Today

- Algorithm Analysis (cont)

Assignments

- HW 3 out
- Quiz 4 on Thurs (resizable arrays and analysis)
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:**

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

Assume primitive steps: assign/initialize, compare, increment, array access

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 $T(n)$ ... **exercise 1**
   - $T(n) \geq 10$

Q: Give the worst-case count as a function $T(n)$ ... **exercise 2**
   - $T(n) \leq 5n + 4$

This is a (worst case) “linear time” algorithm (linear equation over $n$)