MECHTRON 2MD3
Data Structures and Algorithms for Mechatronics
Winter 2022

Tutorial 07

Published: Mar 4, 2022

I recommend you try to solve the problems by yourself before the tutorial. Problems will be solved during the tutorial hours. I have uploaded the companion code in the same folder as this document.

1 Question 1

An array A contains $n-1$ unique integers in the range $[0, n-1]$ (both ends inclusive), that is, there is one number from this range that is not in A. Remember that inclusive range $[0, n-1]$ contains n integers, so when we say an array contains $n-1$ unique integers of that range, it basically means there is one integer missing.

An example of such an array could be $[4, 2, 0, 6, 1, 5]$. It contains 6 elements in the range $[0, 6]$, and obviously element 3 is missing.

Design an algorithm for finding that missing number that fulfills two requirements:

1. with time complexity of $O(n)$
2. You are only allowed to use $O(1)$ additional space besides the array A itself. This means, for example, you are not allowed to use $O(n)$ space; that is, you can not use a space that is proportional to n.

2 Question 2

Suppose we have two different sorting algorithms A and B that given an input array, they can sort that array. The number of primitive operations executed by algorithms A and B is $8nlogn$ and $2n^2$, respectively for an input size of n. Assuming that the primitive operations all take the same time, for which values of n does algorithm B beat algorithm A? (Note: logarithm should be considered as base 2)

3 Question 3

Arrange the following functions by their order of asymptotic growth rate from smaller to larger.

- $4nlogn + 2n$
- 2^n
- 2^{10}
- 4^{logn}
- $2n^2 + 10n$
- 2^{logn}
- $3n + 100logn$
- $nlogn$
- $4n$
- n^3

4 Question 4

Which one is correct? Use the definition of the Big-Oh notation to solve this problem.

- 2^{n+1} is $O(2^n)$
- 2^{2n} is $O(2^n)$

5 Question 5

What is the expected output of the following code fragment? Also, give a high-level description of the task that the algorithm does when it is given an arbitrary positive integer n. Note that `ArrayStack` is an array-based implementation of Stack data structure that you already have seen during the lectures.

```
int main(){
    ArrayStack<int> A;
    int n = 43;
    while (n > 0){
        A.push(n % 2);
        n = n / 2;
    }
    while (!A.empty()){
        cout << A.top();
        A.pop();
    }
    cout << endl;
}
```

6 Question 6

We have seen two implementations of Stack, one using the array and the other using the singly linked list. We also have talked about the fact that we can implement the array-based Stack in a way that when it becomes full, it can perform a resize operation and make its internal array bigger. For example, it can double the array's size and copy the values from the old array to the new array. We also discussed the fact that the Stack implemented using the linked list does not need any resize and never gets full (logically). Why is sometimes the array-based implementation and resorting to resize operation **preferable** over the linked list-based implementation?