

Computational complexity

Tight bound Θ

$$\Theta(g) = \{f; \exists c_1, c_2, n_0 > 0, \\ \forall n > n_0 : \\ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)\}$$

Upper bound \mathcal{O}

$$\mathcal{O}(g) = \{f; \exists c, n_0 > 0, \\ \forall n > n_0 : \\ 0 \leq f(n) \leq cg(n)\}$$

Lower bound Ω

$$\Omega(g) = \{f; \exists c, n_0 > 0, \\ \forall n > n_0 : \\ 0 \leq cg(n) \leq f(n)\}$$

Imprecise boundaries *o* and ω

$$o(g) = \{f; \forall c > 0, \exists n_0 > 0, \forall n > n_0 : 0 \leq f(n) < cg(n)\}$$

$$\omega(g) = \{f; \forall c > 0, \exists n_0 > 0, \forall n > n_0 : 0 \leq cg(n) < f(n)\}$$

Properties

- transitivity $f \in \Theta(g) \wedge g \in \Theta(h) \Rightarrow f \in \Theta(h)$ (for all bounds)
- reflexivity $f \in \Theta(f)$ (for Θ , \mathcal{O} and Ω)
- symmetry $f \in \Theta(g) \Leftrightarrow g \in \Theta(f)$
- transpose symmetry $f \in \mathcal{O}(g) \Leftrightarrow g \in \Omega(f)$
 $f \in o(g) \Leftrightarrow g \in \omega(f)$

Divide and conquer

- divide** the problem into several (equal) parts

- (recursively) **conquer (solve)** each of the sub problems

- combine** sub problem solutions

Simplified Masters

$$T(n) = aT(\frac{n}{b}) + \Theta(n^d) \\ a \geq 1 \\ b > 1 \\ d \geq 0$$

- $a > b^d \rightarrow T(n) = \Theta(n^{\log_b a})$

- $a = b^d \rightarrow T(n) = \Theta(n^d \log_b n)$

- $a < b^d \rightarrow T(n) = \Theta(n^d)$

Masters

$$T(n) = aT(\frac{n}{b}) + f(n) \\ a \geq 1 \\ b > 1$$

- $f(n) = \mathcal{O}(n^{\log_b a - \epsilon}) \rightarrow T(n) = \Theta(n^{\log_b a}), \epsilon > 0$

- $f(n) = \Theta(n^{\log_b a}) \rightarrow T(n) = \Theta(n^{\log_b a} \log(n))$

- $f(n) = \Omega(n^{\log_b a + \epsilon}) \rightarrow T(n) = \Theta(f(n)), \epsilon > 0$ and $af(\frac{n}{b}) \leq cf(n)$ for some $c < 1$ and big enough n

- case2 ext: $f(n) = \Theta(n^{\log_b a} \log^k(n)) \rightarrow T(n) = \Theta(n^{\log_b a} \log^{k+1}(n))$

Akra-Bazzi

$$T(n) = \sum_{i=1}^k a_i T(b_i n) + f(n), n > n_0$$

- $n_0 \geq \frac{1}{b_i}, n_0 \geq \frac{1}{1-b_i}$ for each i ,

- $a_i > 0$ for each i ,

- $0 < b_i$ for each i ,

- $k \geq 1$,

- $f(n)$ is non-negative function,

- $c_1 f(n) \leq f(u) \leq c_2 f(n)$, for each u satisfying condition: $b_i n \leq u \leq n$

$$T(n) = \Theta(n^p(1 + \int_1^n \frac{f(u)}{u^{p+1}} du))$$

we get p from:

$$\sum_{i=1}^k a_i b_i^p = 1$$

Extended Akra-Bazzi

$$T(n) = \sum_{i=1}^k a_i T(b_i n + h_i(n)) + f(n), n > n_0$$

all of the conditions from Akra-Bazzi still hold plus:

$$|h_i(n)| = \mathcal{O}(\frac{n}{\log^2 n})$$

Annihilators

Steps:

- Write the recurrence in operator form.

- Extract the annihilator for the recurrence.

- Factor the annihilator (if necessary).

- Extract the generic solution form the annihilator.

- Solve for coefficients using base cases (if known).

Operator	Definition
addition	$(f + g)(n) := f(n) + g(n)$
subtraction	$(f - g)(n) := f(n) - g(n)$
multiplication	$(a \cdot f)(n) := a \cdot (f(n))$
shift	$Ef(n) := f(n + 1)$
k -fold shift	$E^k f(n) := f(n + k)$
composition	$(X + Y)f := Xf + Yf$ $(X - Y)f := Xf - Yf$ $XYf := X(Yf) = Y(Xf)$
distribution	$X(f + g) = Xf + Xg$

Operator	Functions annihilated
$E - 1$	α
$E - a$	αa^n
$(E - a)(E - b)$	$\alpha a^n + \beta b^n \qquad (a \neq b)$
$(E - a_0)(E - a_1) \cdots (E - a_k)$	$\sum_{i=0}^k \alpha_i a_i^n \quad (a_i \text{ distinct})$
$(E - 1)^2$	$\alpha n + \beta$
$(E - a)^2$	$(\alpha n + \beta)a^n$
$(E - a)^2(E - b)$	$(\alpha n + \beta)a^n + \gamma b^n (a \neq b)$
$(E - a)^d$	$(\sum_{i=0}^{d-1} \alpha_i n^i)a^n$

<div>If X annihilates f, then X also annihilates Ef.</div> <div>If X annihilates both f and g, then X also annihilates $f \pm g$.</div> <div>If X annihilates f, then X also annihilates αf, for any constant α.</div>

<div>If X annihilates f and Y annihilates g, then XY annihilates $f \pm g$.</div>

Randomization

To avoid bad input sequences, the input can be intentionally randomized.

Pseudo random generator

Linear congruential generators

$$x_i = (ax_{i-1} + c) \mod m$$

- RANDU: $x_i = 65539x_{i-1} \pmod{2^{31}}$

- MINSTD $x_i = 16807x_{i-1} \pmod{2^{31} - 1}$

- Combinations of linear congruential generators. Addition, subtraction, bit mixing. Better randomness, small period.

- higher order recursions

Blum-Blum-Shrub

- $p, q \in \mathbb{P}$, large (at least 40 decimal places)

- $m = pq$

- $X_i = X_{i-1}^2 \pmod m$

- $b_i = \text{parity}(X_i)$

Amortized analysis

Aggregated analysis

Determine upper bound $T(n)$ for the total cost of a sequence of n operations. Amortized cost per operation is $\frac{T(n)}{n}$.

Accounting method

Some operations are overcharged to pay for other operations.

Potential method

Potential function is tied to a data structure.

NP-complete problems

- CSAT – logical circuit satisfiability

- FSAT – logical formula satisfiability

- 3CNF-SAT – formula in 3-conjunctive normal form satisfiability

- CLIQUE – existence of cliques in a graph

- VERTEX COVER – a minimal set of vertices that cover all the edges of a graph

- HAM – Hamiltonian cycle of a graph

- TSP – travelling salesman problem

- SUBSET-SUM – the subset of numbers equal to a given number

- BIN-TREE – optimal binary decision tree

- SUBGRAPH-ISOMORPHISM

Linear programming

Standard LP

- given n real numbers c_1, c_2, \dots, c_n

- m real numbers b_1, b_2, \dots, b_m

- $m \times n$ real numbers a_{ij} for $i = 1, \dots, m$ and $j = 1, \dots, n$

- we wish to find n real numbers x_1, \dots, x_n that

maximize $\sum_{j=1}^n c_j x_j$ subject to

$$\sum_{j=1}^n a_{ij} x_j \leq b_i, \forall i = 1, \dots, m$$

$$x_j \geq 0$$

Approximation

LP relaxation, 0-1 integer programming

Local search

- State space: $S = \{S; S_Z \longrightarrow S\}$

- starting state: S_0

- quality of state: $q(S)$

- global optimum: $S_{\text{best}} = \arg \min_{s \in S} q(s)$

- local optima: $S_{\text{local}} = \{S; \forall S \rightarrow S' : q(S) \leq q(S')\}$

Problems

- local extremes

- plato

- ridge

Metroplis algorithm

- If better neighbour exists, move to it.

- Otherwise choose a random neighbour, but accept better neighbours with larger probability.

- Decrease the probability of acceptance.

- In time, stohastic search turns into deterministic LS.

Simulated annealing

- Start with a random state S .

- Select random neighbour S'

- If $q(S') < q(S)$, move to S' .

- Otherwise, move with probability $e^{\frac{-(q(S')-q(S))}{T}}$

Decrease temperature while it's not close to zero. Usually a geometrical rule is used: $T' = \lambda T$, $0 < \lambda < 1$ (typically $\lambda = 0.95$)

Metaheuristics

Tabu search

Idea: to prevent returning back to the same local extreme, supress (parts of) solutions.

Guided local search

Metaheuristics which guide local search and helps it avoid local extremes.

- define properties (attributes) of solutions

- penalize attributes, which occur too often in local extrema

- auxiliary objective function

$$h(s) = g(s) + \lambda \cdot \sum_{i \text{ is a feature}} (p_i \cdot I_i(s))$$

Utility of punishment for property i in local extreme s^*

$$\text{util}_i(s^*) = I_i(s^*) \cdot \frac{c_i}{1 + p}$$

c_i is cost, p_i is current punishment for property i

In local extreme we punish the property with the largest utility (we increment p_i by 1).

Variable neighbourhood search

Idea: define several neighbourhood structures and change neighbourhood when reaching local extreme in one of them. Order neighbourhoods by the efficiency of computation.