CS 61B Discussion 8 Spring 2016

1 Which is faster?

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?

2 More Runtime Analyzing

A. How many times is `lobsterPainting` called? Give your answer in Θ notation in terms of N, assuming `lobsterPainting` does not crash or call any methods.

```java
for (int i = 0; i < N/2; i++) {
    for (int j = i - 1; j < N/2 + 1; j++) {
        lobsterPainting(i, j);
    }
}
```

B. How about here?

```java
for (int i = N - 1; i > 0; i /= 2) {
    for (int j = 0; j < i; j++) {
        lobsterPainting(i, j);
    }
}
```

C. Bonus: And here?

```java
public static void crabDrawing(int N) {
    for (int i = 1; i < N; i *= 2) {
        lobsterPainting(i, i);
        crabDrawing(i);
    }
}
```

3 More? Of Course More

Describe the best-case and worst-case runtimes of the function individually using Θ. Then use them to describe the overall runtime of the function in terms of Θ (if possible) or *O*/Ω if not.

A. Assume `arr` is a **sorted** array of **unique** elements of size *N*. Example of calling this method would be: `hopps(sortedArr, 0, sortedArr.length)`.

```java
public static int hopps(int[] arr, int low, int high) {
    if (high <= low)
        return -1;
    int mid = (low + high) / 2; // (later, see http://goo.gl/gQI0FN )
    if (arr[mid] == mid)
        return mid;
    else if (mid > arr[mid])
        return hopps(arr, mid + 1, high);
    else
        return hopps(arr, low, mid);
}
```

Bonus: What is `hopps` doing?

B. Assume `str` is a String of characters of size *N*.

```java
public static char wilde(String str) {
    Map<Character,Integer> map = new HashMap<>();
    for (char c : str.toCharArray()) {
        if (map.containsKey(c)) {
            map.put(c, map.get(c) + 1);
        } else {
            map.put(c, 1);
        }
    }
    for (int i = 0; i < str.length(); i++) {
        if (map.get(str.charAt(i)) == 1) {
            return str.charAt(i);
        }
    }
    return 0; // 0 represents the NULL character
}
```

Bonus: What is `wilde` doing?

Bonus's Bonus: Can you do it with only 1 `for` loop?

4 Have You Ever Went Faster? (Extra)

Given an integer x and a **sorted** array A[] of N distinct integers, design an algorithm to find if there exists distinct indices i, j, and k such that `A[i] + A[j] + A[k] == x`. Feel free to write pseudocode instead of Java. Your code should run in $\Theta(N^2)$ time.