Algorithm Analysis

Analysis of Algorithms

- How much resource is required ?
- Measures for efficiency
  - Execution time → time complexity
  - Memory space → space complexity
- Observation :
  - The larger amount of input data an algorithm has, the larger amount of resource it requires.
  - Complexities are functions of the amount of input data (input size).

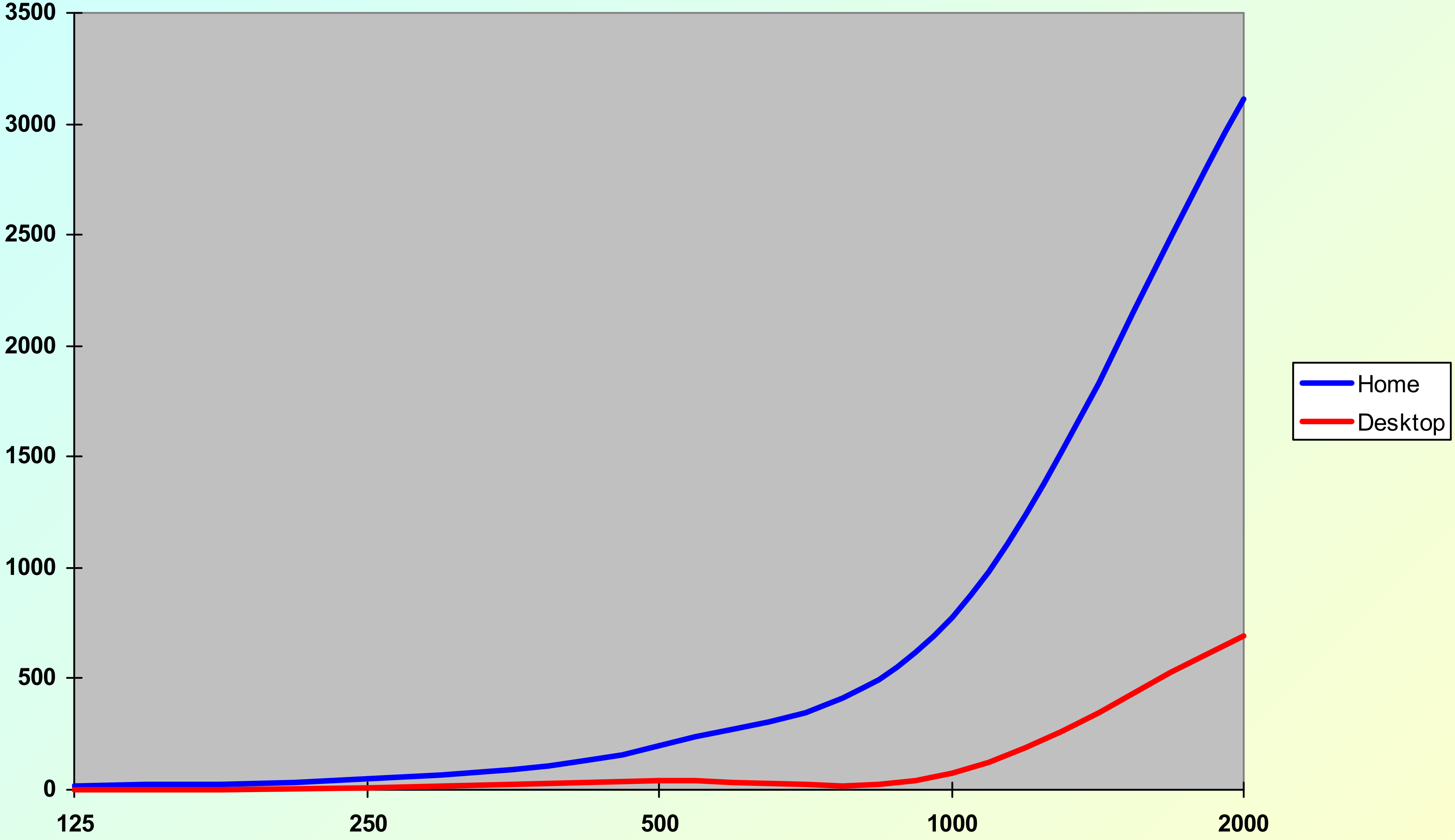
What do we use for a yardstick?

- The same algorithm will run at different speeds and will require different amounts of space.
  - When run on different computers, different programming languages, different compilers.
- But algorithms usually consume resources in some fashion that depends on the size of the problem they solve.
  - Some parameter **n** (for example, number of elements to sort).

An example of a sorting algorithm

- We run this sorting algorithm on two different computers, and note the time (in milliseconds) for different sizes of input.

Array Size n	Home Computer	Desktop Computer
125	12.5	2.8
250	49.3	11.0
500	195.8	43.4
1000	780.3	72.9
2000	3114.9	690.5


3500
3000
2500
2000
1500
1000
500
0
125
250
500
1000
2000
Home
Desktop

Contd.

- **Home Computer :**

  $f_1(n) = 0.0007772\ n^2 + 0.00305\ n + 0.001$

- **Desktop Computer :**

  $f_2(n) = 0.0001724\ n^2 + 0.00040\ n + 0.100$

  - **Both are quadratic function of n.**
  - **The shape of the curve that expresses the running time as a function of the problem size stays the same.**

Complexity Classes

- **The running time for different algorithms fall into different *complexity classes*.**
  - **Each complexity class is characterized by a different family of curves.**
  - **All curves in a given complexity class share the same basic shape.**
- **The *O-notation* is used for talking about the complexity classes of algorithms.**

Running time of algorithms

Assume speed is 10^7 instructions per second.

size n	10	20	30	50	100	1000	10000
n	.001 ms	.002 ms	.003 ms	.005 ms	.01 ms	.1 ms	1 ms
nlogn	.003 ms	.008 ms	.015 ms	.03 ms	.07 ms	1 ms	13 ms
n^2	.01 ms	.04 ms	.09 ms	.25 ms	1 ms	100 ms	10 s
n^3	.1 ms	.8 ms	2.7 ms	12.5 ms	100 ms	100 s	28 h
2^n	.1 ms	.1 s	100 s	3 y	$3x10^{13}$c	inf	inf

- **The complexity classes:**

$\log_2 n$

n

$n \log_2 n$

n^2

n^3

2^n

$n!$

Complexity increases

Introducing the language of O-notation

- **Definition:**

  $f(n) = O(g(n))$ if there exists positive constants c and n_0 such that $f(n) \leq c.g(n)$ when $n \geq n_0$.

- The big-Oh notation is used to categorize the complexity class of algorithms.
  - It gives an upper bound.
  - Other measures also exist, like small-Oh, Omega, Theta, etc.

Examples

- $f(n) = 2n^2+4n+5$ is $O(n^2)$.
  - One possibility: $c=11$, and $n_o=1$.
- $f(n) = 2n^2+4n+5$ is also $O(n^3)$, $O(n^4)$, etc.
  - One possibility: $c=11$, and $n_o=1$.
- $f(n) = n(n-1)/2$ is $O(n^2)$.
  - One possibility: $c=1/2$, and $n_o=1$.
- $f(n) = 5n^4+\log_2 n$ is $O(n^4)$.
  - One possibility: $c=6$, and $n_o=1$.
- $f(n) = 75$ is $O(1)$.
  - One possibility: $c=75$, and $n_o=1$.

Complexities of Known Algorithms

Algorithm	Best-case	Average-case	Worst-case
Selection sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$
Quick sort	$O(n \log_2 n)$	$O(n \log_2 n)$	$O(n^2)$
Merge sort	$O(n \log_2 n)$	$O(n \log_2 n)$	$O(n \log_2 n)$
Linear search	$O(1)$	$O(n)$	$O(n)$
Binary search	$O(1)$	$O(\log_2 n)$	$O(\log_2 n)$

Observations

- There is a big difference between polynomial time complexity and exponential time complexity.
- Hardware advances affect only efficient algorithms and do not help inefficient algorithms.