CS 61B Asymptotic Analysis Fall 2016

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;
        }
    }
}
```

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("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`.

```java
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) 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?