CS 61B
Spring 2017

Asymptotics

Discussion 8: March 8, 2017

1 Practical Tips

1.1 Before attempting to calculate a function's runtime, first try to understand what the function does.

1.2 Try some small sample inputs to get a better intuition of what the function's runtime is like. What is the function doing with the input? How does the runtime change as the input size increases? Can you spot any 'gotchas' in the code that might invalidate our findings for larger inputs?

1.3 Try to lower bound and upper bound the function runtime given what you know about how the function works. This is a good sanity check for your later observations.

1.4 If the function is recursive, draw a call tree to map out the recursive calls. This breaks the problem down into smaller parts that we can analyze individually. Once each part of the tree has been analyzed, we can then reassemble all the parts to determine the overall runtime of the function.

1.5 If the function has a complicated loop, draw a bar graph to map out how much work the body of the loop executes for each iteration.

2 Useful Formulas

2.1 $1 + 2 + 3 + 4 + 5 + \cdots + N \in \Theta(N^2)$

2.2 There are N terms in the sequence $1, 2, 3, 4, 5, \cdots, N$

2.3 $1 + 2 + 4 + 8 + 16 + \cdots + N \in \Theta(N)$

2.4 There are $\log_2 N$ terms in the sequence $1, 2, 4, 8, 16, \cdots, N$

2.5 The number of nodes in a tree is equal to k^h where k is the branching factor and h is the height of the tree

2.6 All logarithms are proportional to each other by the Change of Base formula.

3 Dorado

3.1 Give a tight asymptotic bound for `convoluted` as a function of N, the length of the input arrays `a` and `b`. If possible, give a $\Theta(\cdot)$ bound for the overall runtime. Otherwise, provide a $\Theta(\cdot)$ bound for both the best case and worst case runtime.

```
int[] convoluted(int[] a, int[] b) {
    assert a.length == b.length;
    int[] result = new int[a.length];
    for (int i = 0; i < result.length; i += 1) {
        for (int j = 0; j <= i; j += 1) {
            result[i] += a[j] * b[i];
        }
    }
    return result;
}
```

3.2 Give a tight asymptotic bound for `debugBinarySearch` as a function of N, the length of the input array, `a`. If possible, give a $\Theta(\cdot)$ bound for the overall runtime. Otherwise, provide a $\Theta(\cdot)$ bound for both the best case and worst case runtime.

```
boolean debugBinarySearch(int[] a, int target) {
    int start = 0;
    int end = a.length - 1;

    while (start <= end) {
        int mid = ((end - start) / 2) + start;
        if (a[mid] == target) {
            return true;
        } else if (a[mid] < target) {
            start = mid + 1;
        } else {
            end = mid - 1;
        }

        System.out.print("Searching: [ ");
        for (int i = start; i <= end; i++) {
            System.out.print(a[i] + " ");
        }
        System.out.println("]");
    }
    return false;
}
```