CS 162 Notes (Winter 2017)

Justin Lubin

Contents

Chapter 1

Asymptotic Analysis I

1.1 Analyzing Code (Worst-Case)

1.1.1 Constant Operations (Approximations)

- Arithmetic (fixed width)
- Assignment
- Access any array element

1.1.2 Non-Constant Operations

Control Flow	Time
Consecutive statements	Sum of time of each statement
Conditional	Time of test + time of the slower branch
Loop	Number of iterations * time of body
Function call	Time of function body
Recursion	Solve recurrence relation

1.2 Reducing Big-O Expressions

1.2.1 What to Eliminate

- Eliminate low-order terms
- Eliminate coefficients

1.2.2 Examples

- $4n + 5 \in O(n)$
- $\frac{1}{2}n \log n + 2n + 7 \in O(n \log n)$
- $n^3 + 2^n + 3n \in O(2^n)$
- $n \log(10n^2) + 2n \log n \in O(n \log n)$
  - Note that $n \log(10n^2) = 2n \log(10n)$

1.3 Linear Search

1.3.1 Code

```
int find(int[] arr, int arr_length, int k) {
    for (int i = 0; i < arr_length; ++i) {
        if (arr[i] == k) {
            return 1;
        }
    }
    return 0;
}
```

1.3.2 Analysis

- Best case: approximately six steps
  - $O(1)$
- Worst case: $6 *$ arr_length steps
  - $n =$ arr_length, so $O(n)$

1.4 Binary Search

1.4.1 Code

```
int find(int[] arr, int k, int lo, int hi) {
    return help(arr, k, 0, arr_length);
}

int find(int[] arr, int arr_length, int k) {
    int mid = (hi + lo) / 2;
    if (lo == hi) {
        return 0;
    }
    if (arr[mid] == k) {
        return 1;
```

```
    }
    if (arr[mid] < k) {
        return help(arr, k, mid + 1, k);
    } else {
        return help(arr, k, lo, mid);
    }
}
```

1.4.2 Analysis

Let $T(n)$ be the efficiency of `find`. Then, because each split takes approximately ten operations, we have that:

$$\begin{aligned} T(n) &= 10 + T\left(\frac{n}{2}\right) \\ &= 10 + \left(10 + T\left(\frac{n}{4}\right)\right) \\ &= 10 + \left(10 + \left(10 + T\left(\frac{n}{8}\right)\right)\right) \\ &= 10k + T\left(\frac{n}{2^k}\right). \end{aligned}$$