CS 61B Discussion 8 Spring 2017

1 More Running Time

Give the worst case and best case running time in $\Theta(\cdot)$ notation in terms of M and N.

(a) Assume that `comeon()` $\in \Theta(1)$ 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;
    }
}
```

2 Recursive Running Time

For the following recursive functions, give the worst case and best case running time in the appropriate $O(\cdot)$, $\Omega(\cdot)$, or $\Theta(\cdot)$ notation.

(a) Give the running time in terms of N.

```
public void andslam(int N) {
    if (N > 0) {
        for (int i = 0; i < N; i += 1) {
            System.out.println("datboi.jpg");
        }
        andslam(N / 2);
    }
}
```

(b) Give the running time for `andwelcome(arr, 0, N)` where N is the length of the input array `arr`.

```
public static void andwelcome(int[] arr, int low, int high) {
    System.out.print("[ ");
    for (int i = low; i < high; i += 1) {
         System.out.print("loyal  ");
    }
    System.out.println("]");
    if (high - low > 0) {
        double coin = Math.random();
        if (coin > 0.5) {
            andwelcome(arr, low, low + (high - low) / 2);
        } else {
            andwelcome(arr, low, low + (high - low) / 2);
            andwelcome(arr, low + (high - low) / 2, high);
        }
    }
}
```

(c) Give the running time in terms of N.

```java
public int tothe(int N) {
    if (N <= 1) {
        return N;
    }
    return tothe(N - 1) + tothe(N - 1);
}
```

(d) *Extra Hard!* Give the running time in terms of N

```java
public static void spacejam(int N) {
    if (N == 1) {
        return;
    }
    for (int i = 0; i < N; i += 1) {
        spacejam(N - 1);
    }
}
```

3 Hey you watchu gon do?

For each example below, there are two algorithms solving the same problem. Given the asymptotic runtimes for each, is one of the algorithms **guaranteed** to be faster? If so, which? And if neither is always faster, explain why. Assume the algorithms have very large input (so N is very large).

(a) Algorithm 1: $\Theta(N)$, Algorithm 2: $\Theta(N^2)$

(b) Algorithm 1: $\Omega(N)$, Algorithm 2: $\Omega(N^2)$

(c) Algorithm 1: $O(N)$, Algorithm 2: $O(N^2)$

(d) Algorithm 1: $\Theta(N^2)$, Algorithm 2: $O(\log N)$

(e) Algorithm 1: $O(N \log N)$, Algorithm 2: $\Omega(N \log N)$

Would your answers above change if we did not assume that N was very large?

4 More Extra Problems [Final FA15]

If you have time try, to answer this challenge question. For each answer true or false. If true, explain why and if false provide a counterexample.

(a) If $f(n) \in O(n^2)$ and $g(n) \in O(n)$ are positive-valued functions (that is for all n, $f(n), g(n) > 0$), then $\frac{f(n)}{g(n)} \in O(n)$.

(b) If $f(n) \in \Theta(n^2)$ and $g(n) \in \Theta(n)$ are positive-valued functions, then $\frac{f(n)}{g(n)} \in \Theta(n)$.