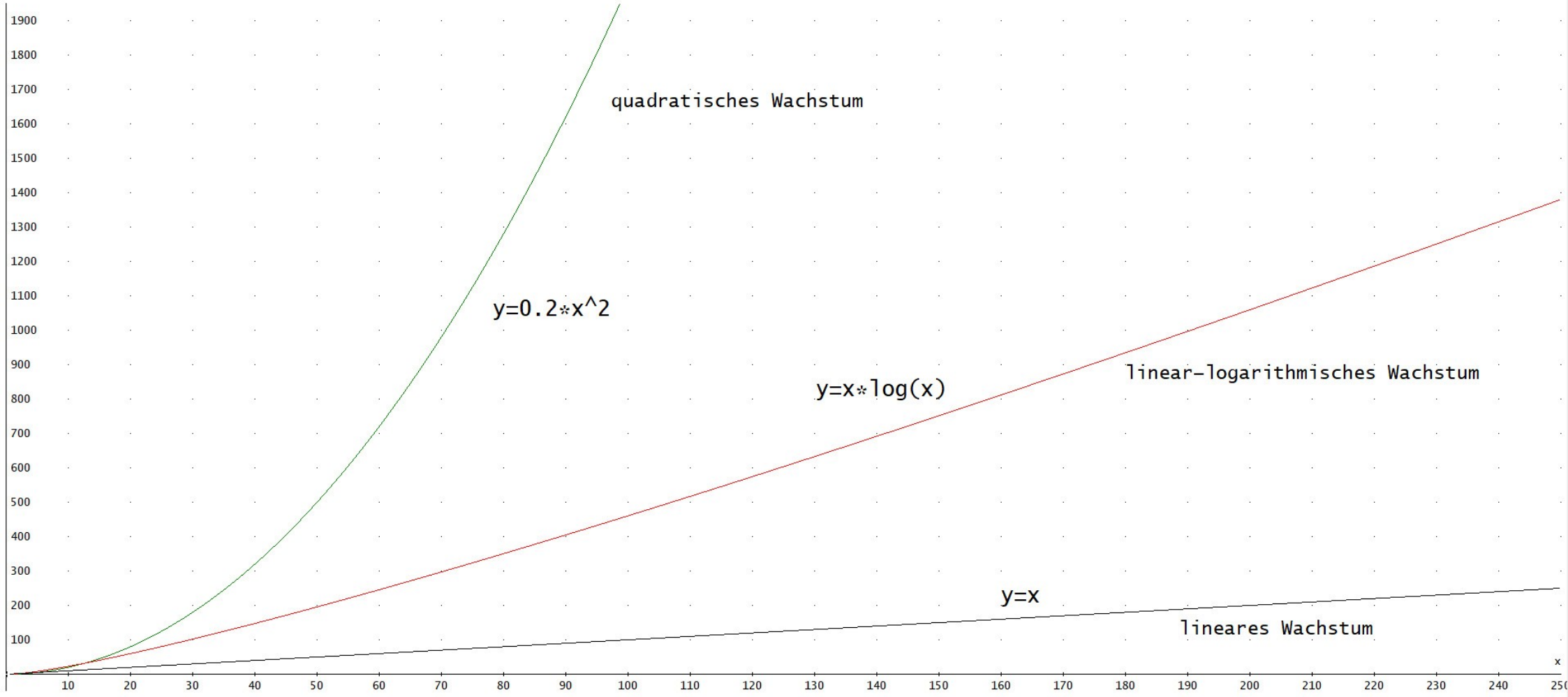
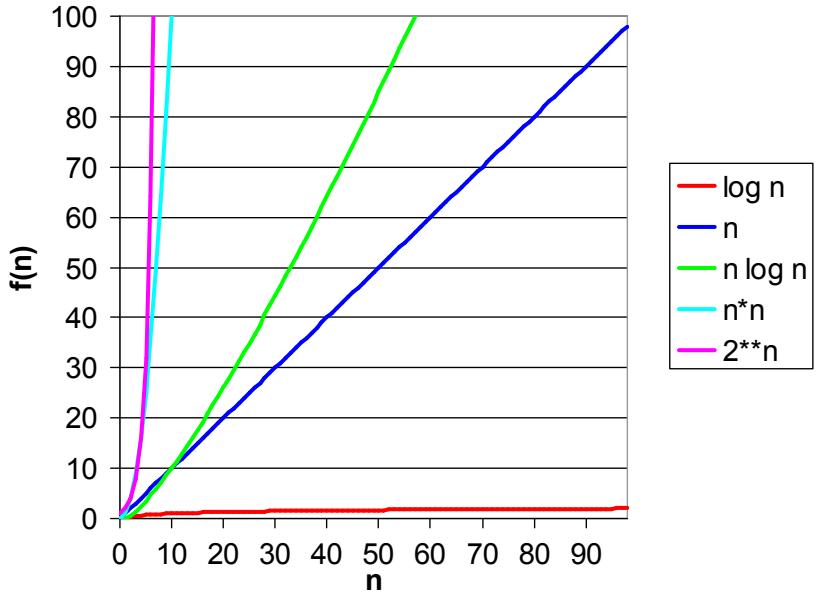

quadratisches Wachstum
y=0.2*x^2
y=x*log(x)
linear-logarithmisches Wachstum
y=x
lineares Wachstum
x
1900
1800
1700
1600
1500
1400
1300
1200
1100
1000
900
800
700
600
500
400
300
200
100
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
190
200
210
220
230
240

Wichtige O-Klassen

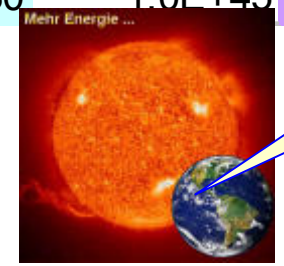
- **$O(2^n)$: Klasse aller exponentiellen Algorithmen**
  - Algorithmen, die ihre Lösung durch systematisches Ausprobieren finden
  - Beispiel: Wie packt man möglichst viele verschieden große Quader in einen Waggon?
  - Hoffnungslos ineffizient für große n
- **$\bigcup_{k \in IN} O(n^k)$: Klasse aller polynomialen Algorithmen**
  - Gelten als „noch praktikable“ Algorithmen
- **$O(n^2)$: Klasse aller quadratischen Algorithmen**
  - Einfache Sortieralgorithmen sind quadratisch (bubbleSort, insertionSort, selectionSort)
- **O(n*log(n)) : loglineare Algorithmen**
  - Gute Sortieralgorithmen sind loglinear (quickSort)
- **O(n) : Klasse aller linearen Algorithman**
  - sehr gut behandelbare Algorithmen
  - Beispiel: lineare Suche
- **O(log(n)) : logarithmische Algorithmen**
  - Extrem effizient
  - Beispiel binäre Suche
- **O(1) : Klasse aller konstanten Algorithmen**
  - Laufzeit unabhängig von Datengrösse



Wachstum einiger Funktionen

n	log n	n	n log n	n*n	n*n*n	2**n
1	0,0	1	0,0	1	1	2
10	3,3	10	33,2	100	1000	1024
100	6,6	100	664,4	10000	1000000	1,E+30
1.000	10,0	1.000	9965,8	1000000	1000000000	1,E+301
10.000	13,3	10.000	132877,1	100000000	1,0E+12	
100.000	16,6	100.000	1660964,0	1,0E+10	1,0E+15	
1.000.000	19,9	1.000.000	19931568,6	1,0E+12	1,0E+18	
10.000.000	23,3	10.000.000	232534966,6	1,0E+14	1,0E+21	
100.000.000	26,6	100.000.000	2657542475,9	1,0E+16	1,0E+24	
1.000.000.000	29,9	1.000.000.000	3,0E+10	1,0E+18	1,0E+27	
1,0E+10	33,2	1,0E+10	3,3E+11	1,0E+20	1,0E+30	
1,0E+11	36,5	1,0E+11	3,7E+12	1,0E+22	1,0E+33	
1,0E+12	39,9	1,0E+12	4,0E+13	1,0E+24	1,0E+36	
1,0E+13	43,2	1,0E+13	4,3E+14	1,0E+26	1,0E+39	
1,0E+14	46,5	1,0E+14	4,7E+15	1,0E+28	1,0E+42	
1,0E+15	49,8	1,0E+15	5,0E+16	1,0E+30	1,0E+45	

Vor ca 1,0 E+14 msec habe ich meine Pyramide gebaut



Die Sonne wiegt ca. 3,0E+35 g

Machbarkeitsüberlegungen

- Angenommen:
  - im Test zeigt sich, dass ein Programm für 10 Datenwerte 1 sec benötigt
- Wenn der Algorithmus Komplexität O(f(n)) hat, wieviele Eingabedaten kann er in 1 Tag, 1 Jahr, 10 Jahren, 1000 Jahren verarbeiten?

Komplexität O(-)	Gegeben: 1 sec	Verarbeitbare Datengrösse 1 Tag	1 Jahr	10 Jahre	1000 Jahre
n	10	864000	315360000	3,E+14	9,9E+18
n*log(n)	10	165551	4724923	41402010	3,3E+09
n*n	10	930	17758	16506697	3153600000
n*n*n	10	95	681	64830	2150492
2**n	10	20	35	38	45

- Folgerung
  - Polynomiale Algorithmen skalieren auch auf große Datenmengen
  - Exponentielle Algorithmen sind für große Datenmengen wertlos