordered_map<int,bool> m;
      for(int i=0;i<n;i++){
         sum+=arr[i];

         if(sum==0)
         return true;

         if(m[sum]==true)
         return true;

         m[sum]=true;
      }
      return false;
   }
```

Three Way Partition

```
void threeWayPartition(vector<int>& array,int a, int b)
   {
      int low = 0, mid = 0, high = array.size()-1;

       while(mid <= high){
          if(array[mid] < a)
             swap(array[low++], array[mid++]);

          else if(array[mid] >b)
             swap(array[mid], array[high--]);

          else
             mid++;
   }
   }
```

Chocolate Distribution Problem

```
long long findMinDiff(vector<long long> a, long long n, long long m){
        long long mindiff = INT_MAX;
        if (m == 0 || n == 0)
        return 0;
        sort(a.begin(), a.end());
        if (n < m)
        return -1;

        for (int i = 0; i + m - 1 < n; i++) {
        long long diff = a[i + m - 1] - a[i];
        if (diff < mindiff)
          mindiff = diff;
        }
        return mindiff;
    }
```

Merge Intervals

```
vector<vector<int>> merge(vector<vector<int>>& intervals) {
      int n = intervals.size();
       sort(intervals.begin(),intervals.end());
       vector<vector<int>> ans;
       for(int i = 0;i<n;i++){
           if(ans.empty() || intervals[i][0] > ans.back()[1]){
               ans.push_back(intervals[i]);
           }
           else{
               ans.back()[1] = max(ans.back()[1],intervals[i][1]);
           }
       }
       return ans;
   }
```