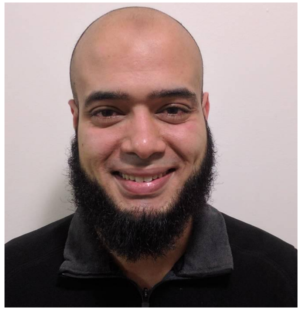
Algorithms
Sorting Homework 1

Mostafa S. Ibrahim
Teaching, Training and Coaching for more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



Problem #1: Insertion Sort v2

- Introduce these changes to the lecture code
- 1) Sort the data in descending order
- 2) Utilize the swap function to reduce the code length
  - Your new code should be only 3 lines
- Which one has fewer number of operations: this version vs the lecture?

Problem #2: Count Sort for Negative Values

- Develop a count sort version to handle the following requirements:
    - 1 <= Array length <= 50000
    - Values range: -50000 <= nums[i] <= 50000
- Input: nums = [-5, 2, -3, 1, 1234, -2453]
- Output: [-2453, -5, -3, 1, 2, 1234]
- Online judge: LeetCode 912 - Sort an Array
    - Refer to the **online judge section** to understand about Online Judges and how to use

Problem #3: Count Sort for a range

- Develop a count sort version to handle the following requirements:
    - Values range: -10^9 <= nums[i] <= 10^9
    - However: the max value - min value <= 500
- Input: nums = [10000107,10000035,10000001]
- Output: [10000001, 10000035, 10000107]
- Online judge (can be helpful): LeetCode 912 - Sort an Array

Problem #4: Count Sort for strings v1

- Implement void countSort(vector<string> &array)
- The function updates a vector of strings sorted using count sort
- Consider the following constraints
  - Every string consists only lower letters (a-z) and is of length >= 1
  - The sorting is only based on the first character of a string
  - The algorithm must be stable
- Input example: ziad, **belal**, adam, **baheir**, ali
- Output: adam, ali, belal, baheir, ziad
- Note: **belal** is equal to **baheir,** as sorting is only based on letter.
- We must maintain input order to be stable, so belal comes first

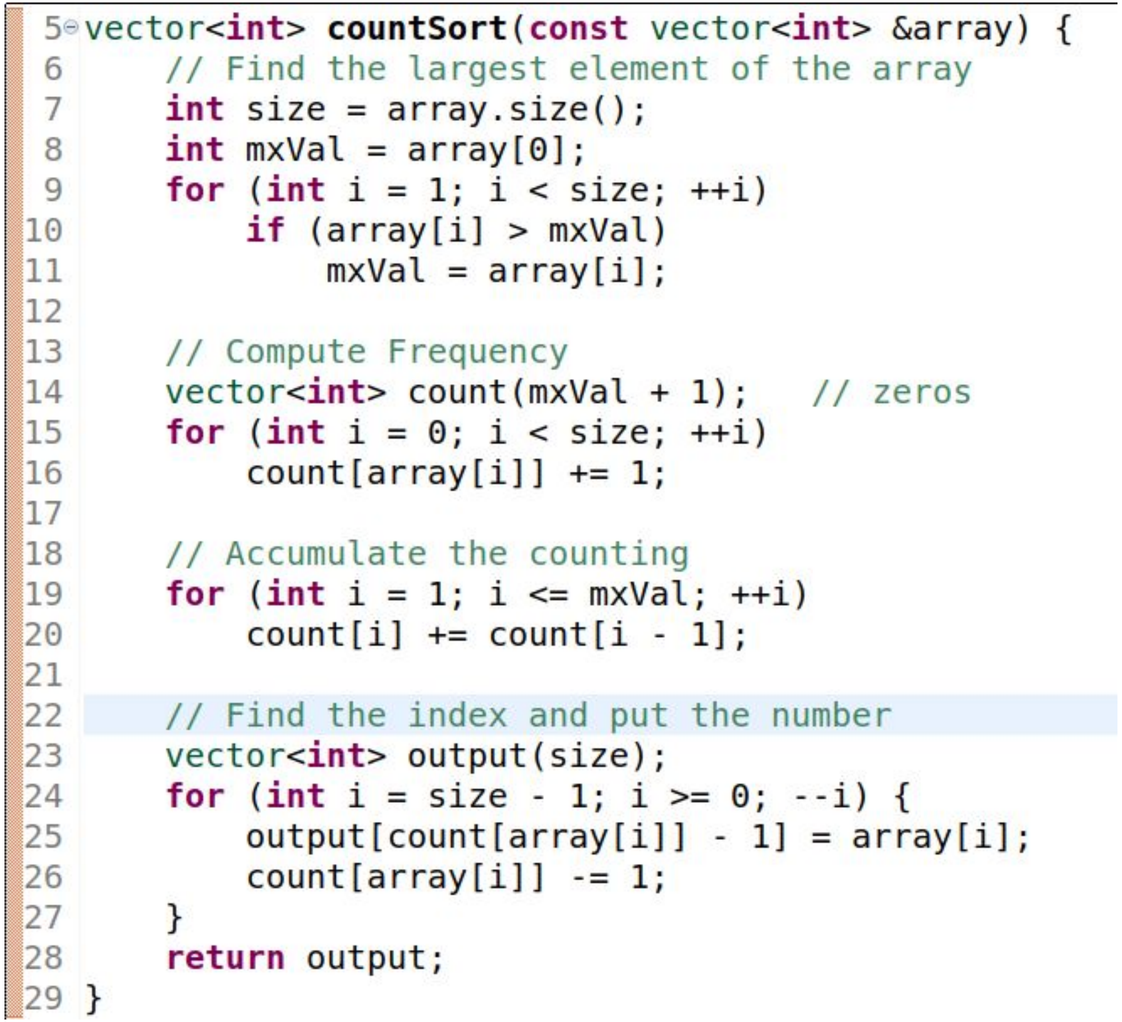
Problem #5: Count Sort for strings v2

- Implement void countSort(vector<string> &array)
- The function updates a vector of strings sorted using count sort
- Consider the following constraints:
  - Every string consists lower letters (a-z) and is of length >= 2
  - The sorting is only based on the **first two** characters of a string
  - The algorithm must be stable
- Input example: axz, axa, zzz, abc, abe
- Output: abc, abe, axz, axa, zzz
  - Prefix ab must come before ax. Within each group, respect the **input order**

Problem #6: Count Sort Version 2

- There is another popular implementation for the count sort
- Please study and understand the code
- Compare the implementation with the lecture implementation
  - What are the pros and cons?

- Observe: the first 2 blocks are the same as the code lecture
- The difference is in how to build the output



```
vector<int> countSort(const vector<int> &array) {
    // Find the largest element of the array
    int size = array.size();
    int mxVal = array[0];
    for (int i = 1; i < size; ++i)
        if (array[i] > mxVal)
            mxVal = array[i];

    // Compute Frequency
    vector<int> count(mxVal + 1);   // zeros
    for (int i = 0; i < size; ++i)
        count[array[i]] += 1;

    // Accumulate the counting
    for (int i = 1; i <= mxVal; ++i)
        count[i] += count[i - 1];

    // Find the index and put the number
    vector<int> output(size);
    for (int i = size - 1; i >= 0; --i) {
        output[count[array[i]] - 1] = array[i];
        count[array[i]] -= 1;
    }
    return output;
}
```

"Acquire knowledge and impart it to the people."

"Seek knowledge from the Cradle to the Grave."