Science 1000: Lecture #3 (Wareham):

Necessary Lies:
Asymptotic Worst-case
Time Complexity Analysis

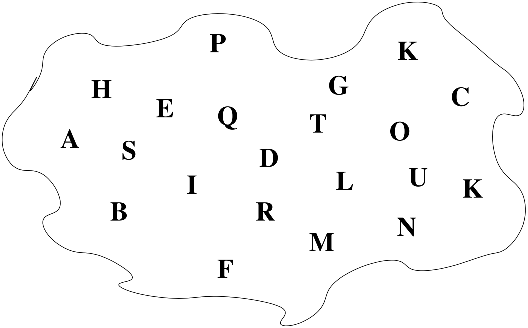
Comparing
running times is hard?
Not really.

Comparing Algorithms: What's Best?

- Best algorithm $=$ algorithm with lowest running time.
- Comparing algorithms by raw running time problematic:
  - Raw running times machine / language / OS dependent.
  - Raw running times input dependent.
  - Algorithm may not be implemented in a program.

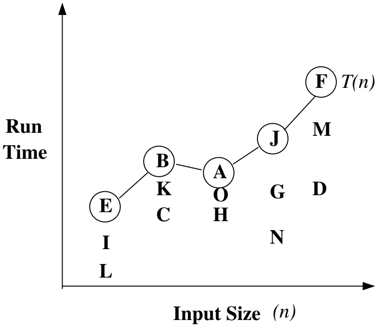
HOW DO WE MEASURE ALGORITHM RUNNING TIME?

Necessary Lie #1: Runtime Equivalence of Instructions



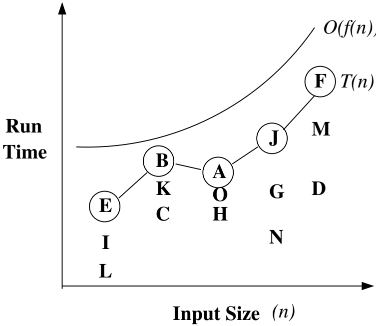
- Compute runtime on an input by counting the number of instructions executed.
- Is machine-independent (raw abstract runtime).

Necessary Lie #2:
Worst-Case Runtime Summary



- Group inputs by input size; summarize each size by largest runtime for that size.
- Is input-independent (worst-case time complexity).

Necessary Lie #3: Asymptotic Smoothing



- Reduce time complexity function to largest term.
- Is simple (asymptotic worst-case time complexity).

Deriving Worst-Case Time Complexities

- If already have time complexity, select largest term, e.g.,

$$
\begin{aligned}
2\log n + 4 \quad &\Rightarrow \quad O(\log n) \\
3n^2 + 1000n + 13 \quad &\Rightarrow \quad O(n^2) \\
12n^4 + 5n^2 + 900 \quad &\Rightarrow \quad O(n^4) \\
(3 \times 2^n) + 900n^{50} + 57 \quad &\Rightarrow \quad O(2^n)
\end{aligned}
$$

Deriving Worst-Case Time Complexities (Cont'd)

- Otherwise, multiply out "deepest" loop-chain in algorithm, *e.g.*, $n \times n = O(n^2)$ time for List Sort.

```
for i = 1 to n - 1 do
      min_pos = i
      for scan = i + 1 to n do
            if (L[scan] < L[min_pos]) then
                  min_pos = scan
      temp = L[min_pos]
      L[min_pos] = L[i]
      L[i] = temp
```

Time Complexity Magnitudes

$O(\log n)$	Logarithmic Time	(Binary Search)
$O(n)$	Linear Time	(Linear Search)
$O(n^2)$	Quadratic Time	(List Sort)
$O(2^n)$	Exponential Time	(Bin Packing)

Polynomial Time $= O(n^c)$ time for constant c

Table of Doom (1 Gigaflop/s Version)

Input Size (n)	Time Complexity: B-Search ($\log_2 n$)	L-Search (n)	Sort (n^2)	MST (n^3)	BP-E (2^n)
10	< 1 second	< 1 second	< 1 second	< 1 second	< 1 second
50	< 1 second	< 1 second	< 1 second	< 1 second	13 days
100	< 1 second	< 1 second	< 1 second	< 1 second	4×10^{13} years
1000	< 1 second	< 1 second	< 1 second	1 second	4×10^{284} years
one million	< 1 second	< 1 second	2 minutes	30 years	–
300 million	< 1 second	< 1 second	10 days	9×10^5 years	–
five billion	< 1 second	5 seconds	8 centuries	4×10^{12} years	–

Science 1000: Lecture #3 (Wareham):

Necessary Lies:
Asymptotic Worst-case
Time Complexity Analysis

Comparing
running times is hard?
Not really.