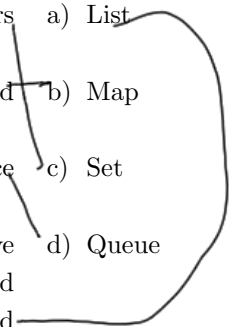


## 1 ADT Matchmaking

Match each task to the correct Abstract Data Type for the job by drawing a line connecting matching pairs.

- |   |          |
|---|----------|
| 1. You want to keep track of all the unique users who have logged on to your system.  | a) List  |
| 2. You are creating a version control system and want to associate each file name with a Blob.  | b) Map   |
| 3. We are running a server and want to service clients in the order they arrive.  | c) Set   |
| 4. We have a lot of books at our library and we want our website to display them in some sorted order. We have multiple copies of some books and we want each listing to be separate. | d) Queue |
- 

## 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.

```
1  for (int i = 0; i < N; i += 1) {  
2      for (int j = 1; j <= M; ) {  
3          if (comeOn()) {  
4              j += 1;  
5          } else {  
6              j *= 2;  
7          }  
8      }  
9  }
```

$\Theta(NM)$

$\Theta(N \log M)$

### 3 Re-cursed with Asymptotics!

- (a) What is the runtime of the code below in terms of  $n$ ?

```

1 public static int[] curse(int n) {
2     if (n <= 0) {
3         return 0;
4     } else {
5         return n + curse(n - 1);
6     }
7 }

```

$O(n)$

- (b) Assume our BST (Binary Search Tree) below is perfectly bushy. What is the runtime of a single find operation in terms of  $N$ , the number of nodes in the tree? In this setup, assume a Tree has a Key (the value of the tree) and then pointers to two other trees, Left and Right.

```

1 public static BST find(BST T, Key sk) {
2     if (T == null)
3         return null;
4     if (sk.compareTo(T.key) == 0)
5         return T;
6     else if (sk.compareTo(T.key) < 0)
7         return find(T.left, sk);
8     else
9         return find(T.right, sk);
10 }

```

$O(\log(N))$

- (c) Can you find a runtime bound for the code below? We can assume the `System.arraycopy` method takes  $\Theta(N)$  time, where  $N$  is the number of elements copied. The official signature is `System.arraycopy(Object sourceArr, int srcPos, Object dest, int destPos, int length)`. Here, `srcPos` and `destPos` are the starting points in the source and destination arrays to start copying and pasting in, respectively, and `length` is the number of elements copied.

```

1 public static void silly(int[] arr) {
2     if (arr.length <= 1) {
3         System.out.println("You won!");
4         return;
5     }
6     int newLen = arr.length / 2;
7     int[] firstHalf = new int[newLen];
8     int[] secondHalf = new int[newLen];
9     System.arraycopy(arr, 0, firstHalf, 0, newLen);
10    System.arraycopy(arr, newLen, secondHalf, 0, newLen);
11    silly(firstHalf);
12    silly(secondHalf);
13 }

```

$\Theta(N \log N)$

## 4 Have You Ever Went Fast? *Extra*

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.

```

1 public static boolean findSum(int[] A, int x) {
2     for (int i = 0; i < A.length; i++){
3         for (int j = 0; j < A.length; j++) {
4             if (A[i] + A[j] == x) return true;
5         }
6     }
7     return false;
8 }
```

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

```

int left = 0;
int right = A.length - 1;
while (left <= right)
    if (A[left] + A[right] == x)
        return true;
    else if (A[left] + A[right] < x) left++;
    else right--;
```

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

$\Theta(N^2)$      $\Theta(N)$