CS 61B Fall 2020

Asymptotic Analysis

Discussion 7: October 5, 2020

1 Binary Search Trees

(a) For a BST that contains N items, what is the asymptotic worst case of each for each of these functions?

i. `add(int x)`

ii. `getMin()`

(b) Now assume that the BST from part (a) is certain to be bushy. What is our new asymptotic worst case?

i. `add(int x)`

ii. `getMin()`

(c) Let's implement the `find` function in our BSTMap. It should take in an integer and return the value associated with that key or null if the key is not in our BSTMap.

```
public class BSTMap {
    private class Node {
        int key;
        int value;
        Node left;
        Node right;
        Node (int key, int value) { ... }
    }

    Node head; // Contains the node at the head of the tree
    ...
    public Integer find(int key) {




    }
    private Integer findHelper(int key, Node n) {









    }
}
```

2 I Am Speed

Give the worst case and best case running time in $\Theta(\cdot)$ notation in terms of M and N. Assume that `comeOn()` is in $\Theta(1)$ and returns a boolean.

```
for (int i = 0; i < N; i += 1) {
    for (int j = 1; j <= M; ) {
        if (comeOn()) {
            j += 1;
        } else {
            j *= 2;
        }
    }
}
```

3 Have You Ever Went Fast?

Given an **int** `x` and a *sorted* array `A` of N distinct integers, design an algorithm to find if there exists indices `i` and `j` such that `A[i] + A[j] == x`.

Let's start with the naive solution.

```
public static boolean findSum(int[] A, int x) {
    for (int i = 0; i < A.length; i++){
        for (int j = 0; j < A.length; j++) {
            if (A[i] + A[j] == x) return true;
        }
    }
    return false;
}
```

(a) How can we improve this solution? *Hint*: Does order matter here?

(b) What is the runtime of both the original and improved algorithm?

4 CTCI *Extra*

(a) **Union** Write the code that returns an array that is the union between two given arrays. The union of two arrays is a list that includes everything that is in both arrays, with no duplicates. Assume the given arrays do not contain duplicates. For example, the union of `{2, 1, 3, 4}` and `{3, 4, 6, 5}` is `{1, 2, 3, 4, 5, 6}`. The method should run in $O(M+N)$ time where M and N is the size of each array. The returned list does not need to remain in the same order as the elements of the input lists.

Hint: Think about using ADTs other than arrays to make the code more efficient.

(b) **Intersect** Now do the same as above, but find the intersection between both arrays. The intersection of two arrays is the list of all elements that are in both arrays. Again assume that neither array has duplicates. For example, the intersection of `{1, 2, 3, 4}` and `{3, 4, 5, 6}` is `{3, 4}`. The returned list does not need to remain in the same order as the elements of the input lists.

Hint: Like in part (a), think about using ADTs other than arrays to make the code more efficient.