CS 61B Asymptotic Analysis Fall 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 `slam()` $\in \Theta(1)$ and returns a boolean.

```java
public void comeon() {
    int j = 0;
    for (int i = 0; i < N; i += 1) {
        for (; j < M; j += 1) {
            if (slam(i, j))
                break;
        }
    }

    for (int k = 0; k < 1000 * N; k += 1) {
        System.out.println("space jam");
    }
}
```

2 Recursive Running Time

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

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

```java
public void andslam(int N) {
    if (N > 0) {
        for (int i = 0; i < N; i += 1) {
            System.out.println("bigballer.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.

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

(d) Give the running time in terms of N

```
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: $\Omega(N\log N)$

Why did we need to assume that N was large?

4 Big Ballin' Bounds

1. Prove the following bounds by finding some constant $M > 0$ and input $N > 0$ for $M \in \mathbb{R}, N \in \mathbb{N}$ such that f and g satisfy the relationship.

   (a) $f \in O(g)$ for $f = 2n$, $g = n^2$

   (b) $f \in \Omega(g)$ for $f = 0.1n$, $g = 40$

   (c) $f \in \Theta(g)$ for $f = \log(n)$, $g = \log(n^a)$, for $a > 0$.

2. Answer the following claims with true or false. If false, provide a counterexample.

   (a) If $f(n) \in O(g(n))$, then $500f(n) \in O(g(n))$.

   (b) If $f(n) \in \Theta(g(n))$, then $2^{f(n)} \in \Theta(2^{g(n)})$.