624. Maximum Distance in Arrays

You are given `m` `arrays`, where each array is sorted in **ascending order**.

You can pick up two integers from two different arrays (each array picks one) and calculate the distance. We define the distance between two integers `a` and `b` to be their absolute difference `|a - b|`.

Return *the maximum distance*.

Example 1:

```
Input: arrays = [[1,2,3],[4,5],[1,2,3]]
Output: 4
Explanation: One way to reach the maximum distance 4 is to pick 1 in the first or third array and pick 5 in the second array.
```

Example 2:

```
Input: arrays = [[1],[1]]
Output: 0
```

Constraints:

- `m == arrays.length`
- `2 <= m <= 10`5
- `1 <= arrays[i].length <= 500`
- `-10`4 `<= arrays[i][j] <= 10`4
- `arrays[i]` is sorted in **ascending order**.
- There will be at most `10`5 integers in all the arrays.

624. Maximum Distance in Arrays

```
/*
   Greedy: Brute force
   Time complexity: O(m^2) -TLE
   Space complexity: O(1)
*/
class Solution {
   public:
      int maxDistance(std::vector<std::vector<int>>& arrays) {
         int m=arrays.size();
         int ans=0;
         for(int i=0;i<m-1;++i){
            auto& Ai=arrays[i];
            for(int j=i+1;j<m;++j){
               auto& Aj=arrays[j];
               ans=std::max({ans,Ai.back()-Aj[0],Aj.back()-Ai[0]});
            }
         }

         return ans;
      }
};
```

624. Maximum Distance in Arrays

```
/*
   Greedy: Simulation
   Time complexity: O(6m)
   Space complexity: O(m)
*/
class Solution {
public:
   int maxDistance(vector<vector<int>>& arrays) {
      int m=arrays.size();
auto find_max=[&](std::vector<bool>& picked)->int{
      int mx=INT_MIN;
      int pos_max=-1;
      for(int i=0;i<m;++i){
         if(picked[i]) continue;
         auto& arr=arrays[i];
         int n=arr.size();
         if(mx<arr[n-1]) {
            mx=arr[n-1];
            if(pos_max!=-1) picked[pos_max]=false;
            pos_max=i;
            picked[i]=true;
         }
      }
      return mx;
   };
auto find_min=[&](std::vector<bool>& picked)->int{
      int mi=INT_MAX;
      int pos_min=-1;
      for(int i=0;i<m;++i){
         if(picked[i]) continue;
         auto& arr=arrays[i];
         if(mi>arr[0]) {
            mi=arr[0];
            if(pos_min!=-1) picked[pos_min]=false;
            pos_min=i;
            picked[i]=true;
         }
      }
      return mi;
   };
      std::vector<bool> picked(m,false);

      int mx1=find_max(picked);
      int mi1=find_min(picked);

      std::fill(picked.begin(),picked.end(),false);

      int mi2=find_min(picked);
      int mx2=find_max(picked);

      return std::max(mx1-mi1,mx2-mi2);
   }
};
```

```cpp
/*
   Greedy: Math Observation
   Time complexity: O(m)
   Space complexity: O(1)
*/
class Solution {
   public:
      int maxDistance(std::vector<std::vector<int>>& arrays) {
         int m=arrays.size();

         int mx=arrays[0].back(),mi=arrays[0][0];
         int ans=0;
         for(int i=1;i<m;++i){
            auto& arr=arrays[i];
            ans=std::max({ans,arr.back()-mi,mx-arr[0]});
            mi=std::min(mi,arr[0]);
            mx=std::max(mx,arr.back());
         }

         return ans;
      }
};
```