<https://leetcode.com/problems/sort-an-array/>

class Solution {

public:

vector<int> sortArray(vector<int>& nums) {

srand(0);

int logn = (log(nums.size()/log(2.0)));

// insertion\_sort\_bottomup(nums, 0, nums.size());

int minVal = \*min\_element(nums.begin(), nums.end());

int maxVal = \*max\_element(nums.begin(), nums.end());

if(maxVal - minVal + 1 <= 2 \* nums.size()) {

cout << "Distributed Counting Sorting is in action...\n";

distCountingSort(nums, minVal, maxVal); // O(n) worst-case time

} else if(quicksort10(nums, 0/\*low\*/, nums.size() - 1/\*high\*/,

0/\*depth\*/, logn)) {

cout << "Coarse Quicksort was successful, Insertion Sort is fine tuning...\n";

insertion\_sort\_bottomup(nums, 0, nums.size());

} else {

cout << "Coarse Quicksort FAILED, Mergesort is to the rescue...\n";

mergesort\_bottomup(nums, 0, nums.size());

}

return nums;

}