Max Minus Min Equals Size (Easy)

Problem category:	Greedy
Expected difficulty:	1000

Solution

First, note that the order of elements in a group is not important. Hence, we can sort the input to make it easier to work with.

Second, since in this version of the problem the elements are **distinct**, once the input is sorted we should notice that there is strict monotonicity of the difference between the maximum and minimum elements in any chosen subarray (i.e., extending it will increase the difference as well as the length; the converse is also true when shrinking it). Therefore, this problem lends itself to a solution using two pointers.

Let us keep two pointers to different positions in the sorted array. If the difference between the element of the right pointer and that of the left pointer is less than or equal to the number of elements between them (inclusive), then we can always form a group using a subset of those elements, such that the size of the set equals the difference; the only exception being a difference of one, in which case it is not possible.

Then, computing the answer is a matter of selecting the maximum difference which satisfies the condition above.

Complexity

Since we need to sort the input, the overall time complexity is $O(n \log n)$. Alternatively, since the value of a_i is bounded by n, one can use a boolean vector to keep track of values to be visited, resulting in linear time complexity.

Code

```
void solve() {
  int n; cin >> n;
  vector<int> a(n);
  for (int i = 0; i < n; i++) {
    cin >> a[i];
  }
  sort(a.begin(), a.end());
  int ans = 0;
  for (int i = 0, j = 1; j < n;) {
    auto diff = a[j] - a[i], len = j - i + 1;
    if (diff <= len && diff > 1) {
      ans = max(ans, diff);
    }
    diff > len ? i++ : j++;
  }
  cout << ans << endl;
}
```