Jun 30

160

Ex 1

$m_1: w_1 > w_2$ $\quad S_{st} = \{(m_1, w_1), (m_2, w_2)\}$

$m_2: w_2 > w_1$

$w_1: m_1 > m_2$

$w_2: m_2 > m_1$

Ex 2

$m_1: w_1 > w_2$ $\quad S_{st_1} = \{(m_1, w_1), (m_2, w_2)\}$

$m_2: w_2 > w_1$ $\quad S_{st_2} = \{(m_1, w_2), (m_2, w_1)\}$

$w_1: m_2 > m_1$

$w_2: m_1 > m_2$

S_f

m valid partner of m $\quad V(m)$

the best valid partner $\quad best(m)$

$m(best(m)) \geq m(w)$ where $w = V(m)$

$$s^* = \{(m, best(m)); m \in M\}$$

the worst valid partner $\quad worst(w)$

$W(worst(w)) \leq w(m)$ where $m = V(w)$

$$S_{ww} = \{(worst(w), w); w \in W\}$$

Prop 1 $S_f = S^*$

Proof Assume $S_f \neq S^*$ choose m s.t. m was 1^{st} rejected by best(m)

& $w \neq best(m) = w_0$

$\exists\, m \in M$ s.t. $(m, w) \in S_f$ & $w \neq best(m) = w_0$

$m(w_0) > m(w)$

Case 1 (m', w_0) when m appl to w_0

$w_0(m') > w_0(m)$

Case 2 (m, w_0) when m' appl to w_0

(m', w_0)

$w_0(m') > w_0(m)$

$\exists$ p. s.t. m S

$(m, w_0) \in S$ $(m', w') \in S$

$m'(w_0) \geq m'(best(m')) \geq m'(w')$ $w' \neq w_0$

instability $\rightarrow$ $m'(w_0) > m'(w')$

Prop 2 $S_f = S_{ww}$

Proof Assume $S_f \neq S_{ww}$

$\exists\, w \in W$ s.t. $(m, w) \in S_f$ & $m \neq worst(w) = m_0$

$(m_0, w) \in S$ $(m, w') \in S$

By Pr. 1, $w = best(m)$

$m(w) > m(w')$ $w_0(m) > w(m_0)$

Contradiction

Types of Complexity Measures of Algorithms

- *Static* **complexity** measures depend only on an algorithm that is measured.
- *Dynamic* **complexity** measures depend both on an algorithm that is measured and on the input.
- *Processual* **complexity** measures depend on an algorithm or program, its realization, and on the input.

Example 1. The lines of the description.

Example 2. The length of the algorithm.

An important type of dynamic complexity measures is **Computational Complexity** of algorithms, which measures resources utilized by the algorithm.

Examples:

Time complexity $T_A(x)$

Space complexity $S_A(x)$

Worst-case complexity

$$T_A(n) = \max \{ T_A(x);\ l(x) = n\}$$

Average complexity

$$T_A(n) = \text{average} \{ T_A(x);\ l(x) = n\}$$

Best-case complexity

$$T_A(n) = \min \{ T_A(x);\ l(x) = n\}$$

Relations between functions

The **asymptotic** behavior of a function $f(n)$ refers to the growth of $f(n)$ as n gets large. We typically ignore small values of n, since we are usually interested in estimating how slow the program will be on large inputs. A good rule of thumb is: the slower the asymptotic growth rate, the better the algorithm (although this is often not the whole story).

By this measure, a linear algorithm, *i.e.*, with time complexity n, is always asymptotically better than a quadratic one, *e.g.*, with time complexity n^2. For moderate values of n, the quadratic algorithm could very well take less time than the linear one. However, the linear algorithm will always be better for sufficiently large inputs.

Asymptotic boundaries

A function $f(n)$ is an **asymptotic upper bound** of a function $g(n)$ if

If there are a constant $c > 0$ and a natural number m such that

$$g(n) \leq cf(n)$$

for all $n > m$

Formal definition

$$\exists\, c > 0\ \exists\, m \in N\ \forall\, n > m\ (\ g(n) \leq cf(n)\ )$$

It is denoted by $g(n) = O(f(n))$

more rigorous → $g(n) \in O(f(n))$

Big O notation

A function $f(n)$ is an **asymptotic lower bound** of a function $g(n)$ if

If there are a constant $c > 0$ and a natural number m such that

$$g(n) \geq cf(n)$$

for all $n > m$

Formal definition

$$\exists\, c > 0\ \exists\, m \in N\ \forall\, n > m\ (\ g(n) \geq cf(n)\ )$$

It is denoted by $g(n) = \Omega(f(n))$.

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

A function $f(n)$ is an **asymptotically tight bound** of a function $g(n)$ if $g(n) = O(f(n))$ and $g(n) = \Omega(f(n))$.

It is denoted by $g(n) = \Theta(f(n))$.

O is omichron