lecture 3 - Sep 09

## 1.1 MODEL

We do not want to build our analysis on one particular computer, programming language, compiler, etc.

⇝ abstract computational model:

(1) standard simple instructions, e.g.,

- simple arithmetic (addition, multiplication, ...)
- comparisons ($a[i] > a[j]$)
- assignments

take 1 unit of time to process.

(2) unbounded memory available

⇝ unrealistic model, but reasonable for comparison of algorithms.

often the size of the input influences the running time more than other aspects of the input.

(e.g. when finding the maximum in an unsorted array of $N$ distinct integers, the length $N$ of the array is what matters most)

Definition 1. For a given algorithm $A$, we denote

- by $T_{avg}(N)$ the average-case running time used by $A$ on inputs of size $N$.
- by $T_{worst}(N)$ the worst-case — " —
- by $T_{best}(N)$ the best-case — " —

## Example 3.

Problem: retrieve data stored under key $k$ in an unsorted table of $N$ data items.

Algorithm: sequential search

$T_{worst}(N) = \underline{N \cdot c}$ for some constant $c$

$T_{best}(N) = c$ —"—

$T_{avg}(N) = \underline{\frac{N}{2} \cdot c}$ —"—

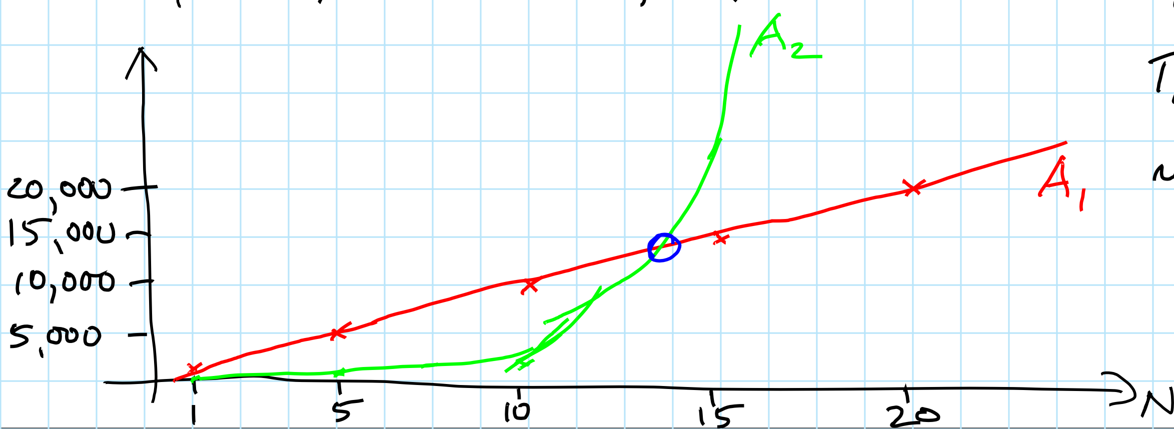
$= N \cdot \frac{c}{2} = N \cdot c'$ for some constant $c'$

How to compare two algorithms $A_1$ and $A_2$?

## Example 4.

Assume you know the following worst-time running time values.

| $N$ | $T^{A_1}_{worst}(N)$ | $T^{A_2}_{worst}(N)$ |
|---|---|---|
| 1 | 1,000 | 2 |
| 5 | 5,000 | 30 |
| 10 | 10,000 | 1,000 |
| 20 | 20,000 | 1,000,000 |




$T^{A_1}_{worst}(N) \approx 1{,}000 N$

$T^{A_2}_{worst}(N) \approx 2^N$

$\rightsquigarrow$ $A_1$ is much more efficient than $A_2$ for <u>large</u> inputs.

⇝ study the efficiency of algorithms for growing input size $N$.

## 1.2. MATHEMATICAL BACKGROUND

We saw that $T_2(N) = 2^N$ asymptotically grows faster than $T_1(N) = 1{,}000 \cdot N$.

Even $\underbrace{0.000000001 \cdot 2^N}_{\text{exponential growth}}$ grows faster than $\underbrace{9^{99} \cdot N + 7^{\pi}}_{\text{linear growth}}$.

growth rate classes

| | e.g., | |
|---|---|---|
| $c \cdot 1$ | constant | |
| $c \cdot \log(N)$ | logarithmic | |
| $c \cdot \sqrt{N}$ | | |
| $c \cdot N$ | linear | |
| $c \cdot N \log(N)$ | | $N \cdot \log(N)$ |
| $c \cdot N^2$ | quadratic | $N \cdot N$ |
| $c \cdot N^3$ | cubic | |
| [ $c \cdot N^k$, $k \geq 2$ | polynomial ] | |
| $c \cdot 2^N$ | exponential | |
| $c \cdot 3^N$ | | |
| $c \cdot 4^N$ | | |

$\mathbb{N} = \{0, 1, 2, \dots\}$ denotes the set of natural numbers.

$\mathbb{R}^{\geq 0}$ denotes the set of non-negative real numbers.

**Definition 2.** Let $T : \mathbb{N} \to \mathbb{R}^{\geq 0}$, $f : \mathbb{N} \to \mathbb{R}^{\geq 0}$ be any two functions.

(a) $T(N) = O(f(N))$ if and only if there are positive constants $c$ and $n_0$ such that

$$T(N) \leq c \cdot f(N) \quad \text{for all } N \geq n_0.$$

for example $1{,}000\,N = O(2^N) \to n_0 \approx 14$

$1{,}000\,N = O(N) \to c \geq 1{,}000$

$1{,}000\,N \leq c \cdot N$

$T(N)$ is asymptotically upper-bounded by $f(N)$