1. Maximum Gap

Given an unsorted array, find the maximum difference between the successive elements in its sorted form.

Return 0 if the array contains less than 2 elements.

**Example 1:**

Input: [3,6,9,1]  
Output: 3  
Explanation: The sorted form of the array is [1,3,6,9], either  
 (3,6) or (6,9) has the maximum difference 3.

**Example 2:**

Input: [10]  
Output: 0  
Explanation: The array contains less than 2 elements, therefore return 0.

**Note:**

* You may assume all elements in the array are non-negative integers and fit in the 32-bit signed integer range.
* Try to solve it in linear time/space.

**解法1** 排序

class Solution {  
public:  
 int maximumGap(vector<int>& nums) {  
 sort(nums.begin(), nums.end());  
 int res = 0;  
 for(int i = 1; i < nums.size(); ++i)res = max(res, nums[i]-nums[i-1]);  
 return res;  
 }  
};

**解法2** 排序算法：桶排序、计数、基数。max-min可能会比较大，计数排序可能需要很大的空间，实现后发现会超时。

**解法2.1** 基数排序

基数排序是按照低位先排序，然后收集；再按照高位排序，然后再收集；依次类推，直到最高位。有时候有些属性是有优先级顺序的，先按低优先级排序，再按高优先级排序。最后的次序就是高优先级高的在前，高优先级相同的低优先级高的在前。

class Solution {  
public:  
 int maximumGap(vector<int>& nums) {  
 if(nums.size() < 2)return 0;  
 radix\_sort(nums);  
 int res = 0;  
 for(int i = 1; i < nums.size(); ++i)res = max(res, nums[i]-nums[i-1]);  
 return res;  
 }  
 void radix\_sort(vector<int>& nums){  
 int maxVal = \*max\_element(nums.begin(), nums.end());  
 int exp = 1, radix = 10;  
 vector<int>aux(nums.size());  
 while(maxVal / exp > 0){  
 vector<int>cnt(radix, 0);  
 for(int i = 0; i < nums.size(); ++i){  
 int idx = (nums[i] / exp) % radix;  
 cnt[idx]++;  
 }  
 for(int i = 1; i < cnt.size(); ++i)cnt[i] += cnt[i-1];  
 for(int i = nums.size() - 1; i >= 0; --i){  
 aux[--cnt[(nums[i] / exp) % 10]] = nums[i];  
 }  
 for(int i = 0; i < nums.size(); ++i)nums[i] = aux[i];  
 exp \*= 10;  
 }  
 }  
};

**解法2.2** 桶排序

间隔为d，则。因此设置n-1个桶，桶高为，每个桶存储最小和最大的元素，最后依次比较**使用过的**相邻两个桶的间隔，取最大值

class Solution {  
public:  
 int maximumGap(vector<int>& nums) {  
 if(nums.size() < 2)return 0;  
 int n\_max = \*max\_element(nums.begin(), nums.end());  
 int n\_min = \*min\_element(nums.begin(), nums.end());  
  
 int b = max(1, int((n\_max - n\_min) / (nums.size() - 1)));  
 vector<vector<int>>cnt(int((n\_max - n\_min) / b) + 1, vector<int>(2));  
  
 for(auto &v : cnt){  
 v[0] = INT\_MAX;  
 v[1] = INT\_MIN;  
 }  
 for(int x : nums){  
 int idx = floor((x - n\_min) / b);  
 cnt[idx][0] = min(cnt[idx][0], x);  
 cnt[idx][1] = max(cnt[idx][1], x);  
 }  
 int res = 0;  
 int pre\_max = cnt[0][1];  
 for(int i = 1; i < cnt.size(); ++i){  
 if(cnt[i][0] == INT\_MAX)continue;  
 res = max(res, cnt[i][0] - pre\_max);  
 pre\_max = cnt[i][1];  
 }  
 return res;  
 }  
};