CS 61B
Spring 2017

Asymptotic Analysis

Discussion 7: March 1, 2017

1 Computational Cost

1.1 Count the operations it takes to execute `kviate` as a function of N = `a.length`. Assume each of the following operations each take 1 timestep to complete.

· variable declaration · assignment · comparison · array access · increment

```
public static void kviate(int[] a, int k) {
    int count = 0, N = a.length;
    for (int i = 0; i < N; i++ {
        if (a[i] == k) {
            count += 1;
        }
    }
    a[k] += count;
}
```

- Variable declaration: 3
- Assignment: 3
- Comparison: $2N+1$, N equality comparisons, $N+1$ less-than comparisons
- Array access: $N+2$, `a[k] += count` requires both read and write
- Increment: $N+1$ to $2N+1$

2 Analysis of Algorithms

2.1 Give a tight asymptotic runtime bound for `linearSearch` as a function of N, the size of the array, in the *best case, worst case, and overall*.

```
public static boolean linearSearch(int[] a, int value, int start) {
    if (start >= a.length) {
        return false;
    } else if (a[start] == value) {
        return true;
    } else {
        return linearSearch(a, value, start + 1);
    }
}
```

$\Theta(1)$ in the best case, $\Theta(N)$ in the worst case, and $O(N)$ overall.

2.2 Give a tight asymptotic runtime bound for `binarySearch` as a function of N, the size of the array, in the *best case, worst case, and overall*. Assume the array is sorted.

```
public static boolean binarySearch(int[] a, int value, int start, int end) {
    if (start == end || start == end - 1) { return a[start] == value; }
    int mid = end + ((start - end) / 2);
    if (a[mid] == value) {
        return true;
    } else if (a[mid] > value) {
        return binarySearch(a, value, start, mid);
    } else {
        return binarySearch(a, value, mid, end);
    }
}
```

$\Theta(1)$ in the best case, $\Theta(\log N)$ in the worst case, and $O(\log N)$ overall.

2.3 Give a tight asymptotic runtime bound for `mysterySearch` as a function of N, the size of the array, in the *best case, worst case, and overall*. Assume the array is sorted.

```
public static boolean mysterySearch(int[] a, int value) {
    if (Math.random() < 0.5) {
        return linearSearch(a, value, 0);
    } else {
        return binarySearch(a, value, 0, a.length);
    }
}
```

$\Theta(1)$ in the best case, $\Theta(N)$ in the worst case, and $O(N)$ overall.

2.4 For each pair of functions $f(n)$ and $g(n)$, state whether $f(n) \in O(g(n))$, $f(n) \in \Omega(g(n))$, or $f(n) \in \Theta(g(n))$. For example, for $f(n) = n^2$ and $g(n) = 2n^2n + 3$, write $f(n) \in \Theta(g(n))$.

(a) $f(n) = n$ and $g(n) = n^2 - n \implies f(n) \in O(g(n))$

(b) $f(n) = n^2$ and $g(n) = n^2 + n \implies f(n) \in \Theta(g(n))$

(c) $f(n) = 8n$ and $g(n) = n^2 \implies f(n) \in O(g(n))$

(d) $f(n) = 2^n$ and $g(n) = n^2 \implies f(n) \in \Omega(g(n))$

(e) $f(n) = 3^n$ and $g(n) = 2^{2n} \implies f(n) \in O(g(n))$

2.5 For each of the following, state the order of growth using $\Theta(\cdot)$ notation. For example, $f(n) \in \Theta(n)$.

(a) $f(n) = 50 \implies f(n) \in \Theta(1)$

(b) $f(n) = n^2 - 2n + 3 \implies f(n) \in \Theta(n^2)$

(c) $f(n) = n + \cdots + 2 + 1 \implies f(n) \in \Theta(n^2)$

(d) $f(n) = n^{100} + 1.01^n \implies f(n) \in \Theta(1.01^n)$

(e) $f(n) = n^{1.1} + n \log n \implies f(n) \in \Theta(n^{1.1})$