## Computational complexity

### Tight bound $\Theta$

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

### Upper bound $\mathcal{O}$

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

### Lower bound $\Omega$

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

### Imprecise boundaries $o$ and $\omega$

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

### 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 $\Theta$, $\mathcal{O}$ and $\Omega$)
- 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

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

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

## Masters

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

- $f(n) = \mathcal{O}(n^{log_b a - \epsilon}) \rightarrow T(n) = \Theta(n^{\log_b a}), \epsilon > 0$
- $f(n) = \Theta(n^{log_b a}) \rightarrow T(n) = \Theta(n^{log_b a} \log(n))$
- $f(n) = \Omega(n^{log_b a + \epsilon}) \rightarrow T(n) = \Theta(f(n)), \epsilon > 0$ and $af(\frac{n}{b}) \le cf(n)$ 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

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

- $n_0 \ge \frac{1}{b_i}$, $n_0 \ge \frac{1}{1-b_i}$ for each $i$,
- $a_i > 0$ for each $i$,
- $0 < b_i$ for each $i$,
- $k \ge 1$,
- $f(n)$ is non-negative function,
- $c_1 f(n) \le f(u) \le c_2 f(n)$, for each $u$ satisfying condition: $b_i n \le u \le n$

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

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 |
|---|---|
| $E-1$ | $\alpha$ |
| $E-a$ | $\alpha a^n$ |
| $(E-a)(E-b)$ | $\alpha a^n + \beta b^n \quad (a \ne b)$ |
| $(E-a_0)(E-a_1)\cdots(E-a_k)$ | $\sum_{i=0}^{k} \alpha_i a_i^n \quad (a_i \text{ distinct})$ |
| $(E-1)^2$ | $\alpha n + \beta$ |
| $(E-a)^2$ | $(\alpha n + \beta)a^n$ |
| $(E-a)^2(E-b)$ | $(\alpha n + \beta)a^n + \gamma b^n (a \ne b)$ |
| $(E-a)^d$ | $(\sum_{i=0}^{d-1} \alpha_i n^i)a^n$ |

If $X$ annihilates $f$, then $X$ also annihilates $Ef$.
If $X$ annihilates both $f$ and $g$,
then $X$ also annihilates $f \pm g$.
If $X$ annihilates $f$, then $X$ also annihilates $\alpha f$,
for any constant $\alpha$.

If $X$ annihilates $f$ and $Y$ annihilates $g$,
then $XY$ annihilates $f \pm g$.

## 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: $x_i = 65539 x_{i-1} \pmod{2^{31}}$
- MINSTD $x_i = 16807 x_{i-1} \pmod{2^{31} - 1}$
- 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-conjuctive 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 $c_1, c_2, \ldots, c_n$
- $m$ real numbers $b_1, b_2, \ldots, b_m$
- $m \times n$ real numbers $a_{ij}$ for $i = 1, \ldots, m$ and $j = 1, \ldots, n$
- we wish to find $n$ real numbers $x_1, \ldots, x_n$ that

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

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

$$x_j \ge 0$$

### 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: $S_{\text{local}} = \{S; \forall S \rightarrow S\prime : q(S) \le q(S\prime)\}$

### 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 $S\prime$
- If $q(S\prime) < q(S)$, move to $S\prime$.
- Otherwise, move with probability $e^{\frac{-(q(S\prime) - q(S))}{T}}$

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