CS 4/56101

Design and Analysis of Algorithms

Kent State University

Dept. of Math & Computer Science

LECT-5

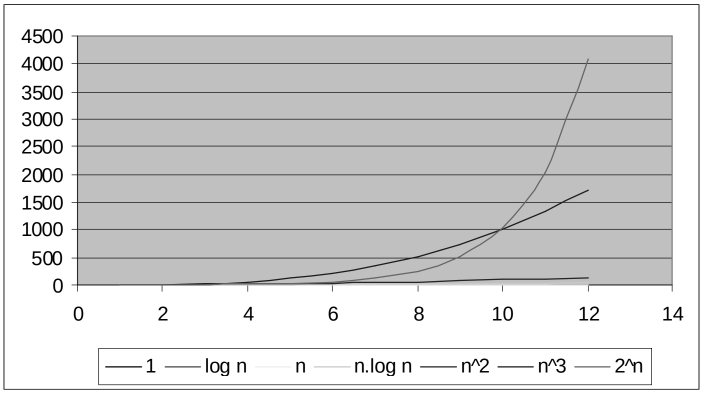
Growth of Functions

How to Compare Functions?

3

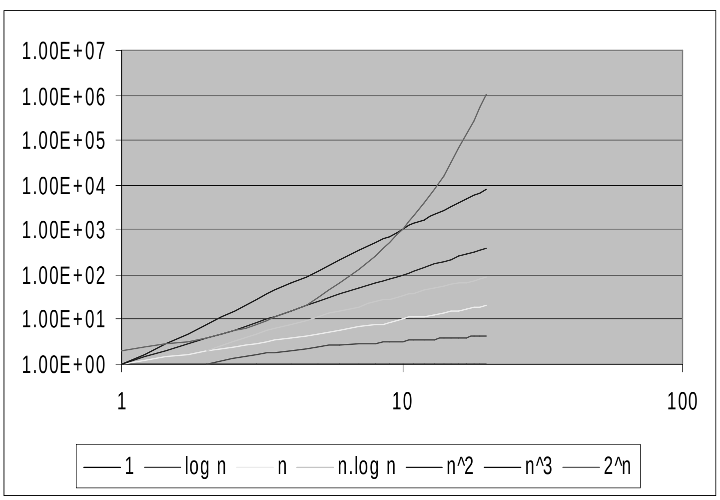
Relative Sizes of Functions (move them at the end)

n	1	log n	n	n.log n	n^2	n^3	2^n
1	1	0	1	0	1	1	2
10	1	3.321928	10	33.21928	100	1000	1024
100	1	6.643856	100	664.3856	10000	1000000	1.26765E+30
1000	1	9.965784	1000	9965.784	1000000	1000000000	1.0715E+301



DESIGN & ALALYSIS OF ALGORITHM

Logarithmic Scale



- Order of Functions

DESIGN & ALALYSIS OF ALGORITHM

Asymptotics

- Asymptotics means the study of functions of a parameter n, as n becomes larger and larger without bound.
- We will focus on:
  - algorithms when it is running on relatively large data.
  - on major characteristics, without being blinded by the details.
  - On general principles that will apply to the analysis and comparison of many classes of algorithms.

DESIGN &
ALALYSIS OF
ALGORITHM

Simplifying Functions: The Big Picture

- $f(n)=\Theta(g(n))$
- $f(n) = O(g(n))$
- $f(n) = \Omega(g(n))$

Θ-notation

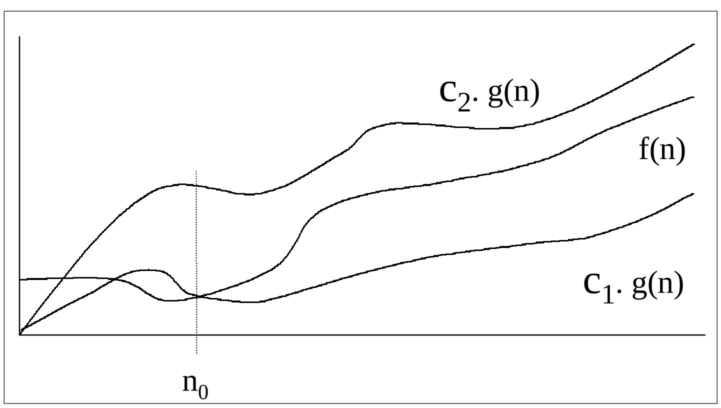
DESIGN &
ALALYSIS OF
ALGORITHM

- A function f(n) belongs to the set $\Theta(g(n))$ if there exist positive constants c_1 and c_2 such that it can be sandwiched between $c_1.g(n)$ and $c_2.g(n)$, for sufficiently large n.
- Example: $2.n^2 + 2 = \Theta(n^2)$
- We say g(n) is an **asymptotically tight bound** for f(n).

Although it should be "belongs to", we write it with equality

Intuitive Picture of Θ-notation

DESIGN &
ALALYSIS OF
ALGORITHM



$f(n) = \Theta(g(n))$

A rule for deriving g(n)

DESIGN & ALALYSIS OF ALGORITHM

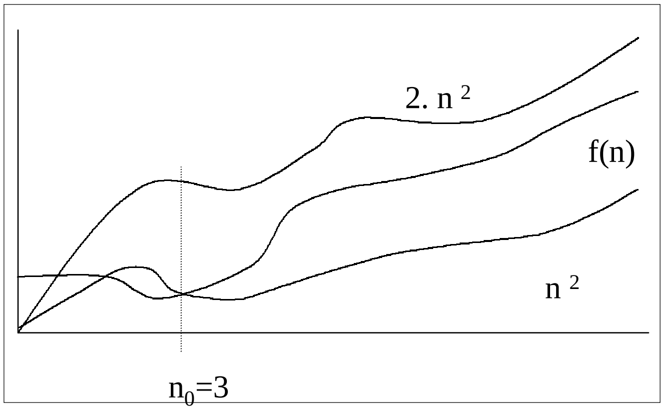
- Given any function f(n), throw away the lower order term, and throw away the leading coefficient. It will give the class g(n).
  - $f(n) = \Theta(g(n))$
- Examples:
  - $2.n^2 - 3n = \Theta(n^2)$
  - $2.n^{.5} + 3n = \Theta(n)$

Proof

DESIGN & ALALYSIS OF ALGORITHM

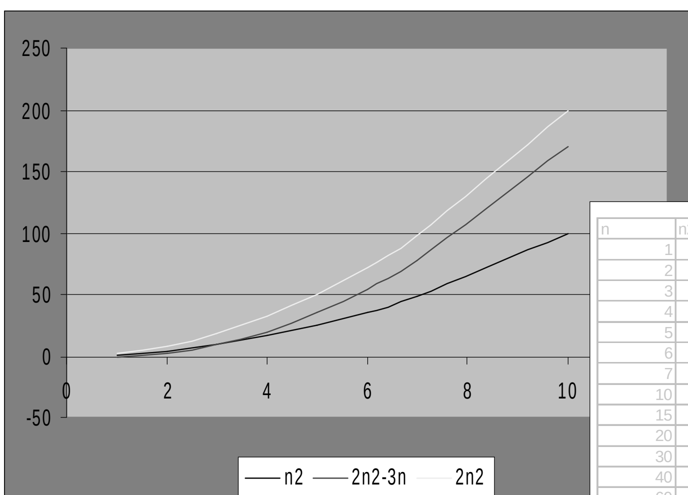
- We must show the positive constants c_1, c_2 and n_0 such that:
  - $c_1 n^2 \le 2.n^2 - 3n \le c_2 n^2$
- For all $n \ge n_0$, dividing by n^2 yields:
  - $c_1 \le 2 - 3/n \le c_2$
- The R.H.S. in equality can be made to hold:
  - for any $n \ge 1$, by choosing $c_2 \ge 2$
- The L.H.S. in equality can be made to hold:
  - for any $n \ge 3$, by choosing $c_1 \le 1$

Θ-notation



DESIGN &
ALALYSIS OF
ALGORITHM

Excel-Plot



n	n2	2n2-3n	2n2
1	1	-1	2
2	4	2	8
3	9	9	18
4	16	20	32
5	25	35	50
6	36	54	72
7	49	77	98
10	100	170	200
15	225	405	450
20	400	740	800
30	900	1710	1800
40	1600	3080	3200
60	3600	7020	7200
100	10000	19700	20000

DESIGN &
ALALYSIS OF
ALGORITHM

O-notation (Big-Oh)

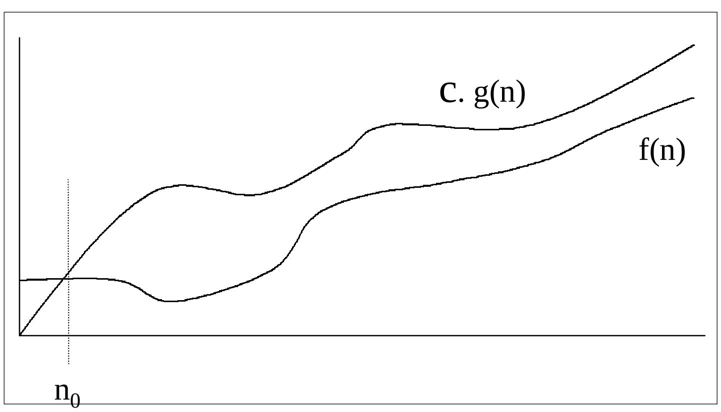
DESIGN & ALALYSIS OF ALGORITHM

- A function f(n) belongs to the set O(g(n)) if there exist a positive constants c such that it can be bounded below c.g(n) for sufficiently large n.
- Examples: $2.n^2 + 2 = O(n^2)$
  - If $f(n) = \Theta(g(n))$, it must be $f(n) = O(g(n))$
  - Also, $2.n^2 + 2 = O(n^3)$
- We say g(n) is an **asymptotically upper bound** for f(n).

Although it should be "belongs to", we write it with equality

Intuitive Picture of O-notation

DESIGN & ALALYSIS OF ALGORITHM



$f(n) = O(g(n))$

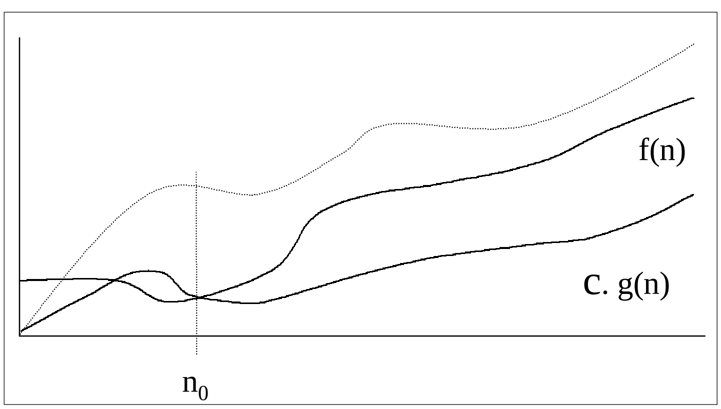
Ω-notation (Omega)

- A function f(n) belongs to the set O(g(n)) if there exist a positive constants c such that it can bounded above c.g(n) for sufficiently large n.
- Example: $2.n^2 + 2 = \Omega(n^2)$
  - If $f(n) = \Theta(g(n))$, it must be $f(n) = \Omega(g(n))$
  - Also, $2.n^2 + 2 = \Omega(n)$
- We say g(n) is an **asymptotically lower bound** for f(n).

Although it should be "belongs to", we write it with equality

Intuitive Picture of Ω -notation

DESIGN & ALALYSIS OF ALGORITHM



$$f(n) = \Omega(g(n))$$

o-notation (little oh)

- O-notation refers to upper bound that is not asymptotically tight.
  – A function f(n) belongs to the set O(g(n)) if for all positive constants c, f(n) can bounded above c.g(n) for sufficiently large n.
  – Or for all c, $0 \le f(n) \le c.g(n)$
- Example:2
  – $n^2 = \mathbf{O}(n^2)$
  – $2n^2 = \mathbf{O}(n^3)$?
  – But:
  – $2n^2 \ne \mathbf{o}(n^2)$
  – $2n^2 = \mathbf{o}(n^3)$

f(n)=o(g(n)) is a way of saying f(n) is insignificant relative to g(n).

DESIGN & ALALYSIS OF ALGORITHM

LECT-05, S-18
ALG00S, javed@kent.edu
Javed I. Khan@1999

ω-notation (little omega)

- ω-notation refers to lower bound that is not asymptotically tight.
  – A function f(n) belongs to the set **ω(g(n))** if for all positive constants c, f(n) can bounded below c.g(n) for sufficiently large n.
  – Or for all c, $c.g(n) \le f(n)$
- Example:2
  – $\mathbf{n^2 = \Omega (n^2)}$
  – $\mathbf{2n^2 = \Omega(n)}$ **?**
  – But:
  – $2n^2 \ne \omega(n^2)$
  – $2n^2 = \omega(n)$

ω-notation is to Ω- notation, what o-nation is to O-notation.

DESIGN & ALALYSIS OF ALGORITHM

LECT-05, S-19
ALG00S, javed@kent.edu
Javed I. Khan@1999

Comparison of Functions (QUIZ)

DESIGN & ALALYSIS OF ALGORITHM

- Transitivity:
- f(n)=Θ(g(n)) and g(n)=Θ(h(n)), f(n)=Θ(h(n))?
- f(n)=O(g(n)) and g(n)=O(h(n)), f(n)=O(h(n))?
- f(n)= Ω(g(n)) and g(n)= Ω(h(n)), f(n)= Ω(h(n))?
- f(n)= ω(g(n)) and g(n)= ω(h(n)), f(n)= ω(h(n))?
- f(n)= o(g(n)) and g(n)= o(h(n)), f(n)= o(h(n))?

LECT-05, S-20

ALG00S, javed@kent.edu
Javed I. Khan@1999

Comparison of Functions (QUIZ)

DESIGN & ALALYSIS OF ALGORITHM

- Reflexivity
- Given: f(n)=Θ(g(n))

- f(n)=O(g(n))?
- f(n)= Ω(g(n))?
- f(n)= ω(g(n))?
- f(n)= o(g(n))?

LECT-05, S-21

ALG00S, javed@kent.edu
Javed I. Khan@1999

Comparison of Functions (QUIZ)

DESIGN &
ALALYSIS OF
ALGORITHM

- Symmetry
- Given: $f(n)=\Theta(g(n))$
- $g(n)=\Theta(f(n))$?

- Given: $f(n)=O(g(n))$
- $g(n)=O(f(n))$?
- $g(n)= \Omega(f(n))$?

Ordering of Function

DESIGN &
ALALYSIS OF
ALGORITHM

- We can order a set of function just like a set of integers by using asymptotics:

- if f(n)=0(g(n)), g(n)=O(h(n)), h(n)=O(k(n))…
- we can say:
- f(n)<g(n)<h(n)...

Analysis of Algorithm

DESIGN & ALALYSIS OF ALGORITHM

- A class of students have been given the task of developing a solution for an algorithm to count the number of snow flakes looking through the window. Here are the running time of the solutions.

$$A = 200 \times \log.(\log n) + 32 \times \log.\log.\log(\log n^2)$$
$$B = 2^{\log n} - 3n^{.5} + 10$$
$$C = \sqrt{2}^{\log n} + n^{\frac{1}{2}}$$
$$D = 3 \times 2^{2^n} + \log.(2 \log n)$$
$$E = 2^n - 5n + 500$$
$$F = 10 + .5 \times \sqrt{n} + 432$$

How can you compare their solutions?

Step-1: Focus on Big-Picture

DESIGN & ALALYSIS OF ALGORITHM

$$A = 200 \times \log.(\log n) + 32 \times \log.\log.\log(\log n^2)$$
$$B = 2^{\log n} - 3n^{.5} + 10$$
$$C = \sqrt{2}^{\log n} + n^{\frac{1}{2}}$$
$$D = 3 \times 2^{2^n} + \log.(2 \log n)$$
$$E = 2^n - 5n + 500$$
$$F = 10 + .5 \times \sqrt{n} + 432$$

$$A = O(\log.(\log n))$$
$$B = O(2^{\log n})$$
$$C = O(\sqrt{2}^{\log n})$$
$$D = O(2^{2^n})$$
$$E = O(2^n)$$
$$F = O(\sqrt{n})$$

- D>E>B>C=F>A