CS367 Lecture 6

Tuesday 24 June 2014

Reminders:

- HW2 assigned today

Last class:

- Complexity
    - Concepts, Definitions, Assumptions, Examples
    - Growth of functions and asymptotic notation

Today:

- Complexity (cont'd)
    - Application to computer programs and analysis

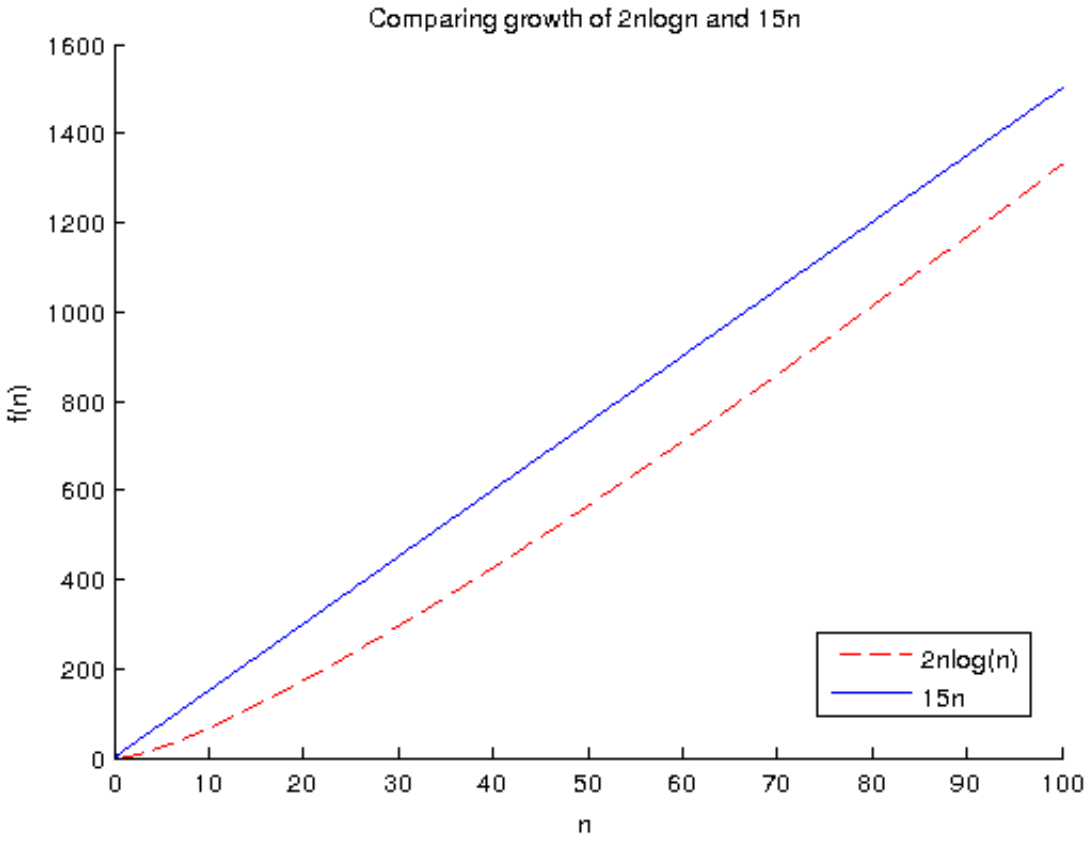
Complexity: Review

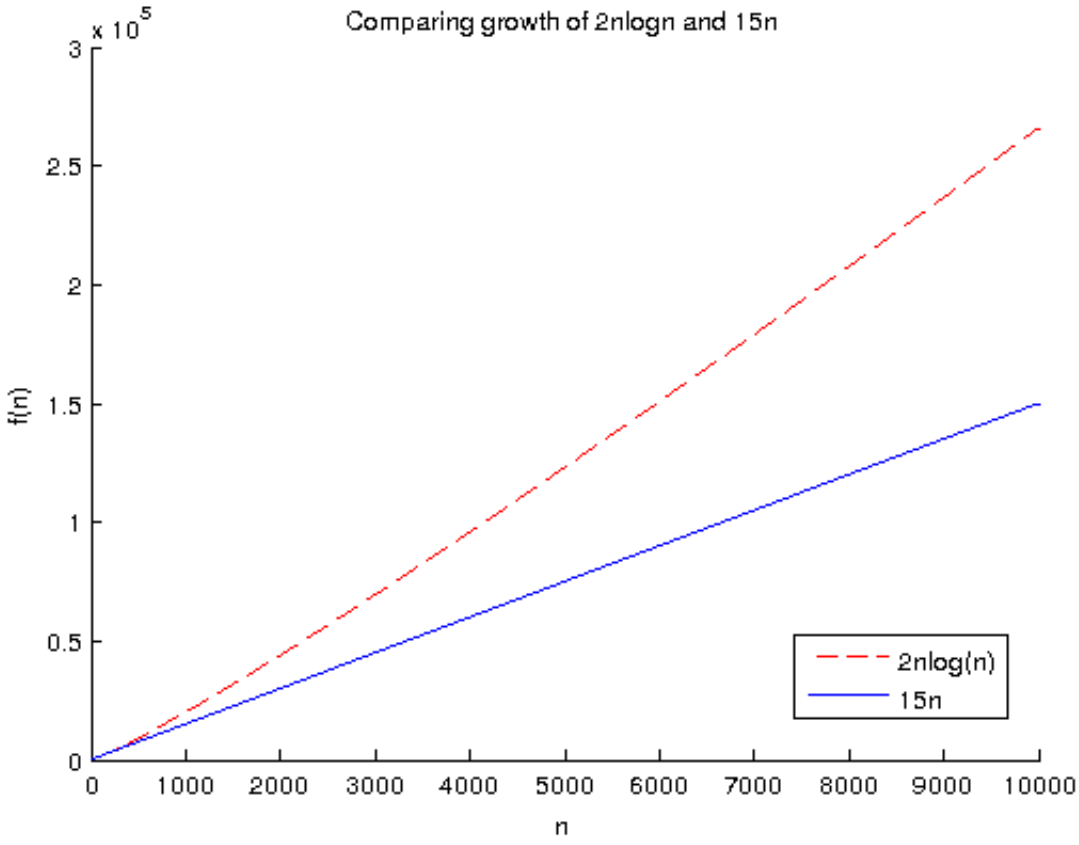
- Quantifying efficiency
- Counting # of operations, dependent on **problem size**
- Model of computation
- Complexity: Predicting how resources required grow with problem size
- Simplifying assumptions
- Higher-order terms, dominating term
- Asymptotic efficiency: Theta and Big-Oh notation

If problem size doubles and the number of operations:

- stays the same
- doubles
- ...

Complexity: Comparing *15n* and *2nlogn*





Complexity: Application to computer programs

Basic statements

Sequences of statements

```
statement1;
statement2;
...
```

if-then-else

```
if (cond) { … }
else { … }
```

Basic loops

```
for (i = 0; i < n; i++) {
    ...
}
```

Problem size?

Nested loops

```
for (i = 0; i < n; i++) {
    for (j = 0; j < m; j++) {
        ...
    }
}
```

Problem size?

Complexity: Application to computer programs (cont'd)

Loops with nested method calls

```
for (i = 0; i < n; i++) {
    f(i); // Assume O(1)
}
```

```
for (i = 0; i < n; i++) {
    f(n); // Assume O(n)
}
```

```
for (i = 0; i < n; i++) {
    f(i); Assume O(i)
}
```

Average-case Complexity

Expected Running Time

Probabilistic Analysis

Randomized Algorithms

Analyzing Complexity – Example 1

Returning *n* papers to *n* students

Problem size?

Operation(s) of interest?

Algorithm 1: Call out each name, student comes up and receives paper

Algorithm 2: Hand pile to first student, they search, take theirs, and pass on.

Best-case complexity:

Worst-case complexity:

Algorithm 3: Sort papers alphabetically, hand pile to first student, binary search, pass on.

Best-case complexity:

Worst-case complexity:

Analyzing Complexity – Example 2

Assume A and B are arrays of length *n*. For each method below, think of what the problem size depends on.

```
public void method1(int[] A, int x, int y) {
    int temp = A[x];
    A[x] = A[y];
    A[y] = temp;
}
```

```
public void method2(int[] A, int s) {
    for (int i = s; i < A.length - 1; i++) {
        if (A[i] > A[i+1]) {
            method1(A, i, i+1);
        }
    }
}
```

```
public void method3(int[] B) {
    for (int i = 0; i < B.length - 1; i++) {
        method2(B, i);
    }
}
```

Analyzing Complexity – Examples 3 and 4

```
public void method4(int Q) {
    int sum = 0, R = 1000;
    for (int i = Q; i > 0; i--) {
        for (int j = 0; j < R; j++) {
            sum += j;
        }
    }
}
```

```
public void method5(int N) {
    int tmp, arr[];

    arr = new int[N];
    for (int i = 0; i < N; i++) {
        arr[i] = N – i;
    }

    for (int i = 0; i < N - 1; i++) {
        for (int j = i; j < N - 2; j++) {
            if (arr[j] > arr[j+1]) {
                tmp = arr[j];
                arr[j] = arr[j+1];
                arr[j+1] = tmp;
            }
        }
    }
}
```

Complexity Caveats

Small problem size

Same complexity