Today
- Analysis Techniques (intro)

Assignments
- Exercise 1 (due thur)
- Quiz 1 on thur
**Basic Algorithm Analysis Ideas**

**Input size** $n$
- We analyze algorithms in terms of the size of their input (e.g., $n$ elements)
- The idea is that algorithms often take longer the more input they have
- E.g., sorting 5 elements is usually faster than sorting 5 million

**“Worst Case” Analysis**
- Certain input patterns (for all $n$) that take the most amount of time
- E.g., some sorting algorithms take longer if input is in reverse order
- Can be many worst cases for an algorithm
- Can be algorithms where all the cases are the same (no worst case)
- Big-O notation is usually related to the worst cases (depending on context)

**“Best Case” Analysis**
- Certain input patterns (for all $n$) that take the least amount of time
- E.g., some sorting algorithms take less time if input is already sorted
- Can be many best cases, or none (all cases the same)
“Average Case” Analysis

- The average amount of time over all inputs (for all $n$)
- Often based on “typical” input frequency distributions

One notion of efficiency analysis: performance testing

- run an implementation and time it (for different input sizes)
- depends heavily on specific implementation, language, machine, etc

Another notion of efficiency analysis: count primitive operations

- Primitive steps (e.g., assembly instructions) as a surrogate for time
- We assume each primitive step takes roughly same amount of (unit) time
- We want to know how many primitive steps needed relative to input size
**Detailed Analysis**

**Goal:** Define $T(n)$ giving number of steps as a function of input size $n$

**Example 1:**

```c
bool member(const int A[], int n, int t) {
    bool found = false;
    for (int i = 0; !found and i < n; ++i)
        if (A[i] == t)
            found = true;
    return found;
}
```

Assume steps: assign-initialize, compare, increment, array access, logical and

Q: Is there a “best case” and “worst case”?

• Yes, either first element is a match or last elem a match

Q: Give the best-case count as a function $T(n)$

• $T(n) \geq 10$

Q: Give the worst-case count as a function $T(n)$

• $T(n) \leq 6n + 4$

This is a (worst case) “linear time” algorithm (i.e., of the form $a \cdot n + b$)