Grade: 9/9

Exercise 2 (for grade) ~ Wednesday, September 14, 2022 ~ CPSC 535 Fall 2022

Write one submission for your entire group, and write all group members' names on that submission. Turn in your submission before the end of class. The ☒ symbol marks where you should write answers.

Recall that our recommended problem-solving process is:

1. **Understand** the problem definition. What is the input? What is the output?
2. **Baseline** algorithm for comparison
3. **Goal** setting: improve on the baseline how?
4. **Design** a more sophisticated algorithm
5. **Inspiration** (if necessary) from patterns, bottleneck in the baseline algorithm, other algorithms
6. **Analyze** your solution; goal met? Trade-offs?

Follow this process for each of the following computational problems. For each problem, your submission should include:

a. State are the input variables and what are the output variables
b. Pseudocode for your baseline algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
a. The Θ-notation time complexity of your baseline algorithm, with justification.

and if you manage to create an improved algorithm:

c. Answer the question: how is your improved algorithm different from your baseline; what did you change to make it faster?
d. Pseudocode for your improved algorithm, that needs to include the data type and an explanation for any variable other than input and output variables
a. The Θ-notation time complexity of your improved algorithm, with justification.

Today's problems are:

1.

List the following functions according to their order of growth from the lowest to the highest:

2^{2n}, $0.001n^4 + 3n^3 + 1$, $\ln^2 n$, $\sqrt[3]{n}$

2.

Consider the problem of sorting an array of *n* floating point numbers which are in range from 0 to 1 and have at most two digits after the decimal point. The elements in the array are not necessarily uniformly distributed across the range [0..1]. Write a linear time O(n) algorithm to sort these *n* values.

Example:

Input : arr[] = { 0.89, 0.7, 0.56, 0.5, 0.65, 0.6, 0.12, 0.34, 0, 0.34, 0.4 }

Output : 0 0.12 0.34 0.34 0.4 0.5 0.56 0.6 0.65 0.7 0.89

3. (Forming the largest number,)

Using Figure 8.3 (below) as a model, illustrate the operation of RADIX-SORT on the following list of English words: BAD, BUT, DAG, FOG, HOT, IAN, NOW, PUT, RIG, TUG, VIO, WAS, TOP, CAT, ARC, OUT.

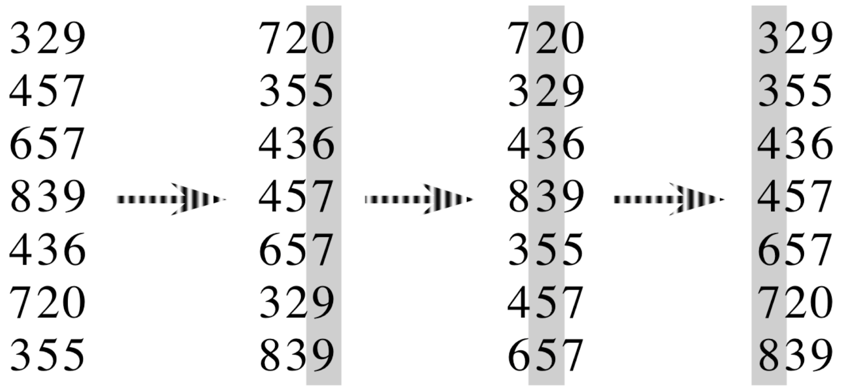


Figure 8.3

Names

Write the names of all group members below.

❎ Jeevika Yarlagadda

Joseph Eggers

Hetal Patel

Exercise 1: Solve and provide answer

❎ List the following functions according to their order of growth from the lowest to the highest:

2^{2n}, $0.001n^4 + 3n^3 + 1$, $\ln^2 n$, $\sqrt[3]{n}$

Joseph solution:

$\sqrt[3]{n}$

$\ln^2 n$ Incorrect

$0.001n^4 + 3n^3 + 1$

2^{2n}

Jeevika solution:

$\ln^2 n$

$\sqrt[3]{n}$ Correct

$0.001n^4 + 3n^3 + 1$

2^{2n}

Exercise 2: Solve and provide answer

Understand

Input/Output

Read and discuss the problem statement, then answer: **Describe the input and output definition in your own words.**

Input : arr[] = an array of floating point numbers from 0 to 1
Output : return sorted array of floating point numbers from 0 to 1

Examples

Write out a concrete input value for the problem, and the concrete correct output. Repeat this at least once (so there are at least two input/outputs).

Input : arr[] = { 0.12, 0.34, 0, 0.34, 0.4 }
Output : 0 0.12 0.34 0.34 0.4

Input: arr[] = {0.01, 0.4, 0.2, 0.1}
Output: 0.01 0.1 0.2 0.4

Corner Cases

A corner case is a special case of input that is particularly unusual or challenging. Are there corner cases for this problem? If so, describe them.

null list with no elements in the list

Use Case

Think of a specific use case for an algorithm that solves this problem. What kind of feature could it support in practical real-world software?

Sorting percentage grades in class for curving grades, ranking students, etc…

Similar Problems

Does this problem resemble any other problems you remember? If so, which problem? Does that problem have an efficient algorithm? If so, what is the run time of that algorithm, inΘ-notation?

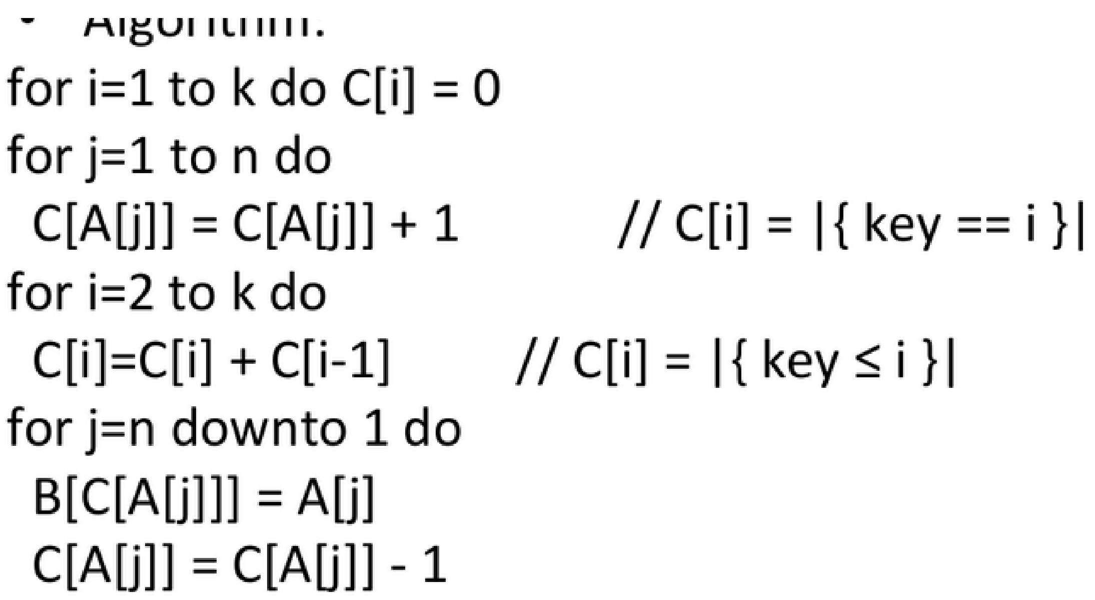
Radix sort since we divide it by 100, Counting sort

Baseline

Design Overview

Design a naïve algorithm that solves the problem. This should be a simple algorithm that is easy to describe and understand. First, describe the basic idea of your algorithm in 1-2 sentences.

We given a list from 0 to 1 , we return a stored list by using counting sort.



```
- Algorithm:
for i=1 to k do C[i] = 0
for j=1 to n do
  C[A[j]] = C[A[j]] + 1          // C[i] = |{ key == i }|
for i=2 to k do
  C[i]=C[i] + C[i-1]         // C[i] = |{ key ≤ i }|
for j=n downto 1 do
  B[C[A[j]]] = A[j]
  C[A[j]] = C[A[j]] - 1
```

Pseudocode

Now, write clear pseudocode for your baseline algorithm.

```
// On^2 solution (first solution)
Int main(list n ){
        Array countList[100] // create a list of 100 indexes
        Array  answerList;
```

```
          for  i in n
                    Temp = N[i] * 100
                    countList[Temp ] = countList[Temp ] + 1
          Endfor

          For i in countList
                    If  ( countList[i] == 0 ) continue;
                    For j in countList[i]
                              answerList.push_back(i/100)       // get float number
                    Endfor
          Endfor

}
```

Correct

(Improved Solution)
Jeevikas solution:

```
K = arr.length()
For i to k do
          Arr[i] = 100*arr[i]

For i =1 to k do
         If  arr[i] > max       // considering max as int_max
          Max = arr[i]

Output = arr[k]
Count = arr[max]

For i =1 to k do
          Count[i] = 0
For j=1 to k do
          Count[arr[j]]+=1
For i =2 to k do
          Count[i] = county[i]+count[i-1]
For i=k to 1 do
          Output[count[arr[i]] = arr[i];
          Count[arr[i]] = -1
For i=1 to k do
          Arr[i] = arr[i]/100 -> to get the floating numbers, since it is similar to radix sort we divide it by 100


Int main(){
```

```
Float arr[] = {0.1, 0.01, 0.4, 0.2}
int n = sizeof(arr) / sizeof(arr[0]);
bucketSort(arr, n);

Return 0;
}

bucketSort(float arr[], int n){

// create buckets
Vector <float> buc[n];

//insert elements
for(int i=0; i<n; i++){
     Int bucI = n * arr[i] // bucket index
     buc[bucI].push_back(arr[i]);
}

//sort the buckets
For (int i =0; i<n;i++)
     sort(buc[i].begin(),buc[i].end());

// Print the buckets
  for (i = 0; i < n; i++) {
    cout << buc[i];
  }


}
```

Efficiency

What is the Θ-notation time complexity of your baseline algorithm? Justify your answer.

☒ Θ(n)
All the loops in the code runs from 0 to size of array, which is n
I.e, the statements in the loop run n times
Thus it is a complexity with Θ(n)

1st solution has two nested for loops which would cause it to be Θ(n^2)

Improvement (Optional)

Now, steps 3-6 involving creating a better algorithm. This part is optional. Only attempt this part if you finished the earlier parts and still have time.

Goal

What aspect of your baseline algorithm do you want to improve? Time efficiency, or something else?

Design Overview

Design a better algorithm for the same problem. First, describe the basic idea of your algorithm in 1-2 sentences.

Pseudocode

Now, write clear pseudocode for your improved algorithm.

Efficiency

What is the Θ-notation time complexity of your improved algorithm? Justify your answer.

Analysis

Did you meet your goal? Why or why not?

Exercise 3: Solve and provide answer

4. *(Forming the largest number,)*

Using Figure 8.3 (below) as a model, illustrate the operation of RADIX-SORT on the following list of English words: BAD, BUT, DAG, FOG, HOT, IAN, NOW, PUT, RIG, TUG, VIO, WAS, TOP, CAT, ARC, OUT.

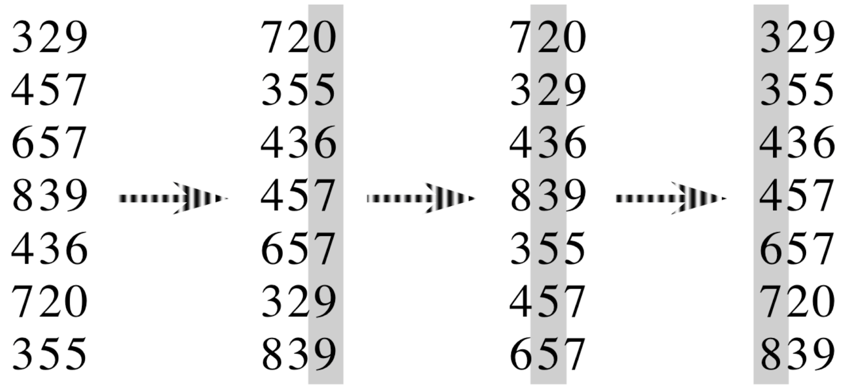


Figure 8.3

Using Radix sort, we show the below operations:

BAD,		ARC		BAD		ARC
DAG,		BAD		DAG		BAD
BUT,		DAG		IAN		BUT
FOG,		FOG,		WAS		CAT
HOT,		RIG,		CAT		DAG
IAN,	——->	TUG		RIG		FOG
NOW,		IAN,		VIO		HOT
PUT,		VIO		FOG	—->	IAN
RIG,		TOP	——-->	TOP		NOW
TUG,		WAS		HOT		OUT
VIO,		BUT		NOW		PUT
WAS,		HOT		ARC		RIG
TOP,		PUT		TUG		TOP
CAT,		CAT		BUT		TUG
ARC,		OUT		PUT		VIO
OUT.		NOW		OUT		WAS

Correct