CSC 225

Algorithms and Data Structures I

Rich Little
rlittle@uvic.ca
ECS 516

Logarithms and Exponential Functions

- Review properties of Logarithms and exponents

$$\log_b a = c \ \text{ if } \ a = b^c$$

$$\log ac = \log a + \log c$$

$$\log a / c = \log a - \log c$$

$$\log a^c = c \log a$$

$$\log_b a = \frac{\log_c a}{\log_c b}$$

$$b^{\log_c a} = a^{\log_c b}$$

$$(b^a)^c = b^{ac}$$

$$b^a b^c = b^{a+c}$$

$$b^a / b^c = b^{a-c}$$

Asymptotic Notation

- Evaluating running time in detail as for `arrayMax` and `recursiveMax` is cumbersome
- Fortunately, there are asymptotic notations which allow us to characterize the main factors affecting an algorithm's running time without going into detail
- A good notation for large inputs

Name	Notation	Analogy
Big-Oh	$f(n) \in O(g(n))$	"$f(n) \leq g(n)$"
Little-Oh	$f(n) \in o(g(n))$	"$f(n) < g(n)$"
Big-Theta	$f(n) \in \Theta(g(n))$	"$f(n) = g(n)$"
Big-Omega	$f(n) \in \Omega(g(n))$	"$f(n) \geq g(n)$"
Little-Omega	$f(n) \in \omega(g(n))$	"$f(n) > g(n)$"

Formal Definition of Big-Oh Notation

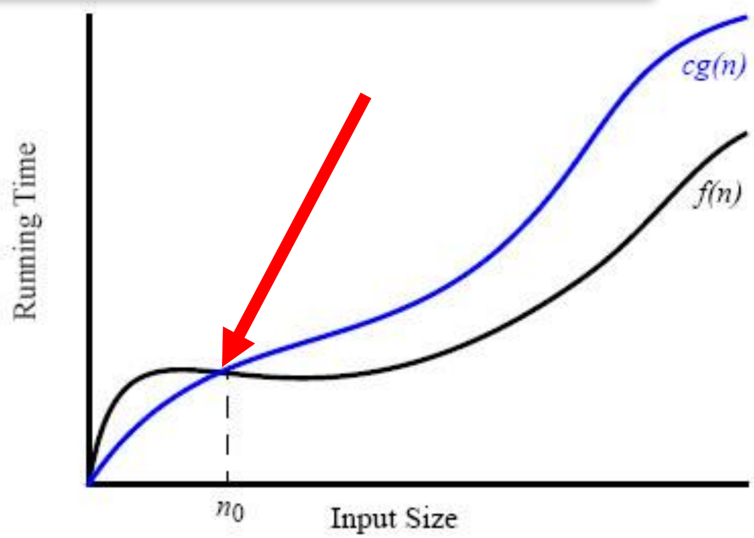
Let $f\colon \mathbb{N} \to \mathbb{R}$ and $g\colon \mathbb{N} \to \mathbb{R}$. $f(n)$ is $O(g(n))$ if and only if **there exists** a real constant $c > 0$ and an integer constant $n_0 > 0$ such that $f(n) \le cg(n)$ for all $n \ge n_0$.

$\mathbb{N}$: non-negative integers

$\mathbb{R}$: real numbers

$f(n) = O($

- We say
  - $f(n)$ is *order* $g(n)$
  - $f(n)$ is *big-Oh* of $g(n)$
  - $f(n) \in O(g(n))$
- Visually, this says that the $f(n)$ curve must eventually fit under the $cg(n)$ curve.



Big-Oh: Examples $f(n)$ is $O(?)$

1. $f(n) = 4n + 20n^4 + 117$

$\in O(n^4)$

$1000000000\, n^3$

2. $f(n) = 1083$ $\in O(1)$

$\exists c, n_0 > 0$, s.t. Let $n_0 = 1$, $c = 1083$

$\boxed{1083 \leq c \cdot 1 \;\; \forall n \geq n_0}$

3. $f(n) = 3\log n$

$\in O(\log n)$

$3\log n \leq 3 \cdot \log n \;\; \forall n \geq 1$

4. $f(n) = 3\log n + \log\log n$ $\in O(\log n)$

w.t.s. $\exists c, n_0 >$ s.t.

$7\log n + \log(\log n) \leq c \cdot \log n$

5. $f(n) = 2^{17}$ $\in O(1)$

$c = 2^{17}$; 1

6. $f(n) = 33/n$

$O(1)$? $O(\frac{1}{n})$?

$\frac{33}{n} \leq 33 \cdot \frac{1}{n}$

7. $f(n) = 2^{\log 2n}$

$O(1)$, $O(n)$

$2^{\log 2n} = 2n$

8. $f(n) = 1^n$

$O(\log n)$, $O(1)$

Theorem

- **R1:** If $d(n)$ is $O(f(n))$, then $ad(n)$ is $O(f(n))$, for any constant $a > 0$
- **R2:** If $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then $d(n) + e(n)$ is $O(f(n) + g(n))$
- **R3:** If $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then $d(n)e(n)$ is $O(f(n)g(n))$
- **R4:** If $d(n)$ is $O(f(n))$ and $f(n)$ is $O(g(n))$, then $d(n)$ is $O(g(n))$
- **R5:** If $f(n) = a_0 + a_1 n + \ldots + a_d n^d$, d and a_k are constants, then $f(n)$ is $O(n^d)$

  $a_d \neq 0$
- **R6:** n^x is $O(a^n)$ for any fixed $x > 0$ and $a > 1$
- **R7:** $\log n^x$ is $O(\log n)$ for any fixed $x > 0$
- **R8:** $\log^x n$ is $O(n^y)$ for any fixed constants $x > 0$ and $y > 0$

Proof of R1

R1: If $d(n)$ is $O(f(n))$, then $ad(n)$ is $O(f(n))$, for any constant $a > 0$

Let $d(n) \in O(f(n))$ and let $a > 0$.

w.t.s $\exists k, n_1 > 0$ s.t. $ad(n) \le k f(n)$ $\forall n \ge n_1$

We know $\exists c, n_0 > 0$, s.t.

$$d(n) \le c f(n) \quad \forall n \ge n_0$$

multiply by a: $\Rightarrow a \cdot d(n) \le a \cdot c f(n) \quad \forall n \ge n_0$

$$k = ac > 0, \; n_1 = n_0 > 0$$

$$\therefore \; ad(n) \in O(f(n))$$

Proof of R2

R2: If $d(n)$ is $O(f(n))$ and $e(n)$ is $O(g(n))$, then $d(n) + e(n)$ is $O(f(n) + g(n))$

W.T.S.: $\exists\, c, n_0 > 0$ such that $d(n) + e(n) \le c(f(n) + g(n))$

$\forall\, n \ge n_0 > 0$.

We know $\exists\, c_1, c_2, n_1, n_2 > 0$, such that

$d(n) \le c_1 f(n)$ for all $n \ge n_1$

$e(n) \le c_2 g(n)$ for all $n \ge n_2$

$\forall\, n \ge$ max

$d(n) + e(n) \le c_1 f(n) + c_2 g(n)$, let $c_0 = \max(c_1, c$

$\le c_0 f(n) + c_0 g(n) = c_0(f(n) + g(n))$

Names of Most Common Big Oh Functions

- Constant $O(1)$
- Logarithmic $O(\log n)$
- Linear $O(n)$
- Linearithmic $O(n\log n)$
- Quadratic $O(n^2)$
- Polynomial $O(n^k)$, k is a constant

$\underbrace{n \cdot (n-1) \cdots 1}_{n}$

$\underbrace{n \cdot n \cdot n \cdots n}_{n}$

- Exponential $O(2^n)$
- Exponential $O(a^n)$, a is a constant and $a > 1$
- Factorial $O(n!)$ $O(n^n)$

Quiz
Which statement is True?

1. 2^n is $O(n!)$?

Show that $\exists\ c > 0, n_0 > 0$
such that $2^n \le c\,n!$
$\forall\ n \ge n_0$

$\underbrace{2 \cdot 2 \cdot 2 \cdots 2}_{n}$

$\underbrace{n \cdot (n-1) \cdot (n-2) \cdots 2}_{n}$

2. $n!$ is $O(2^n)$?

Quiz: 2^n is $O(n!)$ is true

Proof. Let $n_0 = 4$ and $c = 1$ and show that $2^n < n!$ for all $n \geq n_0$ using induction.

we know that $2 < k+1$ when $k \geq 4$

B.C.: Let $n=4$, $c=1$

$2^4 = 16$, $4! = 24$ so $2^4 < 4!$

I.H.: Let $n = k \geq 4$, assume $2^k < k!$

I.S.: Let $n = k+1$, w.t.s. $2^{k+1} < (k+1)!$

$2^{k+1} = 2 \cdot 2^k < 2 \cdot k!$ by I.H.

$< (k+1) \cdot k! = (k+1)!$

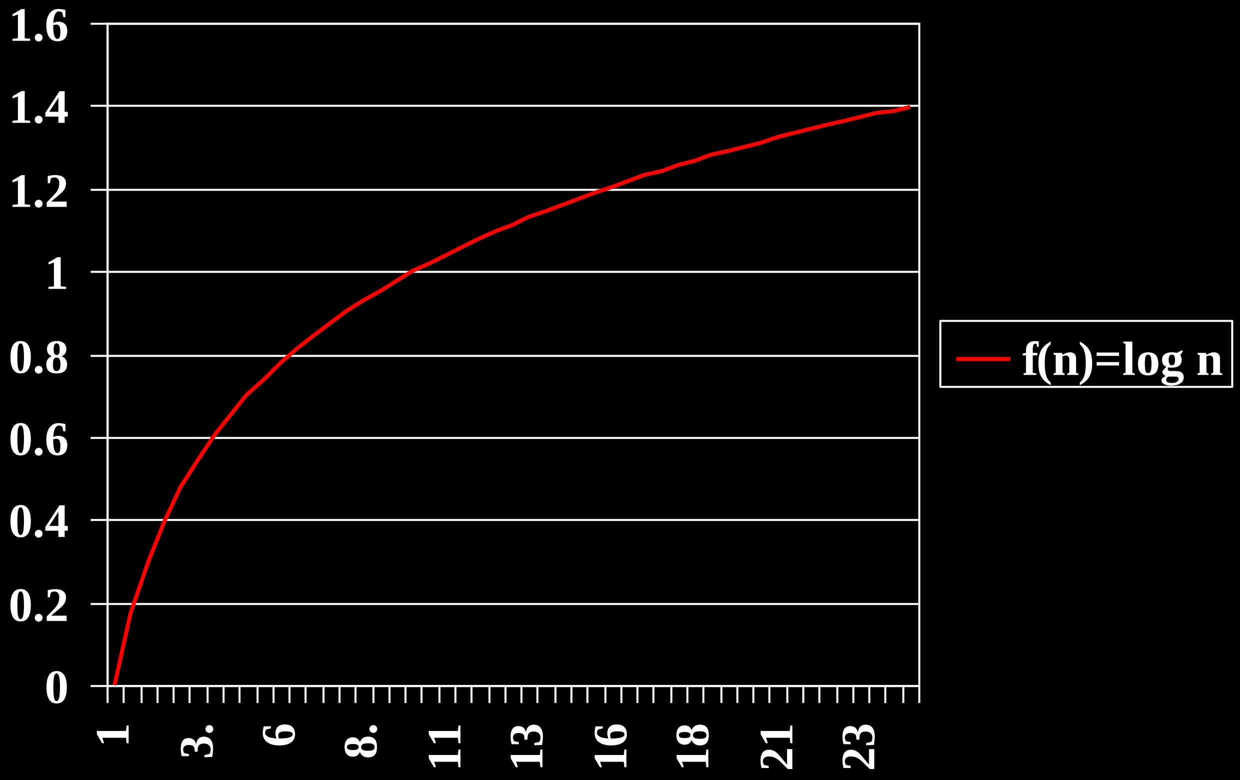
Most Common Functions in Algorithm Analysis Ordered by Growth

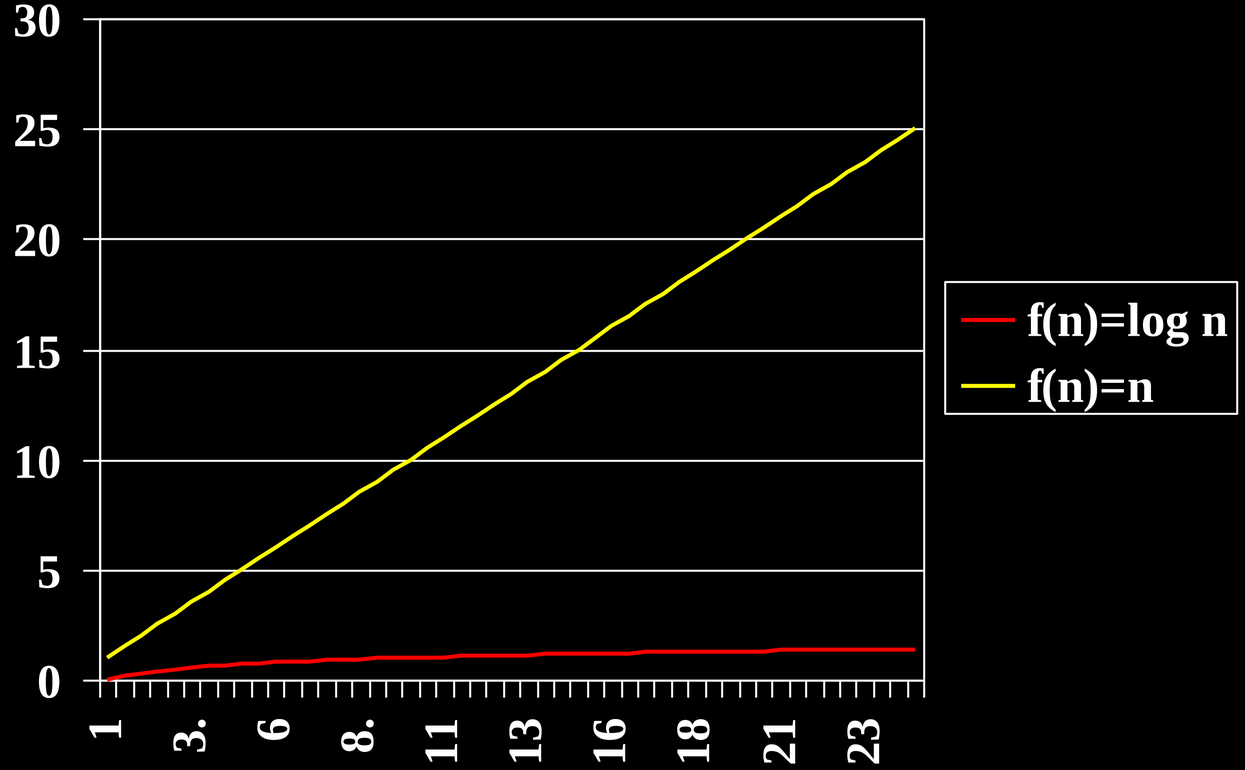
$1 \quad \log\log n \quad \log n \quad \sqrt{n}$ $\quad n^{1/2}$

$n \quad n\log n \quad n^2 \quad n^{2.31}$

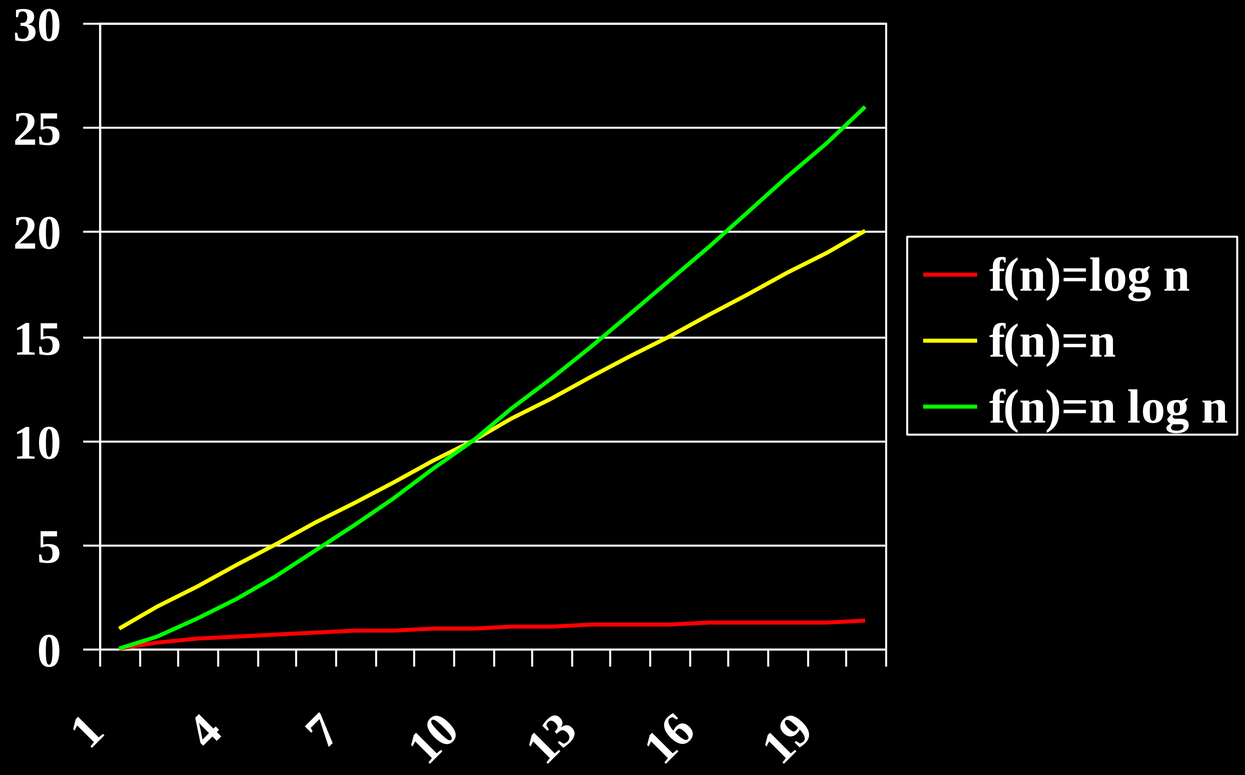
$n^3 \quad n^k, \text{for } k > 3$

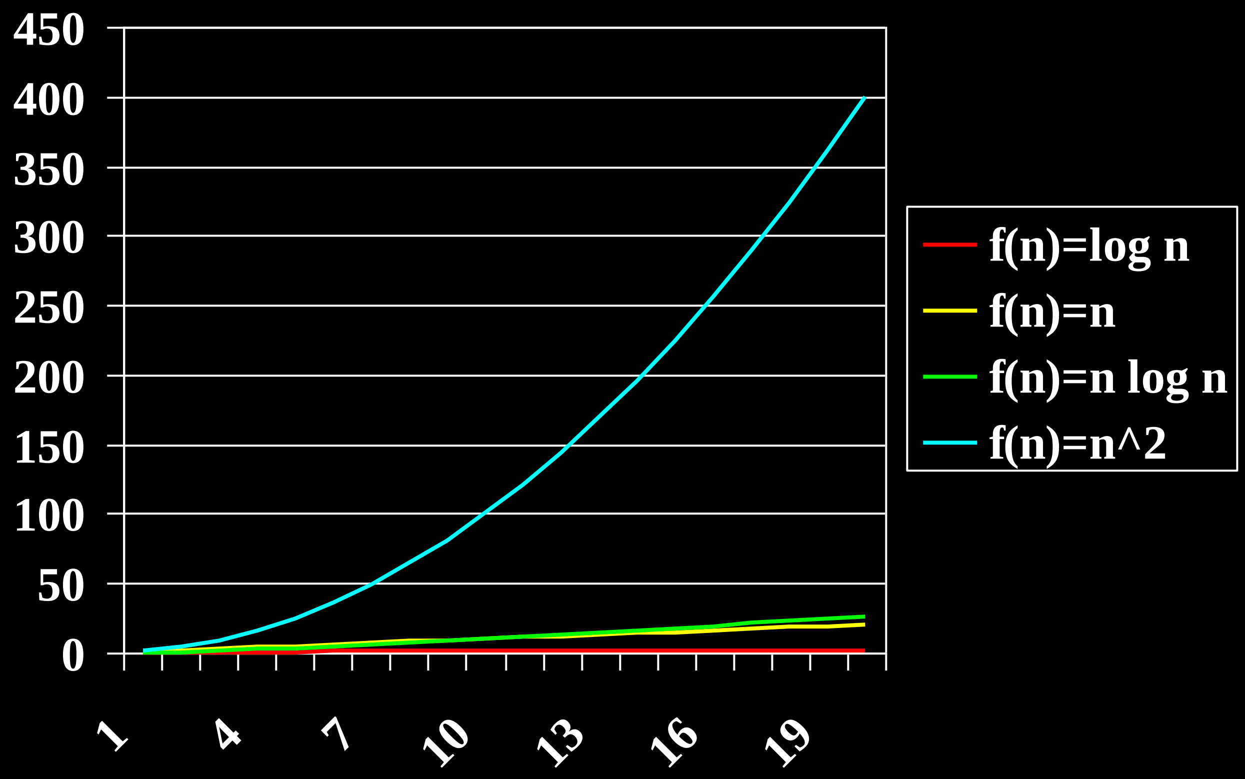
$2^n \quad 3^n \quad n! \quad n^n$


1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
1
3.
6
8.
11
13
16
18
21
23
f(n)=log n

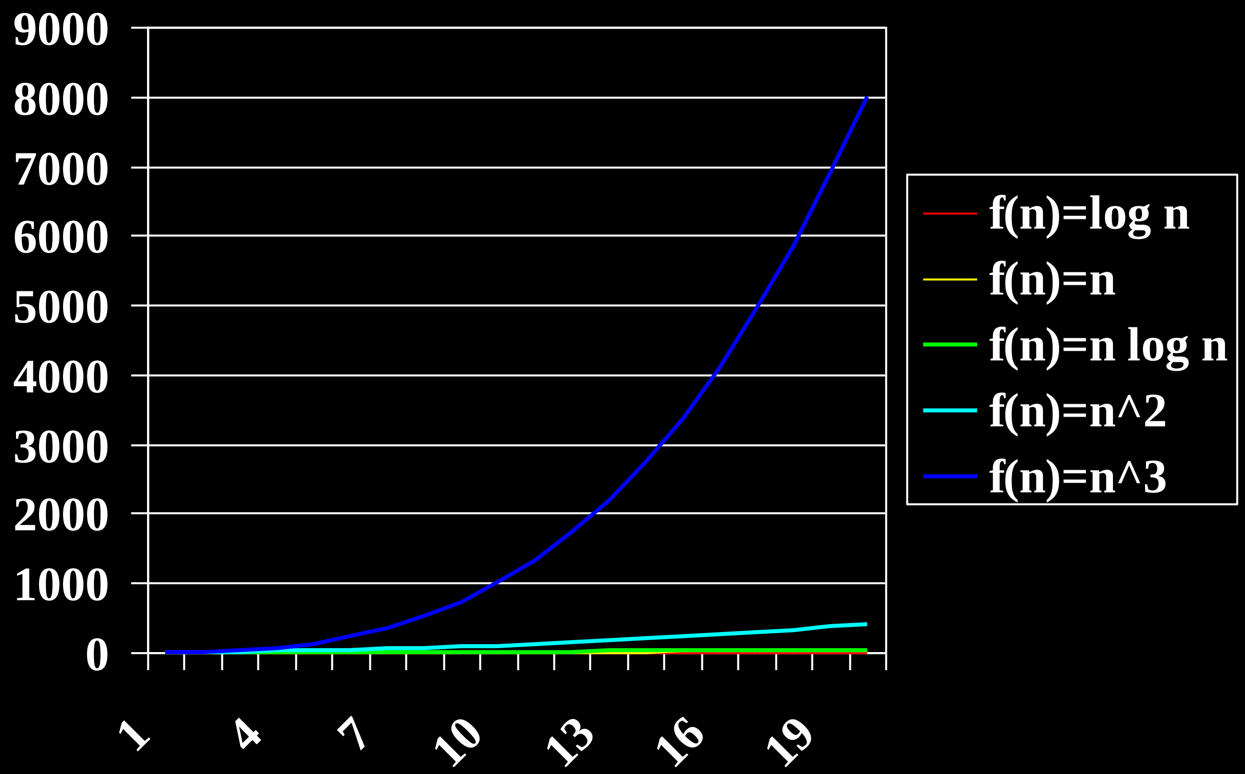


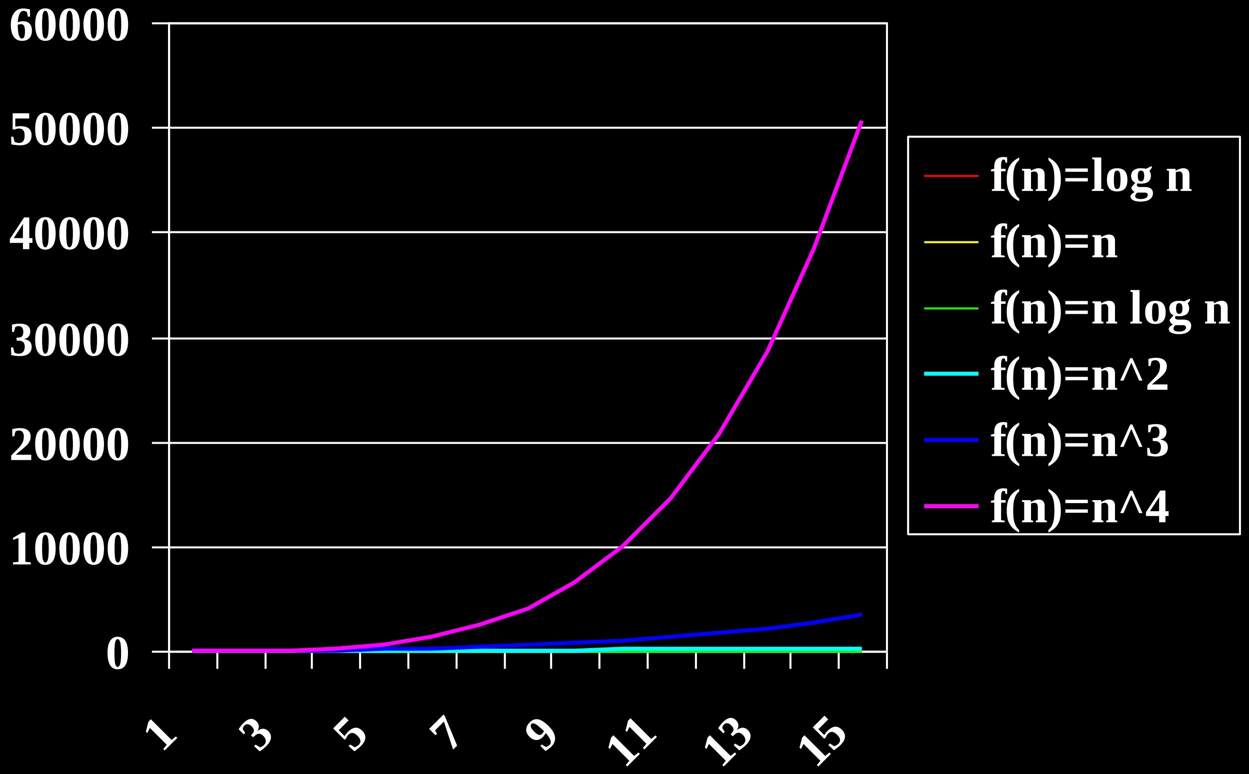
30
25
20
15
10
5
0
1
3.
6
8.
11
13
16
18
21
23
f(n)=log n
f(n)=n

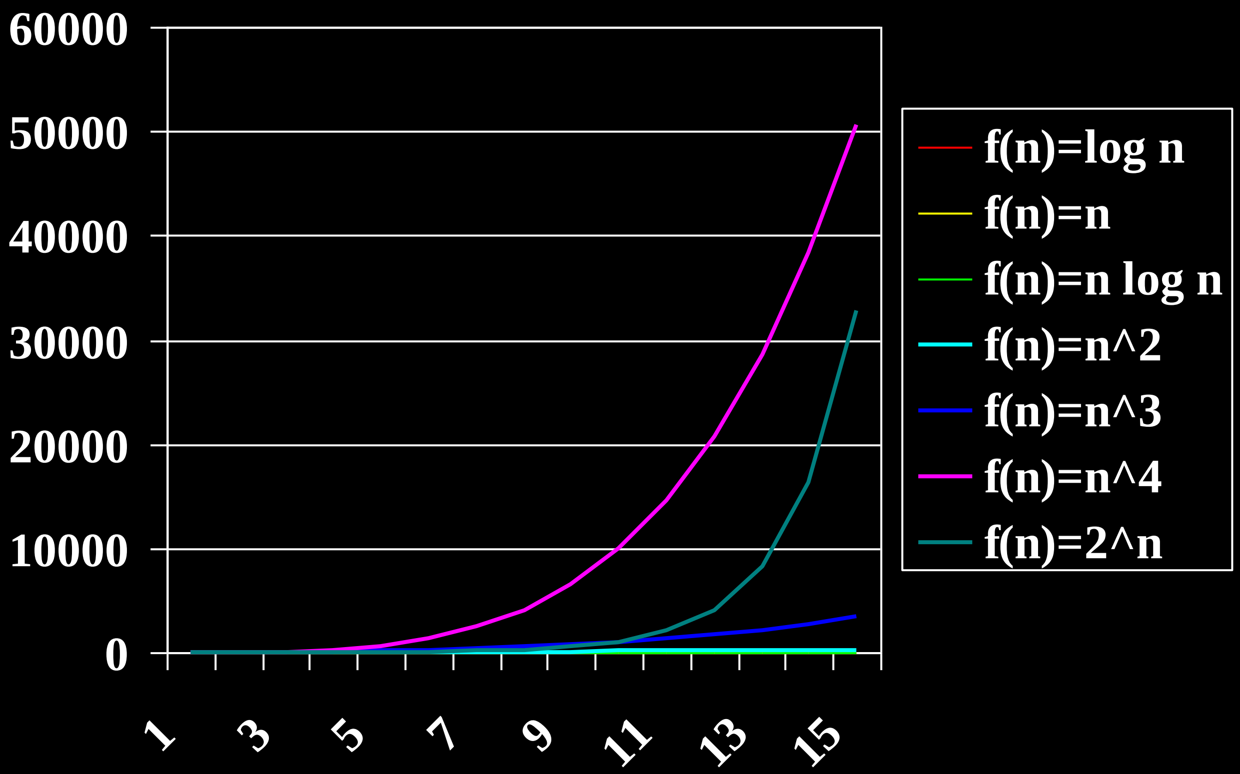

30
25
20
15
10
5
0
1
4
7
10
13
16
19
f(n)=log n
f(n)=n
f(n)=n log n



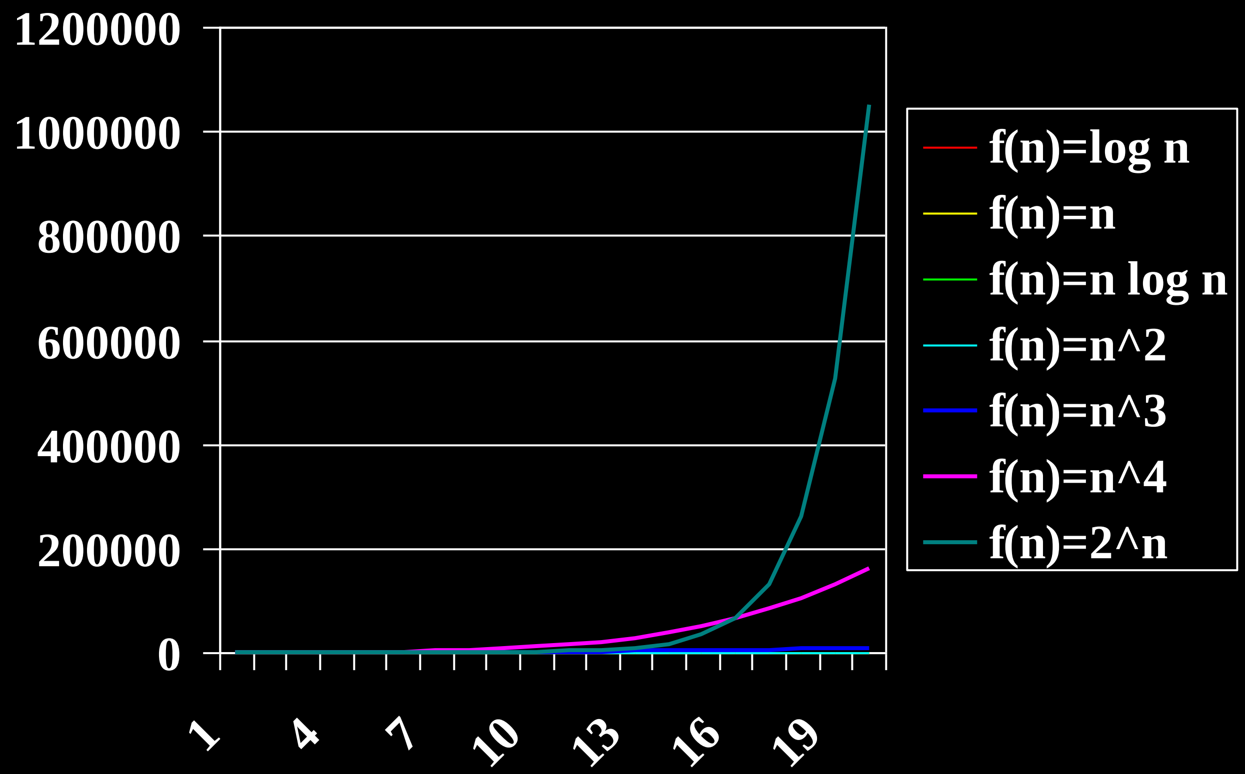
450
400
350
300
250
200
150
100
50
0
1
4
7
10
13
16
19
f(n)=log n
f(n)=n
f(n)=n log n
f(n)=n^2


9000
8000
7000
6000
5000
4000
3000
2000
1000
0
1
4
7
10
13
16
19
f(n)=log n
f(n)=n
f(n)=n log n
f(n)=n^2
f(n)=n^3


60000
50000
40000
30000
20000
10000
0
1
3
5
7
9
11
13
15
f(n)=log n
f(n)=n
f(n)=n log n
f(n)=n^2
f(n)=n^3
f(n)=n^4



60000
50000
40000
30000
20000
10000
0
1
3
5
7
9
11
13
15
f(n)=log n
f(n)=n
f(n)=n log n
f(n)=n^2
f(n)=n^3
f(n)=n^4
f(n)=2^n


1200000
1000000
800000
600000
400000
200000
0
1
4
7
10
13
16
19
f(n)=log n
f(n)=n
f(n)=n log n
f(n)=n^2
f(n)=n^3
f(n)=n^4
f(n)=2^n

Functions Ordered by Growth and Rate

n	log n	n	n log n	n^2	n^3	2^n	n!
10	3.3	10	33	10^2	10^3	10^3	10^6
10^2	6.6	10^2	6.6×10^2	10^4	10^6	10^{30}	10^{158}
10^3	10	10^3	10×10^3	10^6	10^9		
10^4	13	10^4	13×10^4	10^8	10^{12}		
10^5	17	10^5	17×10^5	10^{10}	10^{15}		
10^6	20	10^6	20×10^6	10^{12}	10^{18}		

Assume a computer executing 10^{12} operations per second.
To execute 2^{100} operations takes 4×10^{10} years.
To execute 100! operations takes much longer still.

Functions Ordered by Growth and Rate

- $\log n$
- $\log^2 n$
- $\sqrt{n}$
- n
- $n \log n$
- n^2
- n^3

P = class of polynomial time algorithms

- 2^n

NP = class of *nondeterministic* polynomial time algorithms

A million (US-) dollar question

http://www.claymath.org/millennium/P_vs_NP/

- P = NP?
- Obviously P $\subseteq$ NP.
- There is a bunch of problems (so called NP-complete problems) for which one assumes that none of those can be solved in polynomial time.
- Examples: Graph coloring, Independent Set, Generalized 15-Puzzle
- Widely assumed: P $\neq$ NP