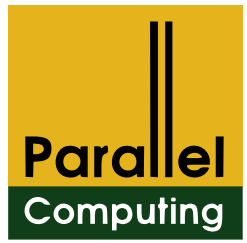
Parallel Computing: Reminders, useful facts and definitions

Jesper Larsson Träff

Research Group Parallel Computing
Institute of Information Systems, Faculty of Informatics
TU Wien, Wien, Austria

August 2013



Standard background textbooks


Bryant, R. E., and O'Hallaron, D. R.
Computer Systems. A Programmer's Perspective, second ed.
Prentice-Hall, 2011.


Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C.
Introduction to Algorithms, third ed.
MIT Press, 2009.


Dongarra, J., Foster, I., Fox, G., Gropp, W., Kennedy, K., Torczon, L., and White, A., Eds.
Sourcebook of Parallel Computing.
Morgan Kaufmann Publishers, 2993.


Grama, A., Karypis, G., Kumar, V., and Gupta, A.
Introduction to Parallel Computing, second ed.
Addison-Wesley, 2003.


Hager, G., and Wellein, G.
Introduction to High Performance Computing for Scientists and Engineers.
CRC Press, 2011.


Herlihy, M., and Shavit, N.
The Art of Multiprocessor Programming, revised 1st ed.
Morgan Kaufmann Publishers, 2012.


JáJá, J.
An Introduction to Parallel Algorithms.
Addison-Wesley, 1992.


McCool, M., Robison, A. D., and Reinders, J.
Structured Parallel Programming.
Morgan Kaufmann Publishers, 2012.


Quinn, M. J.
Parallel Programming in C with MPI and OpenMP.
McGraw-Hill, 2003.


Rauber, T., and Rünger, G.
Parallel Programming for Multicore and Cluster Systems.
Springer-Verlag, 2010.

Asymptotic bounds, definitions

Let $f(n)$ and $g(n)$ be (positive, real) functions.

Upper bound: A function $g(n)$ is $O(f(n))$ (read: "Oh", or "big-O") if there exist a constant $c > 0$ and a constant n_0, such that for all $n \geqslant n_0$ it holds that $0 \leqslant g(n) \leqslant cf(n)$.

Lower bound: A function $g(n)$ is $\Omega(f(n))$ (read: "big-Omega") if there exist a constant $c > 0$ and a constant n_0, such that for all $n \geqslant n_0$ it holds that $g(n) \geqslant cf(n) \geqslant 0$.

Upper and lower bound: A function $g(n)$ is $\Theta(f(n))$ (read: "Theta") if there exist constants $c > 0$ and $d > 0$, and a constant n_0, such that for all $n \geqslant n_0$ it holds that $0 \leqslant cf(n) \leqslant g(n) \leqslant df(n)$.

Lemma

A function $g(n)$ is $O(f(n))$ iff $f(n)$ is $\Omega(g(n))$

Lemma

A function $g(n)$ is $\Theta(f(n))$ iff $g(n)$ is $O(f(n))$ and $\Omega(f(n))$

Lemma

A function $g(n)$ is $\Theta(f(n))$ iff $f(n)$ is $\Theta(g(n))$

Strict upper bound: A function $g(n)$ is $o(f(n))$ (read: "little-o") if for any constant $c > 0$ there exists a constant n_0, such that for all $n \geqslant n_0$ it holds that $0 \leqslant g(n) < cf(n)$.

Strict lower bound: A function $g(n)$ is $\omega(f(n))$ (read: "little-omega") if for any constant $c > 0$ there exists a constant n_0, such that for all $n \geqslant n_0$ it holds that $g(n) > cf(n) \geqslant 0$.

Lemma

A function $g(n)$ is $o(f(n))$ iff $f(n)$ is $\omega(g(n))$

See [2] or other good algorithms textbook for more properties (e.g., transitivity, reflexivity) and examples.

Notations for logarithmic functions

All logarithm functions are of the same order, recall that $\log_b n = \frac{\ln n}{\ln b}$:

Lemma

$\ln n$ is $\Theta(\log_b n)$ for $b > 0$

Definition

- $\log^k n$ means $(\log n)^k$
- $\log \log n$ means $\log(\log n)$ and

$$\log^{(k)} n = \underbrace{\log(\ldots(\log n)\ldots)}_{\text{ktimes}}$$

Definition

Iterated logarithm:

$$\log^* x \quad = \quad \begin{cases} 0 & \text{if } x \leqslant 1 \\ 1 + \log^*(\log x) & \text{if } x > 1 \end{cases}$$

Growth of functions: logarithms, polynomials, exponentials

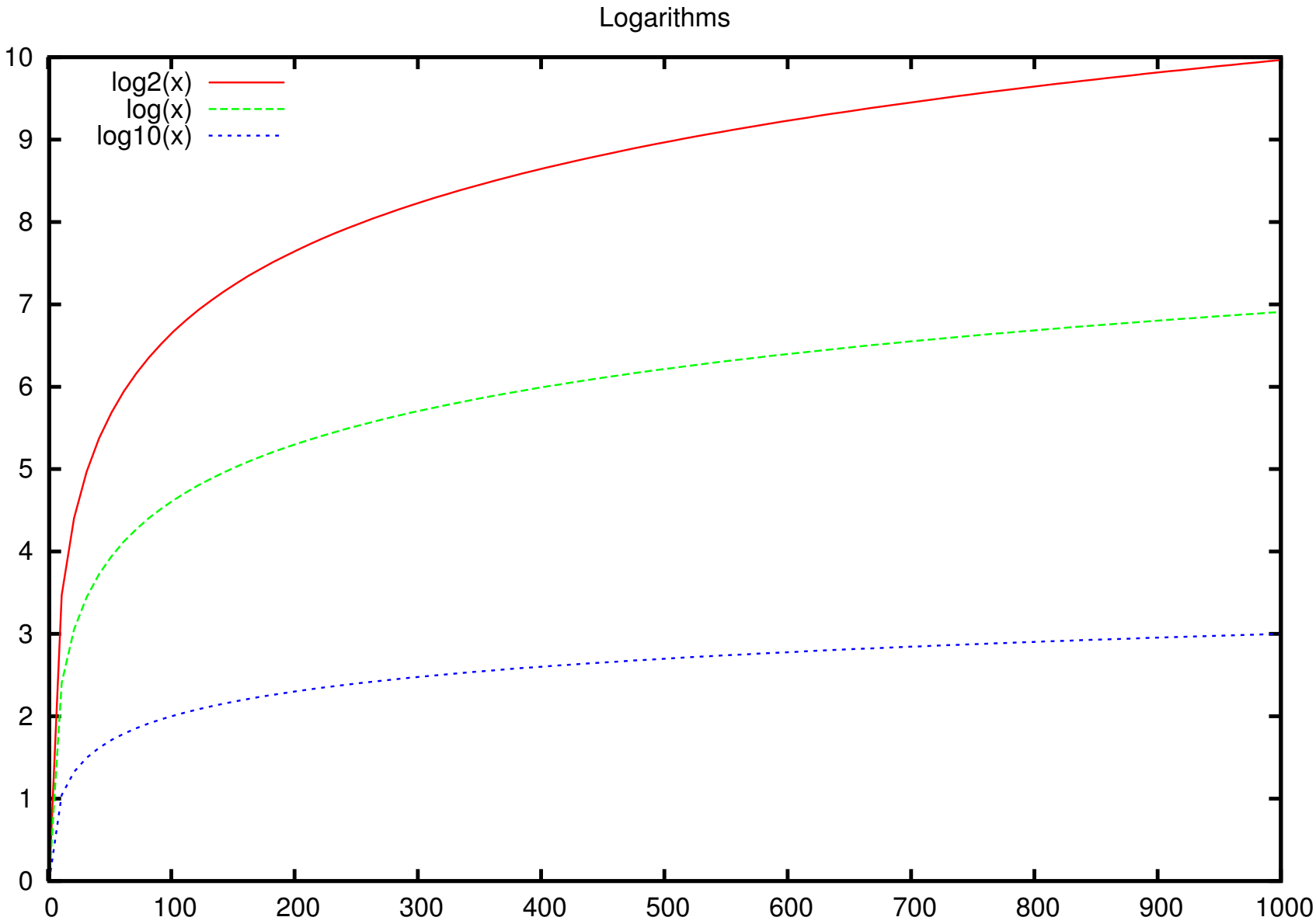
Lemma

Any poly-logarithmic function grows slower than any positive polynomial, that is $\log^b n$ is $o(n^a)$ for any $a > 0, b > 0$.

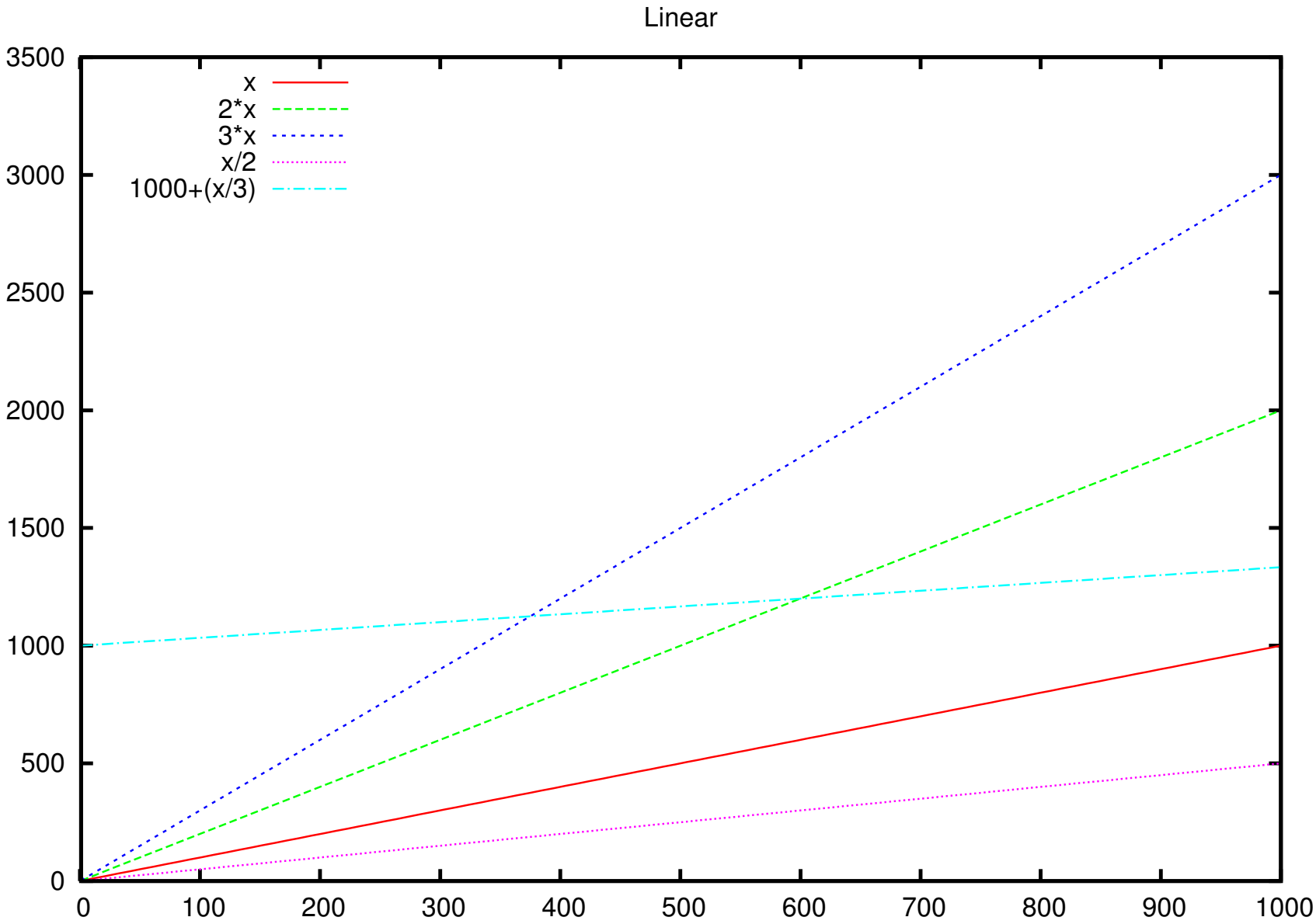
Lemma

Any polynomial grows slower than any positive exponential function, that is n^b is $o(a^n)$ for any $b > 0, a > 1$.

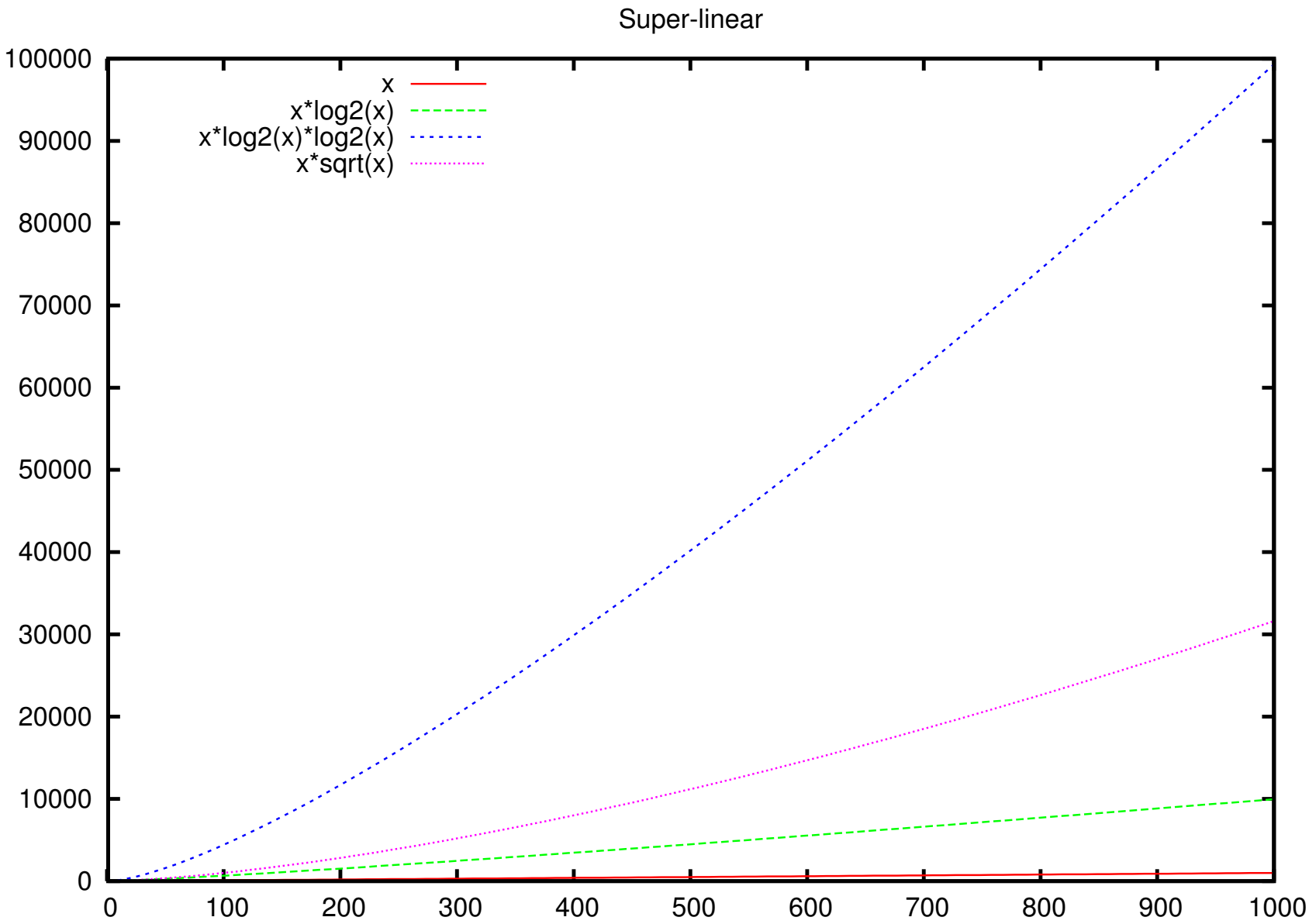
Plots, some standard functions



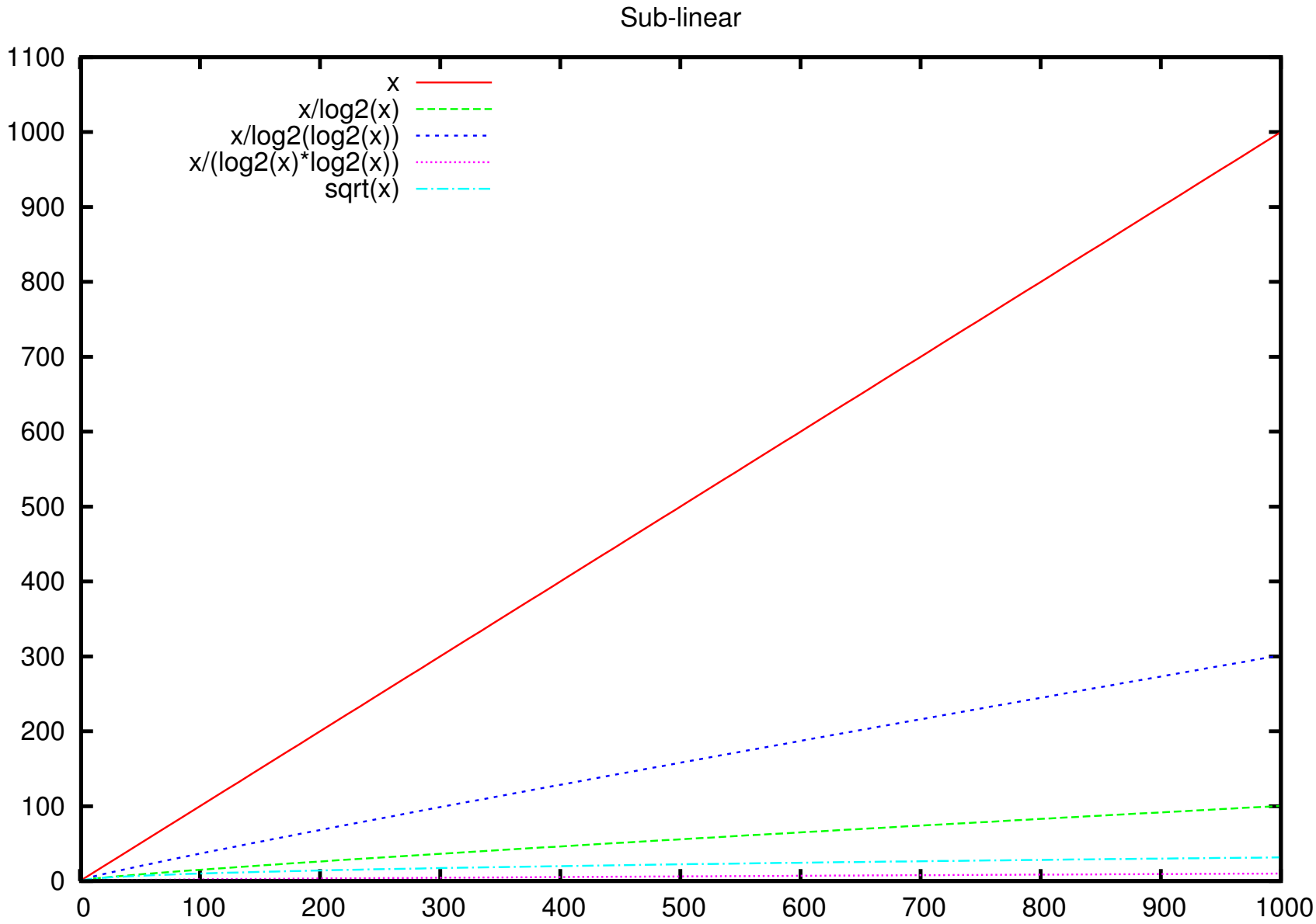
Plots, some standard functions



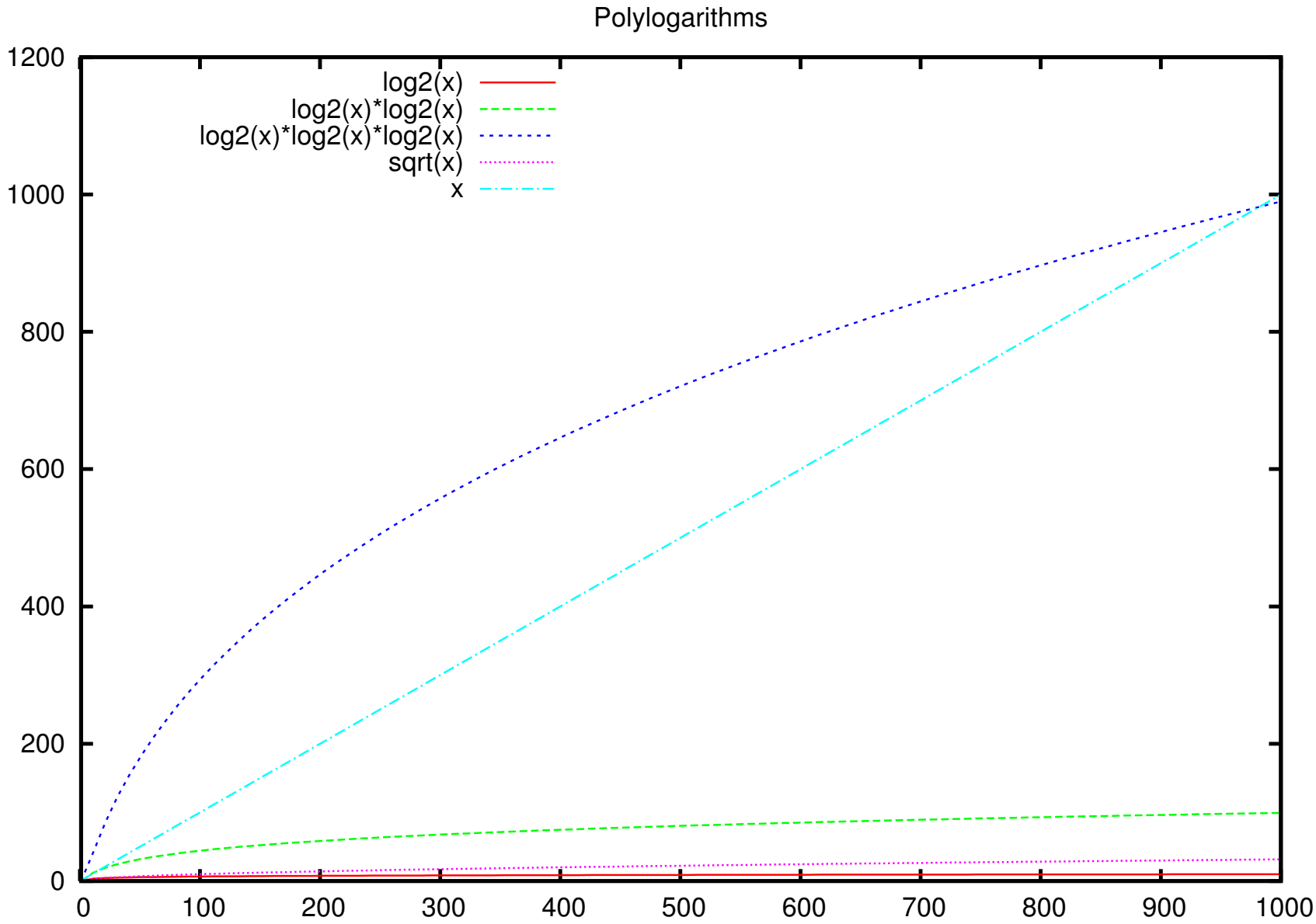
Plots, some standard functions



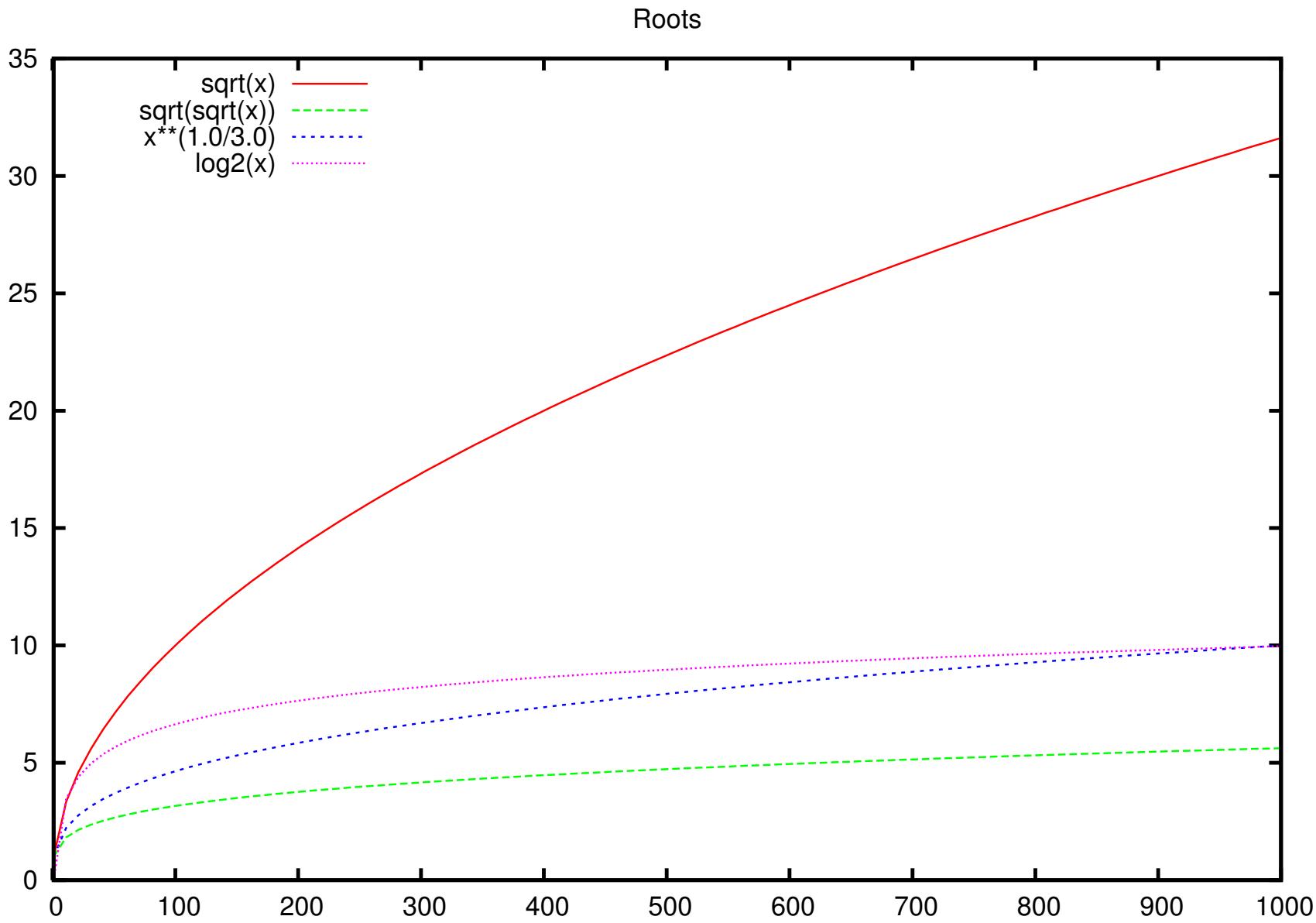
Plots, some standard functions



Plots, some standard functions



Plots, some standard functions



Series and sums

An arithmetic fact (quotient series), for all $q \neq 1$

$$\sum_{i=0}^{n} aq^i = a\frac{1-q^{n+1}}{1-q}$$

For $q = 1$, the sum is $(n+1)a$.
Example: $1 + 1/2 + 1/4 + 1/8 = 2(15/16) = 2 - 1/8$
$(a = 1, q = 1/2)$

An analytic fact (convergence), for $|q| < 1$

$$\sum_{i=0}^{\infty} aq^i = a\frac{1}{1-q}$$

Example: $1 + 1/2 + 1/4 + 1/8 + 1/16 + \ldots = 2$

A sum of the form $\sum_{i=1}^{n}(a_i - a_{i+1}) = a_1 - a_{n+1}$ is called telescoping.

Example:

$$\sum_{i=1}^{n} \frac{1}{i(i+1)} \quad = \quad \sum_{i=1}^{n} \left(\frac{1}{i} - \frac{1}{i+1}\right) \quad = \quad 1 - 1/(n+1)$$

Solving recurrences, Master Theorem (I)

Let $a \geqslant 1, b > 1$ be constants, $f(n)$ a function, and let $T(n)$ be defined by the recurrence

$$T(n) \quad = \quad aT(n/b) + f(n)$$

Then $T(n)$ is

1. $\Theta(n^{\log_b a})$, if $f(n)$ is $O(n^{\log_b a - \epsilon})$ for some $\epsilon > 0$
2. $\Theta(n^{\log_b a} \log n)$, if $f(n)$ is $\Theta(n^{\log_b a})$
3. $\Theta(f(n))$, if $f(n)$ is $\Omega(n^{\log_b a + \epsilon})$ for some $\epsilon > 0$ and $af(n/b) < cf(n)$ for some $c < 1$ and n sufficiently large

Examples

"Quicksort recurrence" (work):

$$T(n) \quad = \quad 2T(n/2) + n$$

Case 2 applies, $T(n) = \Theta(n \log n)$ with $a = 2, b = 2$

"Quicksort recurrence" (time):

$$T(n) \quad = \quad 2T(n/2) + 1$$

Case 2 applies, $T(n) = \Theta(\log n)$ with $a = 1, b = 2$

$$T(n) \quad = \quad 9T(n/3) + n$$

Case 1 applies, $T(n) = \Theta(n^{\log_3 9}) = \Theta(n^2)$ with
$a = 9, b = 3, f(n) = n = n^1 = n^{(\log_3 9)-1}$

$$T(n) \quad = \quad T(2n/3) + 1$$

Case 2 applies, $T(n) = \Theta(n \log n)$ with $a = 1, b = 3/2$

$$T(n) \quad = \quad 3T(n/4) + n \log n$$

Case 3 applies, $T(n) = \Theta(n \log n)$ with
$a = 3, b = 4, f(n) = n \log n > n^{\log_4(3)+\epsilon} = n$ with $\epsilon = 1 - \log_4(3)$

$$T(n) \quad = \quad 2T(n/2) + n \log n$$

Case 3 does not apply, there is no $\epsilon > 0$ such that $n \log n \geqslant n^{1+\epsilon}$

Solving recurrences, Master Theorem (II)

A different version of the Master Theorem: let $T(n)$ be given by the recurrence

$$T(n) \leqslant aT(n/b) + \gamma n^c \log^d n$$

for constants $a \geqslant 1, b > 1, c \geqslant 0, d \geqslant 0, \gamma > 0$. Then $T(n)$ is

1. $O(n^c \log^d n)$ if $b^c > a$
2. $O(n^c \log^{d+1} n)$ if $b^c = a$
3. $O(n^{\log^b a})$ if $b^c < a$

Example:

$$T(n) = 2T(n/2) + n \log n$$

Case 1 applies, $a = 2, b = 2, c = 1, d = 1, \gamma = 1$, and $T(n)$ is $O(n \log^2 n)$

Solving recurrences (III)

Theorem

Let $T(n)$ be defined by the recurrence

$$T(n) \leqslant \sqrt{n}T(\sqrt{n}) + an$$

for a constant $a > 0$. Then $T(n)$ is $O(n \log \log n)$