CS 61B Spring 2018

Asymptotic Analysis

Discussion 7: February 27, 2018

1 Asymptotic Notation

1.1 Order the following big-O runtimes from smallest to largest.

$$O(\log n), O(1), O(n^n), O(n^3), O(n \log n), O(n), O(n!), O(2^n), O(n^2 \log n)$$

$O(1)$ $O(\log n)$ $O(n)$ $O(n\log n)$ $O(n^2\log n)$ $O(n^3)$ $O(2^n)$ $O(n!)$ $O(n^n)$

1.2 Are the statements in the right column true or false? If false, correct the asymptotic notation ($\Omega(\cdot)$, $\Theta(\cdot)$, $O(\cdot)$). Be sure to give the tightest bound. $\Omega(\cdot)$ is the opposite of $O(\cdot)$, i.e. $f(n) \in \Omega(g(n)) \iff g(n) \in O(f(n))$.

$f(n) = 20501$	$g(n) = 1$	$f(n) \in O(g(n))$	✓	
$f(n) = n^2 + n$	$g(n) = 0.000001n^3$	$f(n) \in \Omega(g(n))$	✗	$f(n) \in O(g(n))$
$f(n) = 2^{2n} + 1000$	$g(n) = 4^n + n^{100}$	$f(n) \in O(g(n))$	✓	
$f(n) = \log(n^{100})$	$g(n) = n \log n$	$f(n) \in \Theta(g(n))$	✗	$f(n) \in O(g(n))$
$f(n) = n \log n + 3^n + n$	$g(n) = n^2 + n + \log n$	$f(n) \in \Omega(g(n))$	✓	
$f(n) = n \log n + n^2$	$g(n) = \log n + n^2$	$f(n) \in \Theta(g(n))$	✓	
$f(n) = n \log n$	$g(n) = (\log n)^2$	$f(n) \in O(g(n))$	✗	$f(n) \in \Omega(g(n))$

2 Analyzing Runtime

2.1 Give the worst case and best case runtime in terms of M and N.Assume `ping` is in $\Theta(1)$ and returns an **int**.

```java
int j = 0;
for (int i = N; i > 0; i--) {
    for (; j <= M; j++) {
        if (ping(i, j) > 64) {
            break;
        }
    }
}
```

best case: $\Theta(N)$

worst case: $\cancel{\Theta(MN)}$

$\Theta(M+N)$

2.2 Give the worst case and best case runtime where N = `array.length`. Assume `mrpoolsort(array)` is in $\Theta(N \log N)$.

```
public static boolean mystery(int[] array) {
    array = mrpoolsort(array);
    int N = array.length;
    for (int i = 0; i < N; i += 1) {
        boolean x = false;
        for (int j = 0; j < N; j += 1) {
            if (i != j && array[i] == array[j])
                x = true;
        }
        if (!x) {
            return false;
        }
    }
    return true;
}
```

best case $\Theta(N \log N)$

worst case: $\Theta(N^2)$

Achilles Added Additional Amazing Asymptotic And Algorithmic Analysis Achievements

(a) What is `mystery()` doing?

to see if all of item has duplicates

(b) Using an ADT, describe how to implement `mystery()` with a better runtime. Then, if we make the assumption an **int** can appear in the **array** at most twice, develop a solution using only constant memory.

$\Theta(N)$ → use a map

2.3 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;
        }
    }
}
```

best: $\Theta(N \log M)$

worse: $\Theta(MN)$

3 Have You Ever Went Fast?

3.1 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.

```java
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?

left right

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

$\Theta(N^2)$

best: $\Theta(1)$ worst: $\Theta(N)$

4 CTCI *Extra*

4.1 **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 `{1, 2, 3, 4}` and `{3, 4, 5, 6}` is `{1, 2, 3, 4, 5, 6}`.

Hint: The method should run in $O(M + N)$ time where M and N is the size of each array.

4.2 **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}`.

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