


Grade:
9/9

Exercise 4 ~ Monday, March 22 ~ CPSC 535 Spring 2021

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
- c. The Θ -notation time complexity of your baseline algorithm, with justification.

and if you manage to create an improved algorithm:

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

Today's problems are:

1.

Show all legal B-trees of minimum degree 2 that represent {4,6,8,10}.

2.

Given an array of $2n$ elements in the following format { $a_1, a_2, a_3, a_4, \dots, a_n, b_1, b_2, b_3, b_4, \dots, b_n$ }. The task is shuffle the array to { $a_1, b_1, a_2, b_2, a_3, b_3, \dots, a_n, b_n$ } without using extra space.

Examples:

Input : $arr[] = \{ 1, 2, 9, 15 \}$

Output : 1 9 2 15

Input : $arr[] = \{ 1, 2, 3, 4, 5, 6 \}$

Output : 1 4 2 5 3 6

3.

Find what elements are needed to make an array to become a subset of another array: Given two arrays: $arr1[0..m-1]$ and $arr2[0..n-1]$ each having non-repetitive (distinct) elements, find what elements from **$arr2[]$ need to be added to $arr1[]$** in order for $arr2[]$ to become a subset of $arr1[]$. The arrays are not necessarily in sorted order.

Examples:

Input: $arr1[] = \{11, 1, 13, 21, 3, 7\}$, $arr2[] = \{11, 3, 7, 1\}$

Output: Set of elements = Empty set (since $arr2[]$ is already a subset of $arr1[]$)

Input: $arr1[] = \{1, 2, 3, 4, 5, 6\}$, $arr2[] = \{1, 2, 4\}$

Output: Set of elements = Empty set (since $arr2[]$ is already a subset of $arr1[]$)

Input: $arr1[] = \{10, 5, 2, 23, 19\}$, $arr2[] = \{19, 5, 3\}$

Output: Set of elements = [3]

Your goal: baseline $\Theta(m * n)$ time, improved to $\Theta(n \log n + m \log m)$ worst-case time or $\Theta(m + n)$ expected time.

Names

Write the names of all group members below.

☒ Kristopher Swartzbaugh

Austin Nguyen

Binh Trinh

Bryan Ambriz

Exercise 1: Solve and provide answer

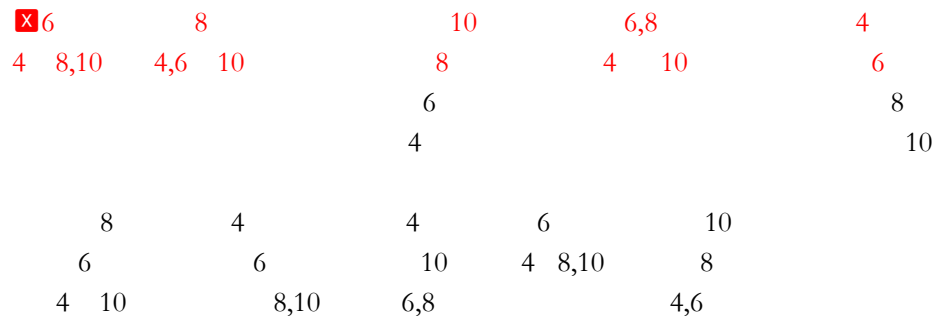
Show all legal B-trees of minimum degree 2 that represent $\{4,6,8,10\}$.

Deg = 2

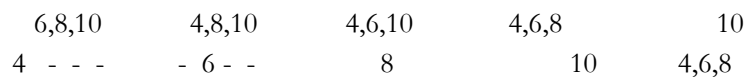


Not all are correct since the trees need to be balanced

deg = 3



Deg = 4



Deg >= 5

4,6,8,10

Exercise 2: Solve and provide answer

Given an array of $2n$ elements in the following format $\{a_1, a_2, a_3, a_4, \dots, a_n, b_1, b_2, b_3, b_4, \dots, b_n\}$. The task is shuffle the array to $\{a_1, b_1, a_2, b_2, a_3, b_3, \dots, a_n, b_n\}$ without using extra space.

Examples:

Input : arr[] = { 1, 2, 9, 15 }

Output : 1 9 2 15

Input : arr[] = { 1, 2, 3, 4, 5, 6 }
Output : 1 4 2 5 3 6

Input/Output

Input: an arr ARR of n elements

Output: the same array, with the elements properly shuffled

Examples

Input : arr[] = { 1, 2, 9, 15 }

Output : 1 9 2 15

Input : arr[] = { 1, 2, 3, 4, 5, 6 }

Output : 1 4 2 5 3 6

Corner Cases

There are no particularly challenging corner cases

Use Case

This can serve as a simple shuffling algorithm

Similar Problems

Design Overview

For each element in the b sub array, insert it into the a sub array at the correct index, and shift all elements over.

Pseudocode

```
Void Shuffle(arr) {  
    Int n = length of arr / 2  
  
    Int b = 0  
  
    While (b < n) {  
        Int c = arr[b + n]  
        Shift all elements from arr[b*2 + 1] to arr[b + n - 1] (inclusive) 1 element to the right  
        arr[b*2 + 1] = c  
    }
```

```

        }
        b++
    }
}

```

Correct algorithm, but I assume that the space complexity is $O(1)$, right?

$N = 4$

{ a1, a2, a3, a4, b1, b2, b3, b4, }

{ a1, b1, a2, a3, a4, b2, b3, b4, }.

{ a1, b1, a2, b2, a3, a4, b3, b4, }.

{ a1, b1, a2, b2, a3, b3, a4, b4, }.

Efficiency

$O(n^2)$

While loop loops n times.

Inside the loop, we shift part of the array over. The size of the sub array ranges for 0 to n , so the shuffle step is $O(n)$

Improvement

Possible to improve time to $n \log n$

Observation: first half of a sub array remains in the first half of the whole array, second half ends up in the second half.

Similar is true for the b sub array.

Swap the second half of a sub array with first half of b sub array

Each half of the array now contains the correct set of elements, just in the wrong order

{ a1, a2, a3, a4, b1, b2, b3, b4, }

=>

{ a1, a2, b1, b2, a3, a4, b3, b4, }

^^^ By lucky happenstance, each half has same structure as input array

Recursively apply method to each sub half

Exercise 3: Solve and provide answer

Find what elements are needed to make an array to become a subset of another array: Given two arrays: $arr1[0..m-1]$ and $arr2[0..n-1]$ each having non-repetitive (distinct) elements, find what elements from $arr1[]$ need to be added to $arr2[]$ in order for $arr2[]$ to become a subset of $arr1[]$. The arrays are not necessarily in sorted order.

Examples:

Input: $arr1[] = \{11, 1, 13, 21, 3, 7\}$, $arr2[] = \{11, 3, 7, 1\}$

Output: Set of elements = Empty set (since $arr2[]$ is already a subset of $arr1[]$)

Input: $arr1[] = \{1, 2, 3, 4, 5, 6\}$, $arr2[] = \{1, 2, 4\}$

Output: Set of elements = Empty set (since $arr2[]$ is already a subset of $arr1[]$)

Input: $arr1[] = \{10, 5, 2, 23, 19\}$, $arr2[] = \{19, 5, 3\}$

Output: Set of elements = [3]

Your goal: baseline $\Theta(m * n)$ time, improved to $\Theta(n \log n + m \log m)$ worst-case time or $\Theta(m + n)$ expected time.

Understand

Input/Output

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

- ✖ Input: two arrays where each contains unique elements in random order
- ✖ Output: The array of numbers that must be added to $arr1$ so that $arr2$ is a subset of $arr1$

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).

- ✖ $arr1 = [12, 13, 14, 15, 16]$
 $arr2 = [11, 12, 13, 14]$
 Output: [11]
- ✖ $arr1 = [9, 12, 14, 18, 29]$
 $arr2 = [29, 12, 17]$
 Output: [17]

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.

- ✖ If the $arr2$ is already a subset of $arr1$
 If all elements in $arr2$ must be added to $arr1$ for $arr2$ to be a subset of $arr1$

If arr1 or arr2 are the empty set

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?

✖ Merging data that may contain a lot of the same items.

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, in Θ -notation?

✖ This process is similar to determining if two arrays are identical. $O(n*m)$ run time.

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.

✖ For each element in arr2, check to see if that element is in arr1. If it is not, add it to the output.

Pseudocode

Now, write clear pseudocode for your baseline algorithm.

```
✖ output = []  
  For items in arr2  
    If items is not in arr1  
      Add items to output  
  Print output
```

Correct

Efficiency

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

✖ $O(m*n)$. Worst case scenario for each item in arr2 you have to loop through all of arr1 doing a comparison.

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?

✗ Time efficiency by applying preprocessing to make the data easier to search through.

Design Overview

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

✗ Sort arr1, then for each element in arr2 you can do a binary search, if the element is not found add it to the output.

Pseudocode

Now, write clear pseudocode for your improved algorithm.

✗

```
Quicksort arr1
  For each item in arr2
    Binary search sorted arr1
    If not found, add to output
Print output
```

Quicksort is $O(n^2)$ so your algorithm has the same time complexity as brute-force one

Efficiency

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

✗ $O(m \log m + n \log n)$ which is the preprocessing time (sort arr1) and the binary search time.

Analysis

Did you meet your goal? Why or why not?

✗ Yes, we reduced the time complexity from the original $O(m \cdot n)$ to the $O(m \log m + n \log n)$ requested.