Asymptotic Analysis

- Worst Case, Average Case (involves statistics), Best Case
- We compare efficiency for asymptotically large values of input size

O-notation: $f(n) \in O(g(n))$ if $\exists$ constants $c > 0, n_0 > 0$ s.t. $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$

- If $\lim \frac{f(n)}{g(n)} < \infty$, then $f(n) \in O(g(n))$.

Ω-notation: $f(n) \in \Omega(g(n))$ if $\exists$ constants $c > 0, n_0 > 0$ s.t. $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$

- If $\lim \frac{f(n)}{g(n)} > 0$, then $f(n) \in \Omega(g(n))$.

Θ-notation: $f(n) \in \Theta(g(n))$ if $\exists$ constants $c_1 > 0, c_2 > 0, n_0 > 0$ s.t. $0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n)$ for all $n \geq n_0$

o-notation: $f(n) \in o(g(n))$ if for all constants $c > 0$, there exists a constant $n_0 > 0$ s.t. $0 \leq f(n) < cg(n)$ for all $n \geq n_0$

- $\lim \frac{f(n)}{g(n)} = 0$ if and only if $f(n) \in o(g(n))$.

ω-notation: $f(n) \in \omega(g(n))$ if for all constants $c > 0$, there exists a constant $n_0 > 0$ s.t. $0 \leq cg(n) < f(n)$ for all $n \geq n_0$

- $\lim \frac{f(n)}{g(n)} = \infty$ if and only if $f(n) \in \omega(g(n))$.

Properties

- Transitive: $\Theta, O, \Omega, o, \omega$
- Reflexive: Θ, O, Ω
- Symmetry: $f(n) \in \Theta(g(n))$ iff $g(n) \in \Theta(f(n))$.
- Complementary: Θ with O. o with ω.

Useful facts

- $e^x \geq 1 + x$
- $n^k = o(a^n)$
- (Stirling Approx). $n! = \sqrt{2\pi n}\left(\frac{n}{e}\right)^n\left(1 + \frac{c}{n}\right)$ for some constant c.
- $\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{en}{k}\right)^k$
- $\lg(n!) = \Theta(n \lg n)$
- $\sum\limits_{k=\lfloor n/2 \rfloor}^{n-1} \leq \frac{3n^2}{8}$

- $H_n = \sum\limits_{k=1}^{n} \frac{1}{k} = \ln n + O(1)$.
- (L'Hopital) $\lim\limits_{x\to\infty} \frac{f(x)}{g(x)} = \lim\limits_{x\to\infty} \frac{f'(x)}{g'(x)}$

Recurrences

- Telescoping method
  - Try to create the form $a_n - a_{n-1}$.
  - Dividing $T(n)$ by some function $g(n)$ and then solving might be a good idea.
- Recursion tree: Work per level x number of level. Questions like $T(n) = T(n-a) + T(a) + cn$ is suited for this method → $T(n) \leq C(\frac{n^2}{a})$.
- Substitution method: Guess form + prove by induction
- Master's method

Master's Theorem

Preconditions: Applies to recurrences of the form $T(n) = aT(\frac{n}{b}) + f(n)$ where $a \geq 1, b > 1$ and f is asymptotically positive.

1. $\Theta(n^{\lg_b a})$ if $f(n) = O(n^c)$ where $c < \lg_b a$
2. $\Theta(n^{\lg_b a} \lg^{k+1} n)$ if $f(n) = \Theta(n^c)$ where $c = \lg_b a$
3. $\Theta(f(n))$ if $f(n) = \Omega(n^c)$ where $c > \lg_b a$ and $f(n)$ satisfies regularity condition, i.e. $af(\frac{n}{b}) \leq cf(n)$ for some $c < 1$.

Correctness

Of an iterative algorithm

Loop invariant:

1. Initialization: True before first iteration
2. Maintenance: If true before of an iteration, and remains true at the beginning of the next iteration.
3. Termination: The invariant provides useful property for showing correctness after algorithm terminates

Prove 1 and 2, to get 3

Of a recursive algorithm

Usually use strong induction on size of problem.

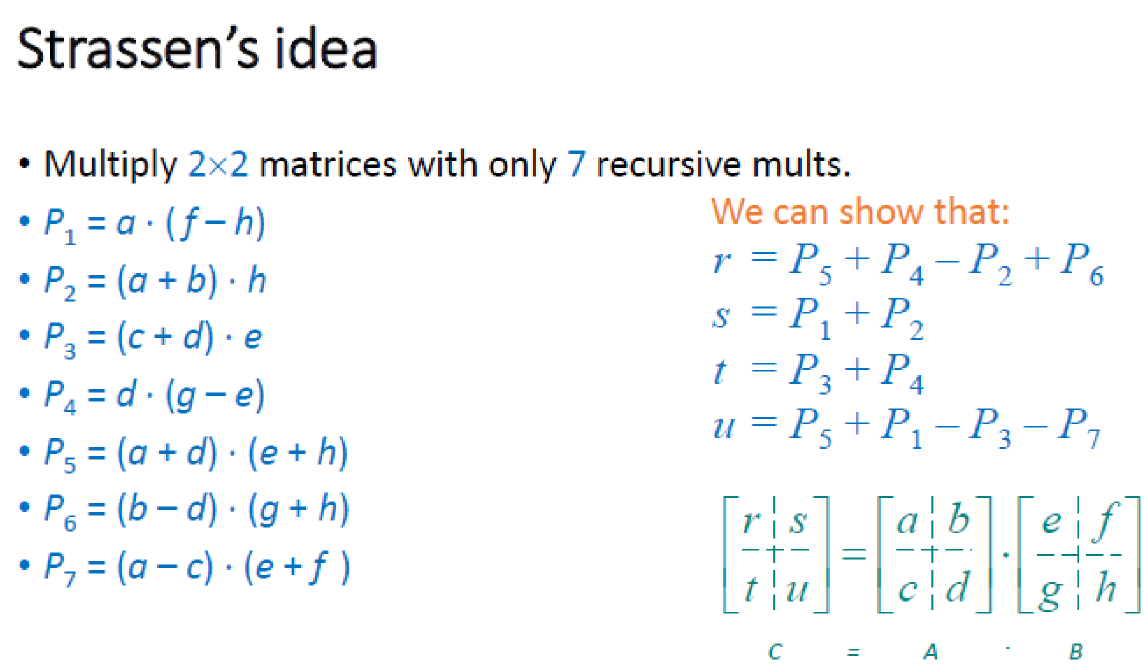
Divide and Conquer

in 3 simple steps

1. Divide the problem (instance) into subproblems
2. Conquer subproblems by solving recursively
3. Merge subproblem solutions

Problems:

- Binary search, merge sort
- Fast Power : $\Theta(logn)$
    - $f(n,m) = \begin{cases} f(\lceil n/2 \rceil, m)^2 & \text{for n even} \\ f(1,m) \cdot f(\lceil n/2 \rceil, m)^2 & \text{for n odd} \end{cases}$
- Fibonacci → $\Theta(logn)$ by viewing $F_n = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$.
- Strassen → better than $\Theta(n^3)$ standard matrix multiplication.



Sorting

Note: quick sort is avg-case fast

Properties

1. In-place: use very little additional memory, usually $O(1)$ or $O(\lg n)$.
    - Insertion sort, Quicksort ($O(\lg n)$ additional memory with proper implementations)
2. Stable: If the original order of equal elements is preserved in sorted output

- Insertion sort, Merge sort

Decision trees

A binary-tree like model where every node is a comparison, every branch represents the outcome of the comparison, and every leaf represents the final decision after all comparisons.

Lower bound for comparison based sorting

We can use decision tree to model any comparison based algorithm. Since the decision tree must contain $\geq n!$ leaves, the height of the tree (the number of comparison needed to be made) is at least $\lg(n!) = \Theta(n \lg n)$ via Stirling's approximation.

Linear time sort

Counting sort → $\Theta(n + k)$ where k is the number of bucket.

1. Set C[i] to be the number of elements equal i
2. Set C[i] to be number of elements <= i
3. Move elements equal i to B[C[i - 1] + 1...C[i]] (from right to left, to maintain stability)

Radix sort : digit by digit sort starting from least-significant digit first w/ auxiliary stable sort → $\Theta(dn)$ where numbers are bounded within a n^d interval.

Randomized Algorithm

- **Las Vegas** Algorithms: always correct, but RV running time
- **Monte Carlo** algorithms: output might be incorrect with some probability, but running time is deterministic.

Tactic:

- Use indicator variables → $C_{i,j} = 1$ if some event happened, otherwise 0.
- Linearity of expectation $E(X + Y) = E(X) + E(Y)$.

Example:

- M Ball and N Bins
  - Probability that a bin x is empty: $(1 - \frac{1}{N})^M$
  - What is the expected number of empty bins: $N(1 - \frac{1}{N})^M$
- Randomized Quick Sort

- Probability that e_i and e_j is compared : $\frac{2}{j-i+1}$. (basically only if either e_i or e_j is first to be selected as a pivot within interval $[i, j]$)
- Number of comparisons: $\sum\limits_{i<j} Y_{i,j}$ where $Y_{i,j} = 1$ if e_i and e_j is compared.