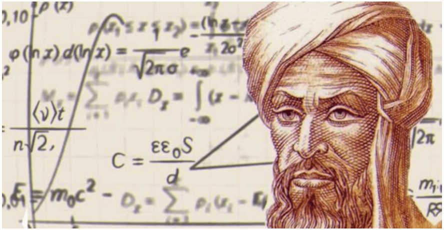

THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

Algorithms and Data Structures

Lecture 2 How to measure running time?

Jiamou Liu
The University of Auckland



Some Big Numbers

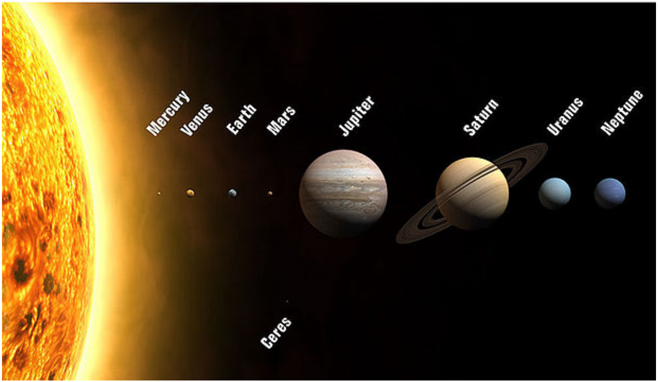
Age of the Universe.[1] $13.86 \times 10^9 \times 356 \times 24 \times 3600 = 4.2763 \times 10^{17}$ seconds.

[1] https://www.newscientist.com/question/how-old-is-the-universe/

Some Big Numbers

Nanometers from the Neptune to the Sun.[2]

$4.50 \times 10^9 \times 10^3 \times 10^9 = 4.5 \times 10^{21}$ nanometers.



[2] https://solarsystem.nasa.gov/planets/neptune/in-depth/

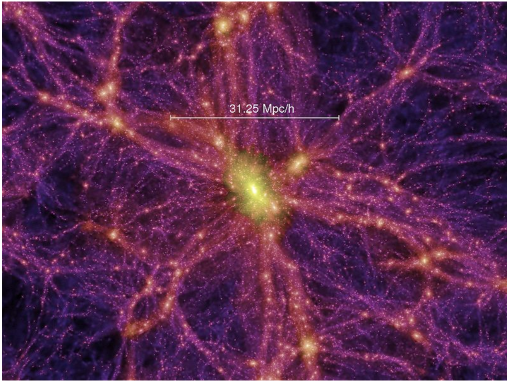
Some Big Numbers

Drops of Water in the Oceans.[3] $1.5 \times 10^9 \times 10^{15} \times 25 = 3.75 \times 10^{25}$ drops.

[3] "Ocean Volume and Depth." Van Nostrand's Scientific Encyclopedia 10th ed. 2008.

Some Big Numbers

Total Number of Particles in the Universe.[4] Between 10^{72} to 10^{87}.



[4] "The Biggest Numbers in the Universe. Bryan Clair 2001"

Characterising an Algorithm

There are three main characteristics of an algorithm:

- **Domain of definition:** The set of legal inputs.

  E.g.

  - The input to the **FASTFIB** algorithm is a natural number $n \in \mathbb{N}$
  - Suppose an algorithm takes an array of integers and looks for the maximum element in the array. The domain of definition is an array of integers (of arbitrary length).

- **Correctness:** Whether the algorithm gives the intended output for each legal input.

  E.g. **SLOWFIB** and **FASTFIB** are both correct as they output the nth Fibonacci number $F(n)$ on input $n \in \mathbb{N}$.

- **Resource use:** Running time and memory space used by the algorithm.

Algorithm analysis is the rigorous (i.e., mathematical) study of the resource use of algorithms.

Resource Use

Computational resource use

- Time: Execution time of an algorithm
- Space: Memory space taken by an algorithm

Note.

- Resource use depends on input size: usually grows as input becomes larger.

  E.g. Computing $F(0), F(1), F(2), F(3), F(4), F(5), \ldots$

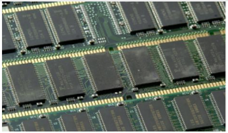
  We usually only when the input becomes large.
- Time-space trade-off.
- Running time is usually more important than memory space use.

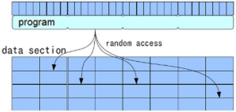
Running Time

- Running time of an algorithm depends on various implementation details (e.g., hardware, operating system, programming, language, etc.).
- Analysis of the algorithm must be based on the assumption that it is implemented with certain underlying model of computation.

Definition [Machine model]

- A models of computation is a collection of assumptions/idealisation about the type of machines and system environment that the algorithm is running on.
- We mostly assume the random access machine (RAM) model: A single processor connected to memory which it can access each part in constant time.





Definition [elementary operation]

An elementary operation is the basic measuring unit of running time, and it represents any instruction whose execution time does not depend on the input size.

Note.

- The elementary operations are usually assumed by a model of computation.
- RAM assumes that the following operations are elementary:
  - creating and accessing a single primitive variable.
  - arithmetic instructions (e.g. addition/multiplication) on primitive variables.
  - memory dereferencing
  - logical operations
  - control flow
  - creating/accessing an array
  - etc.

Definition [running time]

The running time of an algorithm `algo` running on input `inp` is defined as T(`inp`) which is the number of elementary operations used when `inp` is fed into `algo`.

E.g.

Recall: Algorithm FASTFIB

```
function FASTFIB(integer n)
    if n < 0 then return 0
    else if n = 0 then return 0
    else if n = 1 then return 1
    else
        a ← 1
        b ← 0
        for i ← 2 to n do
            t ← a
            a ← a + b
            b ← t
    return a
```

The running time of **FASTFIB** on input n is a linear function $An + B$ for some constants $A > 0$ and B.

Definition [running time]

The running time of an algorithm `algo` running on input `inp` is defined as $T(\texttt{inp})$ which is the number of elementary operations used when `inp`is fed into `algo`.

E.g.

Recall: Algorithm FASTFIB

```
function SLOWFIB(integer n)
    if n < 0 then return 0
    else if n = 0 then return 0
    else if n = 1 then return 1
    else return SLOWFIB(n − 1) + SLOWFIB(n − 2)
```

The running time of **SLOWFIB** on input n is bigger than 1.618^n.

Running Time Scales with Input Size

- Assume that `algo` has running time 1 on input of size 10.
- We use a function to describe the growth rate of running time of `algo`.

Growth rate		**Running time**			
Function	*Notation*	10	100	1000	10^7
Constant	1	1	1	1	1
Logarithmic	$\lg n$	1	2	3	7
Linear	$n/10$	1	10	100	10^6
"Linearithmic"	$\frac{1}{10} n \lg n$	1	20	300	7×10^6
Quadratic	$(n/10)^2$	1	100	10000	10^{12}
Cubic	$(n/10)^3$	1	1000	10^6	10^{18}
Exponential	2^{n-10}	1	10^{27}	10^{298}	$10^{3010296}$

Supercomputer Fugaku is the leader in the TOP500 list of the world's fastest supercomputers, which achieved more than 1 exaFLOPS (10^{18} floating point operations per second)[5].

Question. How long would take `algo` to run on Supercomputer Fugaku over inputs of different sizes?

- If `algo` has linear $n/10$ running time growth rate:

Growth rate	**Running time**			
Function	10	100	150	300
$n/10$	1	10	15	30
Time taken	1attosecond	10attosecond	15attosecond	30attosecond

- If `algo` has exponential 2^{n-10} running time growth rate:

Growth rate	**Running time**			
Function	10	100	150	300
2^{n-10}	1	10^{27}	10^{42}	10^{87}
Time taken	1 attosecond	31.7 years	31,700 trilion years	npiu😱

"npiu" stands for "number of particles in the universe".

[5] as of Feb 2022: https://en.wikipedia.org/wiki/Fugaku_(supercomputer)

Exercise.

(a) A quadratic algorithm with running time $T(n) = cn^2$ uses 500 elementary operations for processing 10 data items. How many will it use for processing 1000 data items?

Exercise.

(a) A quadratic algorithm with running time $T(n) = cn^2$ uses 500 elementary operations for processing 10 data items. How many will it use for processing 1000 data items?

Answer.

- We know that $T(10) = c \times 10^2 = 100c = 500$.
- Thus $c = 5$.
- Therefore $T(1000) = 5 \times 1000^2 = 5000,000$

(b) Algorithm A takes n^2 elementary operations to sort a file of n lines, while Algorithm B takes $50n \lg n$. Which algorithm is better when $n = 10$? When $n = 10^6$?

Exercise.

(a) A quadratic algorithm with running time $T(n) = cn^2$ uses 500 elementary operations for processing 10 data items. How many will it use for processing 1000 data items?

Answer.

- We know that $T(10) = c \times 10^2 = 100c = 500$.
- Thus $c = 5$.
- Therefore $T(1000) = 5 \times 1000^2 = 5000,000$

(b) Algorithm A takes n^2 elementary operations to sort a file of n lines, while Algorithm B takes $50n \lg n$. Which algorithm is better when $n = 10$? When $n = 10^6$?

Exercise.

(a) A quadratic algorithm with running time $T(n) = cn^2$ uses 500 elementary operations for processing 10 data items. How many will it use for processing 1000 data items?

Answer.

- We know that $T(10) = c \times 10^2 = 100c = 500$.
- Thus $c = 5$.
- Therefore $T(1000) = 5 \times 1000^2 = 5000,000$

(b) Algorithm A takes n^2 elementary operations to sort a file of n lines, while Algorithm B takes $50n \lg n$. Which algorithm is better when $n = 10$? When $n = 10^6$?

Answer.

- $n = 10$: A takes $10^2 = 100$, B takes $50 \times 10 \lg 10 = 500$ elementary operations.
- $n = 10^6$: A takes $(10^6)^2 = 10^{12}$, B takes $50 \times 10^6 \log 10^6 = 3 \times 10^8$ elementary operations.

To decide which algorithm to use, we need to have a rough idea about the size of the input which we will run the algorithm on.

(c) Algorithms A and B use exactly $T_A(n) = c_A n \lg n$ and $T_B(n) = c_B n^2$ elementary operations, respectively, for a problem of size n. Find the faster algorithm for processing $n = 2^{20}$ data items if A and B spend 10 and 1 operations, respectively, to process $2^{10} = 1024$ items.

(c) Algorithms A and B use exactly $T_A(n) = c_A n \lg n$ and $T_B(n) = c_B n^2$ elementary operations, respectively, for a problem of size n. Find the faster algorithm for processing $n = 2^{20}$ data items if A and B spend 10 and 1 operations, respectively, to process $2^{10} = 1024$ items.

Answer. We first need to find c_A and c_B.

- $T_A(2^{10}) = c_A 2^{10} \lg 2^{10} = 10$. Thus $10 c_A 2^{10} \lg 2 = 10$. Therefore $c_A = \frac{1}{2^{10}}$.
- $T_B(2^{10}) = c_B (2^{10})^2 = 1$. Thus $2^{20} c_B = 1$. Therefore $c_B = \frac{1}{2^{20}}$.

Then we compute the running time when $n = 2^{20}$.

- $T_A(2^{20}) = \frac{1}{2^{10}} 2^{20} \lg 2^{20} = 20 \times 2^{10}$.
- $T_B(2^{20}) = \frac{1}{2^{20}} (2^{20})^2 = 2^{20}$.

Therefore the faster algorithm for this task is A.

Summary

In this lecture, we covered the following:

- Main characteristics of an algorithm: input, correctness, resource use.
- **Algorithm analysis** is the rigorous study of the resource use of algorithms.
- Time and space resource use
- Machine model and RAM
- Elementary operations
- Running time: the number of elementary operations expressed as a function of input size
- Different growth rate: constant, logarithmic, linear, linearithmic, quadratic, cubic, exponential


THE UNIVERSITY OF
AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND