IFT2125 - Introduction to algorithms

The analysis of the algorithms (BB, Chapters 3 and 4)

Pierre McKenzie

DIRO, University of Montreal

Fall 2017

Re-recall: orders

Let $f: \mathbb{N} \to \mathbb{R}_{\geq 0}$.

The order of f is

$$O(f) = \{t: \mathbb{N} \to \mathbb{R}_{\geq 0} \mid (\exists c \in \mathbb{R}_{\geq 0}) (\underbrace{\overset{\infty}{\forall}}_{\text{for all } n \text{ sufficiently large}} n \in \mathbb{N}) [t(n) \leq cf(n)]\}$$

The omega f is

$$\Omega(f) = \{t: \mathbb{N} \to \mathbb{R}_{\geq 0} \mid (\exists d \in \mathbb{R}_{+}) (\overset{\infty}{\forall} n \in \mathbb{N}) [t(n) \geq df(n)]\}$$

The exact order of f is

$$\Theta(f) = Y(f) \cap \Omega(f).$$

Tools Available [1]:

1. if lim $\frac{f(n)}{g(n)}$ c R + Then f (n) E $G(\cdot g(n))$,
2. if lim $\frac{f(n)}{g(n)} = 0$ then f (n) $E\,0\,(g\,(n))$ but f (n) $EO\,i\,(g$ (n)), and
3. if lim $\frac{f}{\text{gin)}}$ = ± thenf oo (n) Q (g (n)) butf (n) $E\,t)$ (g (n)).

As an exercise in manipulating asymptotic notation, let us now prove a useful fact:

1. Passage extracted from BB, as well as any passage of my transparencies visibly reproduced from a book and not explicitly attributed.

Our approach to the analysis of algae, in summary

Analytic

In the worst case

Often the number of executions b of an instruction there will be barometer

We deem $t\,(n)$ to the order close, if possible at the exact order -près

- one that f $\overset{\infty}{\forall}$ n and any size of copy *No* majorises this b copy checks $t\,(n) \in O\,(f\,(n))$
- a g that $\overset{\infty}{\forall}$ n and at least one copy size n b minore this copy checks $t\,(n) \in \Omega\,(g\,(n))$
- provided that $g \in \Omega\,(f)$ t is concluded $(n) \in \Theta\,(f\,(n))$

Algorithms versus Computational Complexity Theory

A problem P is given.

Spring algorithmic:

To develop an efficient algorithm to solve P

determine the exact order of the execution time of the algorithm A.

Spring computational complexity:

borrow algorithmic his best algorithm, A to P

demonstrate that no algorithm is better than A to P
conclude that the complexity of the problem P is given by time
A performance.

Other possible measures

The memory used.
Can we always cut in memory?
Can we always do it in exchange for a longer time?

The time "parallel".
Access to m processors it accelerates?
Ideally: m times faster, or even better.

Supplement on orders (BB chapter 3)

Conditional Order

O, Ω and Θ have more secrets (is not it?).

Conditional order of $f(n)$ = notation for declaring a bound on $f(n)$ which is only valid for some n

Coupled to a condition on f,
$t(n) \in$ conditional order of $f(n) \Rightarrow t(n) \in$ unconditional order

Example: $\Omega(f(n) \mid n$ is power of 2) versus $\Omega(f(n))$.

Conditional Order

Formal definition (case O, same for the others)

Let f: $\mathbb{N} \to \mathbb{R}_{\geq 0}$ and P: $\mathbb{N} \to \{\text{true}, \text{false}\}$.

So

$$O(f(n) \mid P(n))$$

East

$$\{T: \mathbb{N} \to \mathbb{R}_{\geq 0} \mid \exists c \in \mathbb{R}_{\geq 0}, \overset{\infty}{\forall} n \in N, [P(n) = \Rightarrow t(n) \leq cf(n)]\}.$$

Rule of Harmony ("smoothness" rule)

If Θ (same for others)

$b \in \mathbb{N}_{\geq 2}$
$t(n) \in \Theta(f(n) \mid n$ is power $b)$.

The harmony rule serves to eliminate "n b is power":

Yes

$t(n)$ is é.nd (optionally non-decreasing), ie,
$\overset{\infty}{\forall} n \in N, t(n) \leq t(n+1)$
$f(n)$ is smooth, ie,
$f(n)$ is é.nd and $f(bn) \in O(f(n))$

so

$t(n) \in \Theta(f(n))$.

Operations on orders

$O(n^2 + n^3)$ has been set, but do $O(n^2) + O(n^3)$ has a direction? No !

It is therefore necessary to invent a definition. There she is :

$$O(f) + O(g)$$

East

$\{H: \mathbb{N} \rightarrow \mathbb{R}^{\geq 0} \mid \exists h_1 \in O(f), h_2 \in O(g) \overset{\infty}{\forall} n \in N, [h(n) = h_1(n) + h_2(n)]\}$

Sometimes useful to estimate the time of a block A followed by a block B.

Extends to other operators, for example $\times$ instead of $+$, and to any any pair of sets of functions, such as $Y(f) + \Theta(g)$.

Recurrence Supplement (BB Section 4.7)

Intelligent guesswork method

Method: estimate the shape of the solution by eye, then prove.

Eg significant recurrence (here n power b)

$$T(b^k) = \begin{cases} c = 0 & \text{if } k = k_0 \\ aT(b^{k-1}) + f(b^k) & \text{where } k > k_0, \end{cases}$$

where $k \in \mathbb{N}_0$, $a \in \mathbb{R}_{\geq 1}$, $b \in \mathbb{N}_{\geq 2}$, $f: \mathbb{N} \dashrightarrow R_{>0}$.

$$T(b^k) = \begin{cases} c = 0 & \text{if } k = k_0 \\ aT(b^{k-1}) + f(b^k) & \text{where } k > k_0, \end{cases}$$

Let $g(b^{k_0}) = c / (a^{k_0})$ and $g(b^k) = f(b^k) / (a^k)$ for $k > k_0$.
By induction on $k \geq k_0$:

$$T(b^k) = a^k \times [g(b^{k_0}) + g(b^{k_0+1}) + \cdots + g(b^k)].$$

Already $T(n) \in \Omega(n^{\log_b a} \mid n$ is the power of $b)$.
$(B^k)^{\log_b a} = a^k$

To get better, we must analyze $[+ \cdots +]$

$$T(b^k) = \begin{cases} c = 0 & \text{if } k = k_0 \\ aT(b^{k-1}) + f(b^k) & \text{where } k > k_0. \end{cases}$$

Lemma ("powers of b")

If $\varepsilon > 0$ *and* $f(n) \in O(n^{\log_b a - \varepsilon})$ *then*
$T(n) \in \Theta(n^{\log_b a} \mid n$ *is power* $b)$.

If $\varepsilon > 0$ *and* $f(n) \in O(n^{\log_b a + \varepsilon})$ *then*
$T(n) \in O(n^{\log_b a + \varepsilon} \mid n$ *is the power of* $b)$.

If $\varepsilon \geq 0$ *and* $f(n) \in O(n^{\log_b a} (\log n)^{\varepsilon})$ *then*
$T(n) \in O(n^{\log_b a} (\log n)^{\varepsilon + 1} \mid n$ *is power* $b)$.

Page 16

Application to asymptotic recurrence

Let $t: \mathbb{N} \longrightarrow \mathbb{R}_{>0}$ which is known only

$$t(n) \in {}_{1} t(\lceil n/b \rceil) + a_2 t(\lfloor n/b \rfloor) + O(f(n)),$$

or

- $f: \mathbb{N} \longrightarrow \mathbb{R}_{>0}$
- $a_1, a_2 \in \mathbb{R}_{\geq 0}, a_1 + a_2 \geq 1$
- $b \in \mathbb{N}_{\geq 2}.$

The following theorem solves this recurrence:

Page 17

Theorem (Solution of the asymptotic recurrence)

Let $a = a_1 + a_2.$

1. *If* $\varepsilon > 0$ *and* $(\log_b a - \varepsilon) \geq 0$ *and* $f(n) \in O(n^{\log_b a - \varepsilon})$ *then* $t(n) \in O(n^{\log_b a}).$
2. *If* $\varepsilon > 0$ *and* $f(n) \in O(n^{\log_b a + \varepsilon})$ *then* $t(n) \in O(n^{\log_b a + \varepsilon}).$
3. *If* $\varepsilon \geq 0$ *and* $f(n) \in O(n^{\log_b a} (\log n)^{\varepsilon})$ *then* $t(n) \in O(n^{\log_b a} (\log n)^{\varepsilon\ 1}).$

Also valid when

$$t(n) \in {}_{1} t(\lceil n/b \rceil) + a_2 t(\lfloor n/b \rfloor) + \Omega(f(n)),$$

and all the "O" replaced by "Ω".

Page 18

Proof (sketch) of the case $f(n) \in O(n^{\log_b a - \varepsilon})$

$$t(n) \in {}_{1} t(\lceil n/b \rceil) + a_2 t(\lfloor n/b \rfloor) + O(n^{\log_b a - \varepsilon})$$

1. Select c and k_0 such that for all $n \geq b^{k_0}$,
   $t(n) \leq {}_{1} t(\lceil n/b \rceil) + a_2 t(\lfloor n/b \rfloor) + \underbrace{cn^{\log_b a - \varepsilon}}_{\text{note: not decreasing}}$
2. Show that for all n, $t(n) \leq T(n)$ where

$$T(n) = \begin{cases} \max\{t(0), t(1), \ldots, t(b^{k_0})\} & \text{if } n \le b^{k_0} \\ {}_1 T(\lceil n/b \rceil) + a_2 T(\lfloor n/b \rfloor)\, cn + {}^{\log_b a - \varepsilon} & \textit{otherwise}. \end{cases}$$

3 Lemma powers $\Rightarrow T(n) \in O(n^{\log_b a} \mid n$ is the power of $b)$

4 Demo 3 $\Rightarrow T(n)$ is é.nd

5 $n^{\log_b a}$ is harmonious

6 Smoothness $\Rightarrow$ rule $T(n) \in O(n^{\log_b a})$

7 Points (2) and (6) $\Rightarrow t(n) \in O(n^{\log_b a})$.

Method of the characteristic equation

Already studied in the prerequisite courses for the resolution of recurrences linear equations with constant coefficients.

Several examples in BB.

Recurrence resolution

Linear homogeneous recurrences with constant coefficients:

Either the recurrence R

$$a_0 t_n + a_1 t_{n-1} + \ldots + a_k t_{n-k} = 0$$

Here are the steps of the resolution:

1) Find the characteristic polynomial P (x) of the recurrence R

2) Find the roots of P (x)

If these roots are distinct

3) The general solution is of the form $t_n = \sum_{i=1}^{k} c_i r_i^n$

4) Solve the system of linear equations given by the conditions Initial to find the value of the constants $c_1, c_2, \ldots, c_k$

5) Write the solution according to these constants

Recurrence resolution

Linear homogeneous recurrences with constant coefficients:

Either the recurrence R

$_0 t_n + _1 t_{n1} + \ldots + a_k t_{nk} = 0$

Here are the steps of the resolution:

1) Find the characteristic polynomial P (x) of the recurrence R

2) Find the roots of P (x)

If these roots are not all distinct

3) The general solution is of the form $t_n = \sum_{i=1} \sum_{j=0}^{m_i - 1} c_{ij} n^j r_i^n$

where we have roots of multiplicity r_i m_i

4) Solve the system of linear equations given by the conditions Initial to find the value of the constants $c_1, c_2, \ldots, c_k$

5) Write the solution according to these constants c_i

Non-homogeneous case

Type of recurrence:

$$_0 t_n + _1 t_{n-1} + \cdots + a_k t_{n-k} = b_1^n p_1(n) + b_2^n p_2(n) + \cdots$$

where a_i, b_j are constants,
$p_i(n)$ are polynomials.

Take as characteristic equation:

$$(A_0 x^k + _1 x^{k-1} + \cdots + a_k)(x - b_1)^{1 + \text{degree}(p_1)}(x - b_2)^{1 + \text{degree}(p_2)} \cdots = 0.$$