2275. Largest Combination With Bitwise AND Greater Than Zero

The **bitwise AND** of an array `nums` is the bitwise AND of all integers in `nums`.

- For example, for `nums = [1, 5, 3]`, the bitwise AND is equal to `1 & 5 & 3 = 1`.
- Also, for `nums = [7]`, the bitwise AND is `7`.

You are given an array of positive integers `candidates`. Evaluate the **bitwise AND** of every **combination** of numbers of `candidates`. Each number in `candidates` may only be used **once** in each combination.

Return *the size of the* ***largest*** *combination of* `candidates` *with a bitwise AND* ***greater*** *than* `0`.

Example 1:

```
Input: candidates = [16,17,71,62,12,24,14]
Output: 4
Explanation: The combination [16,17,62,24] has a bitwise AND of 16 & 17 & 62 & 24 =
16 > 0.
The size of the combination is 4.
It can be shown that no combination with a size greater than 4 has a bitwise AND
greater than 0.
Note that more than one combination may have the largest size.
For example, the combination [62,12,24,14] has a bitwise AND of 62 & 12 & 24 & 14 =
8 > 0.
```

Example 2:

```
Input: candidates = [8,8]
Output: 2
Explanation: The largest combination [8,8] has a bitwise AND of 8 & 8 = 8 > 0.
The size of the combination is 2, so we return 2.
```

Constraints:

- `1 <= candidates.length <= 10^5`
- `1 <= candidates[i] <= 10^7`

2275. Largest Combination With Bitwise AND Greater Than Zero

```
/*
   Bit Manipulation
   Time complexity: O(n+n log2 mx)
   Space complexity: O(1)
   n: number of elements in `candidates` array
   mx: max element in `candidates` array
*/
class Solution {
   public:
      int largestCombination(std::vector<int>& candidates){
         int k=32-__builtin_clz(*std::max_element(candidates.begin(),candidates.end()))-1;
         int ans=0;
         for(int i=0;i<=k;++i){
            int cnt=0;
            for(auto& e: candidates) cnt+=int((e&(1<<i))!=0);
            ans=std::max(ans,cnt);
         }
         return ans;
      }
};
```

2275. Largest Combination With Bitwise AND Greater Than Zero

```
/*
   Bit Manipulation
   Time complexity: O(32n)=O(n)
   Space complexity: O(1)
   n: number of elements in `candidates` array
   mx: max element in `candidates` array
*/
class Solution {
public:
   int largestCombination(std::vector<int>& candidates){
      int ans=0;
      for(int i=0;i<32;++i){
         int cnt=0;
         for(auto& e: candidates) cnt+=int((e&(1<<i))!=0);
         ans=std::max(ans,cnt);
      }
      return ans;
   }
};
```