Aggressive Cows

Approach :

Same as allocation of pages. Here we need to maximize distance so we find whether a distance in range 1 to arr[n-1]-arr[0] is possible. Initially we sort the element and then define this search space. So basically the idea is to search between low and high whether the cows can be placed keeping the mid as the dist between the cows. If we can place all cows then it is part of answer and if not then we need to reduce the distance as in Allocate minimum number of pages. We keep moving the cow to new position once a distance exceeds the difference of initial pos of cow and current position. Also a new cow is introduces at this position.

Code :

```
#include <bits/stdc++.h>
bool isPossible(vector<int> positions,int n,int cows,int dist)
{
  int c_pos=positions[0],cnt=1;
  for(int i=1;i<n;i++)
  {
    if(positions[i]-c_pos>=dist)
    {
      cnt++;
      c_pos=positions[i];
    }
    if(cnt==cows)
      return true;
  }

  return false;
}
int chessTournament(vector<int> positions , int n ,  int c){
  // Write your code here
  sort(positions.begin(),positions.end());
  int low=1,high=positions[n-1]-positions[0],res;
  while(low<=high)
  {
    int mid=low+(high-low)/2;
    if(isPossible(positions,n,c,mid))
    {
      res=mid;
      low=mid+1;
    }
    else{
```

```
      high=mid-1;
    }
  }
  return res;
}
```

- Time Complexity : O(n*log(m))
- Space Complexity : O(1)