CS 261 – Data Structures

Big-Oh Review

Measuring Performance

What is a Benchmark?

Why is it not a general characterization of your code performance?

- Program Size (N)
- Processor differences
- Load on the machine
- Compiler differences

Big Oh - Intuition

How do you find a number in a randomly organized phonebook of N names?

How much time does each comparison take?

How many comparisons?

Worst and Average case

What if we double the number of names?

Key: The time for each comparison doesn't really matter, the time to run is proportional to N for *linear search*

Big Oh - Intuition

How would you find a phone number in an alphabetically ordered phone book?

We can divide an *ordered list* in half log(N) times

Worst Case: Log (N) comparisons

What if we double N?

Time to search a list of N words = C * log N

$$C * \mathrm{Log}(2N) = C*\overset{1}{\mathrm{Log}(2)} + C*\mathrm{Log}(N)$$

$$= C + C*\mathrm{Log}(N) = C(1+\mathrm{Log}N)$$

When double list, you expect to perform just 1 extra calculation!!!

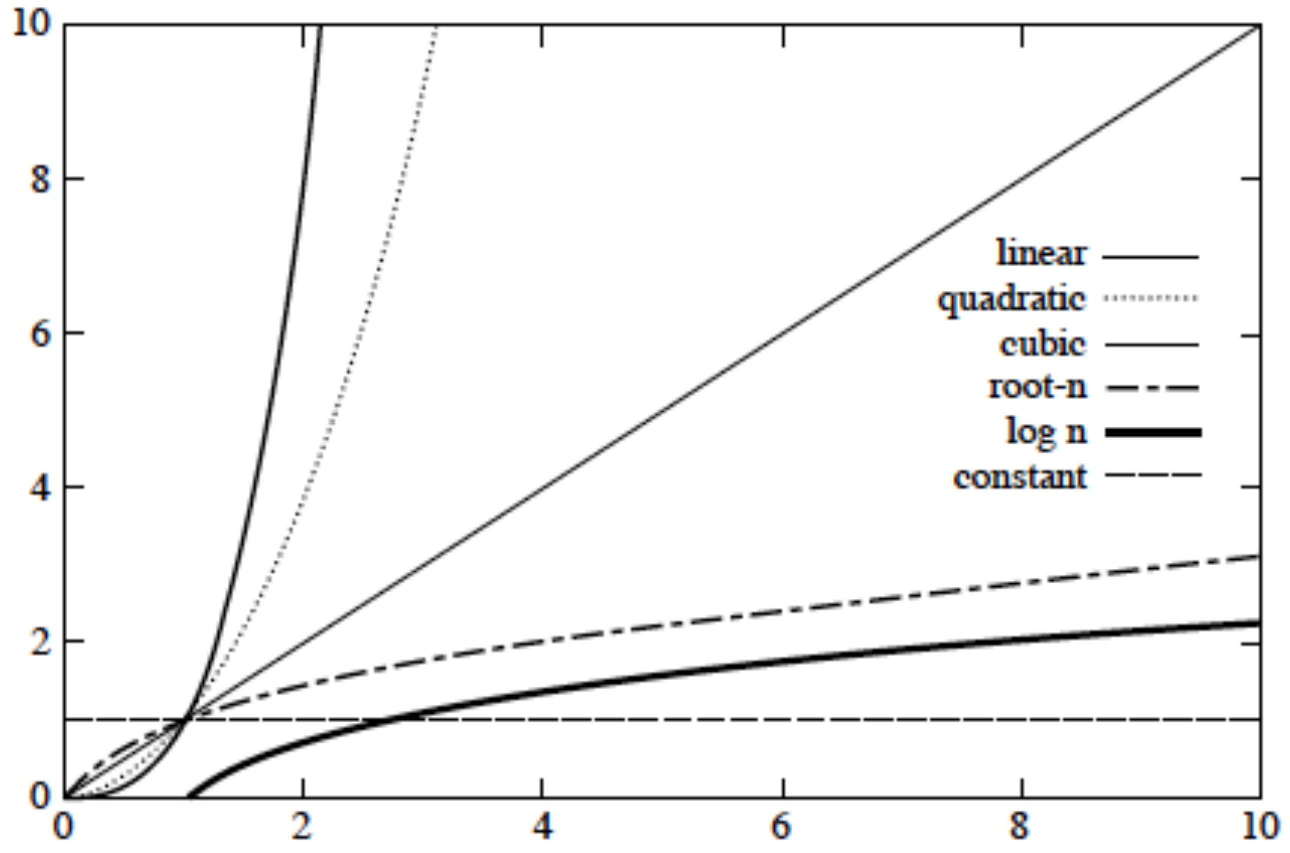
Big Oh - Intuition

When we say an algorithm is O(N) we are saying that it's execution time is ***bounded*** by some constant, C, times N

Abstracted away the details of the machine, context, etc. and focus purely on size of the input data , N

Growth Functions

We've abstracted run time as a characterization by these functions that describe the rate of growth in time as N grows

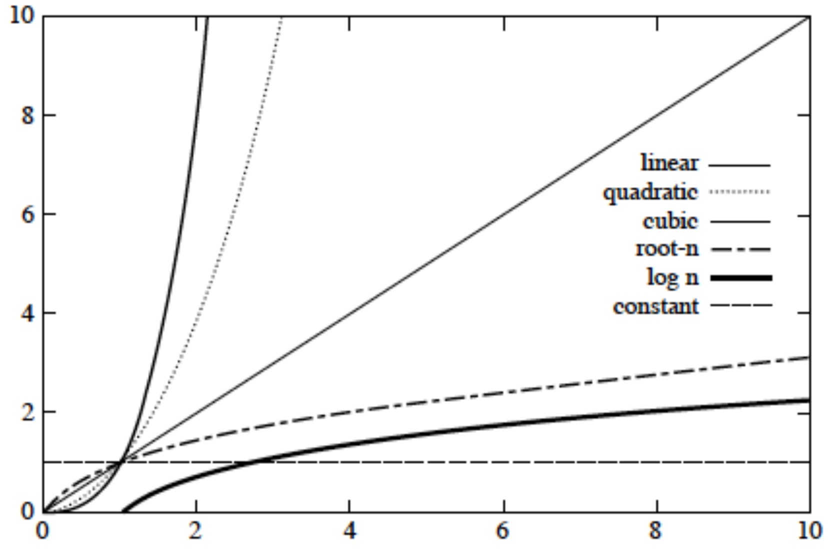


Summing Execution Times

Most code consists of several parts that must be done sequentially and thus added together to find the execution time

One function dominates another if, as input grows, one is always larger than the other no matter what constants are involved

So, when summing execution terms, throw away all but the dominant term



Summing Execution Times

What is the Big-Oh execution time of the following:

$3N^2 + 2N + 5$ operations

```
void foo () {
for(s =0; s< N; s++)
      sum = sum + 1;

for(i = 0; i < N; i++)
  for(j =0; j < N; j++)
      printf( …)
}
```

Limits of Big-Oh

Applies as values get very large

When N is small, big-Oh behavior may not be consistent

ie. $O(N^3)$ algorithm may finish before $O(\log N)$ algorithm, due to the constants. (But who really cares what happens when small!)

Estimating Wall Clock Time

If we know the clock time for some N and we know the Big-Oh, we can estimate the time for another N

$$\log(N1)/ T1 = \mathrm{Log}(N2)/X$$

Practice

Worksheet 9 - Summing Execution Times