Algorithms:

by Robert Sedgewick

Textbook:

- Personal Preference
- program design
  implementation details
- Pascal language (C, C++, Modula 3, Java)

Goal:

- Survey most important algorithms
- Ability to design clear & efficient algorithms

Algorithm Development

- design
  - working version (simple)
  - program optimization : inner loop
    recursion removal
    :
- implementation
  - simple, inefficient ?
  - simple, efficient .
  - complicated, efficient ?
- mathematical analysis
  - average / worst case complexity $\Leftarrow$ recurrent relation
  - upper / lower bound :
  - asymptotic analysis : $H_N = \frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{N}$
    $= \ln N + \gamma + O(\frac{1}{N})$
- empirical analysis
  - benchmarks

Algorithm Design — a case study

Find $gcd(u, v)$ = greatest common divisor of u, v

$gcd(24, 60) = 12$.

(1) Brute-force

```
t := min(u, v);
while (u mod t ≠0) or (v mod t ≠0) do t:= t-1;
gcd := t;
```

(2) Euclid's algorithm (輾轉相除法)

- mathematics: $(u, v) = (v, u-v)$
  $= (v, u-2v)$
  ⋮
  $= (v, u \bmod v)$

- Recursive Program:

```
if v=0 then
    gcd := u;
else gcd := gcd(v, u mod v);
```

1	3240	2898	8
	2898	2736	
2	342	162	
	324	162	
	18	0	

(3240, 2898)
(2898, 342)
(342, 162)
(162, 18)
(18, 0)

- Recursion: call itself
  - reduced to small problem — simple, clear
  - termination condition — tricky, inefficient
  - programming environment (stack)

NOT in FORTRAN.

(3) Recursion Removal

- Non recursive program

```
While (v≠0) {
    t := u mod v;
    u := v;
    v := t;
}
gcd := u;
```

- recursion removal
  - easy with single recursion
  - more efficient

Analysis of Algorithms

- Compare different programs for resources used
  - Memory : $S(N)$ N: input size
  - Computing time : $T(N)$
  - Plotter distance
    ⋮
  - Average case ↔ Worst case
    ↑
    typical case
  - space/time tradeoff

- Example : gcd
  - Brute-force : $S = 3$
    $T(u,v) = \min(u,v) - \gcd(u,v) + 1$
  - Euclid's algorithm :
    - (overhead for recursive call) * (# recursive calls)
      - language features
    - Complicated

$$\frac{12 \ln 2 \ln V}{\pi^2}$$

"90% CPU time on 10% code" !!!

- Leads to better understanding of the program.
- " improvements to the program.

Program complexity

$= a N \log N + bN + C$

$= \underbrace{a N \log N}_{\text{leading term}} + O(N)$

- 1 constant : each statement constant times

$\log N$ log : $C_N = C_{\frac{N}{2}} + 1 \Rightarrow C_N = \lg N$, $C_1 = 0$ — each step half input

N linear
- $C_N = C_{\frac{N}{2}} + N \Rightarrow C_N = 2N$
- $C_N = 2C_{\frac{N}{2}} + 1 \Rightarrow C_N = 2N$ — divide + conq.

$N \log N$
- $C_N = 2C_{\frac{N}{2}} + N \Rightarrow C_N = N \log N$

N^2 quadratic
- $C_N = C_{N-1} + N$ (recurrent relation)
  $= N + N-1 + N-2 + \cdots + 1 = \frac{1}{2} N(N+1)$

N^3 cubic

$\vdots$

2^N exponential

- Common Mistakes
  - Ignore performance : large N → efficient progm.
  - Too concerned performance : Small N → Simple program

$N^2 \longrightarrow N \log N$?

Elementary Data Structures

- data structures : (stack)

  implementation : array, linked list,

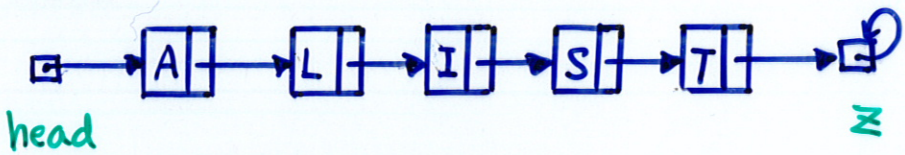
  \+

  operations : PUSH, POP

  : abstract data type

  - cannot access data directly, only thru operations
  - modular, simple interface

- Arrays : a[1..N], a[i]
- Linked lists



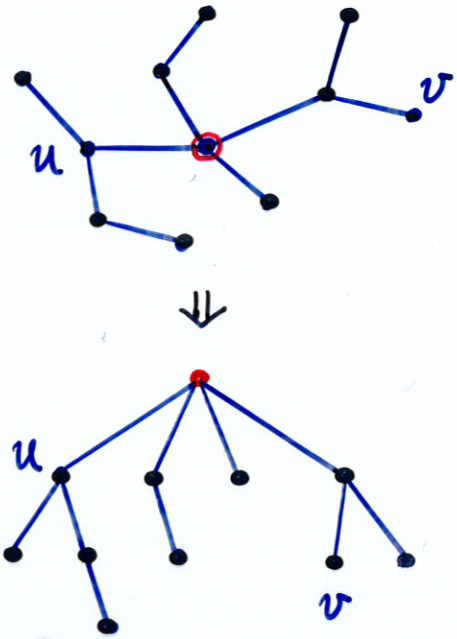
  - circular linked lists
  - doubly linked lists

- Stack FILO
- Queue FIFO

Tree : undirected, connected, cycless graph

$\Leftrightarrow$ " , $\exists$ 1 path between any nodes u, v

- rooted (oriented) tree
  free tree
- root
  parent, children, sibling
- terminal node = leaf (external)
  nonterminal node (internal)
- level
  height
  internal path length, $I = 0+1+1+1+2$
  external " , $E = 2+3+2+1+2+2$

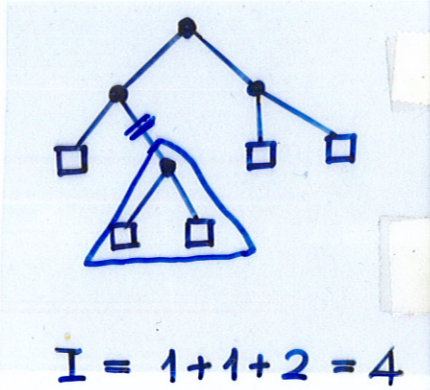


- binary tree : 0 or 2 children

- Property : N nodes trees $\Rightarrow$ $N-1$ edges

- Property : Binary trees with N internal nodes

$$\Rightarrow \begin{cases} (1)\ N+1 \text{ external nodes} \\ (2)\ E = I + 2N \end{cases}$$



$I = 1+1+2 = 4$

$E = 2+3+3+2+2 = 12$

Pf : (1) ext $\leftrightarrow$ player
int $\leftrightarrow$ game

(2) $E = I +$ (# edges)

※ $2 \rightarrow E$
$1 \rightarrow I$

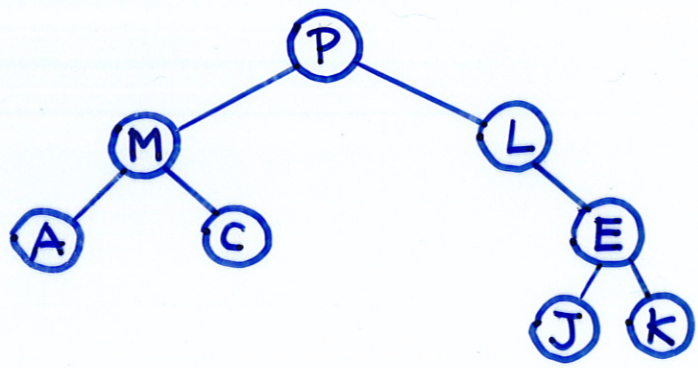
Tree traversal

- preorder
  - recursive = (recursion removal) ⇒ non recursive

```
traverse(t)
    if t ≠ nil then {
        visit(t);
        traverse(t.left);
        traverse(t.right);
    }
```

```
traverse(t)
    push(t);
    repeat
        t := pop();
        visit(t);
        if t.left ≠ nil then
            push(t.right);
        if t.right ≠ nil then
            push(t.left);
    until stack = ∅
```

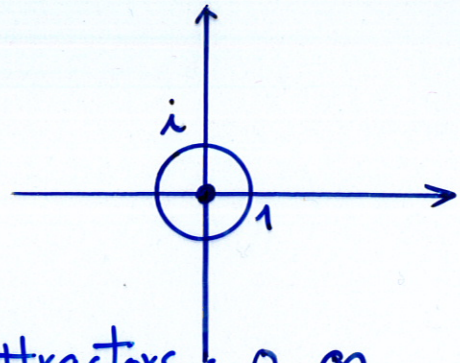
- inorder
- postorder
- level-order



Fractal Geometry

- Mandelbrot process

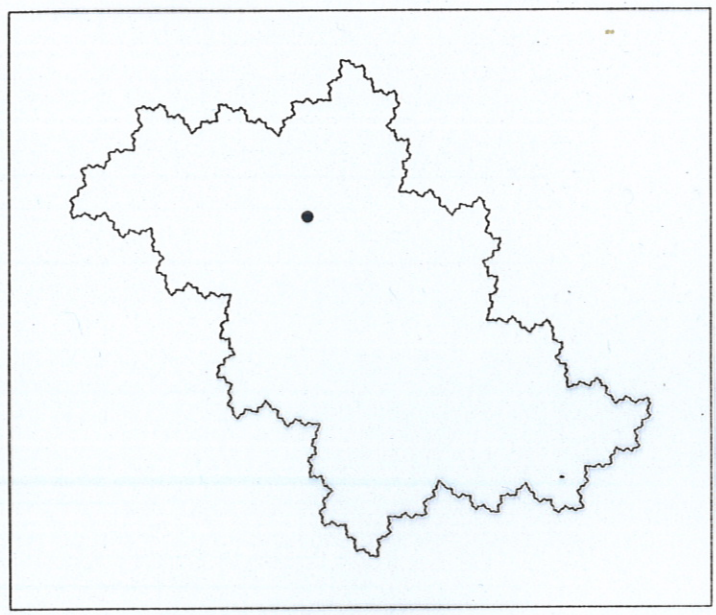
$$z_{n+1} = z_n^2 + C$$



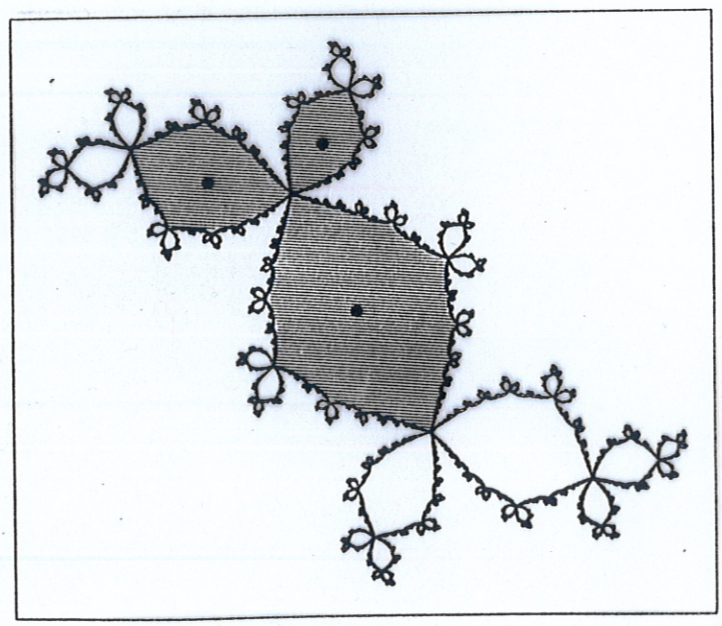
attractors : 0, ∞

- $C = 0$

- $C = -0.12375 + 0.56508i$



- $C = -0.12 + 0.74i$



- break into pieces recursive

Julia set

8

9

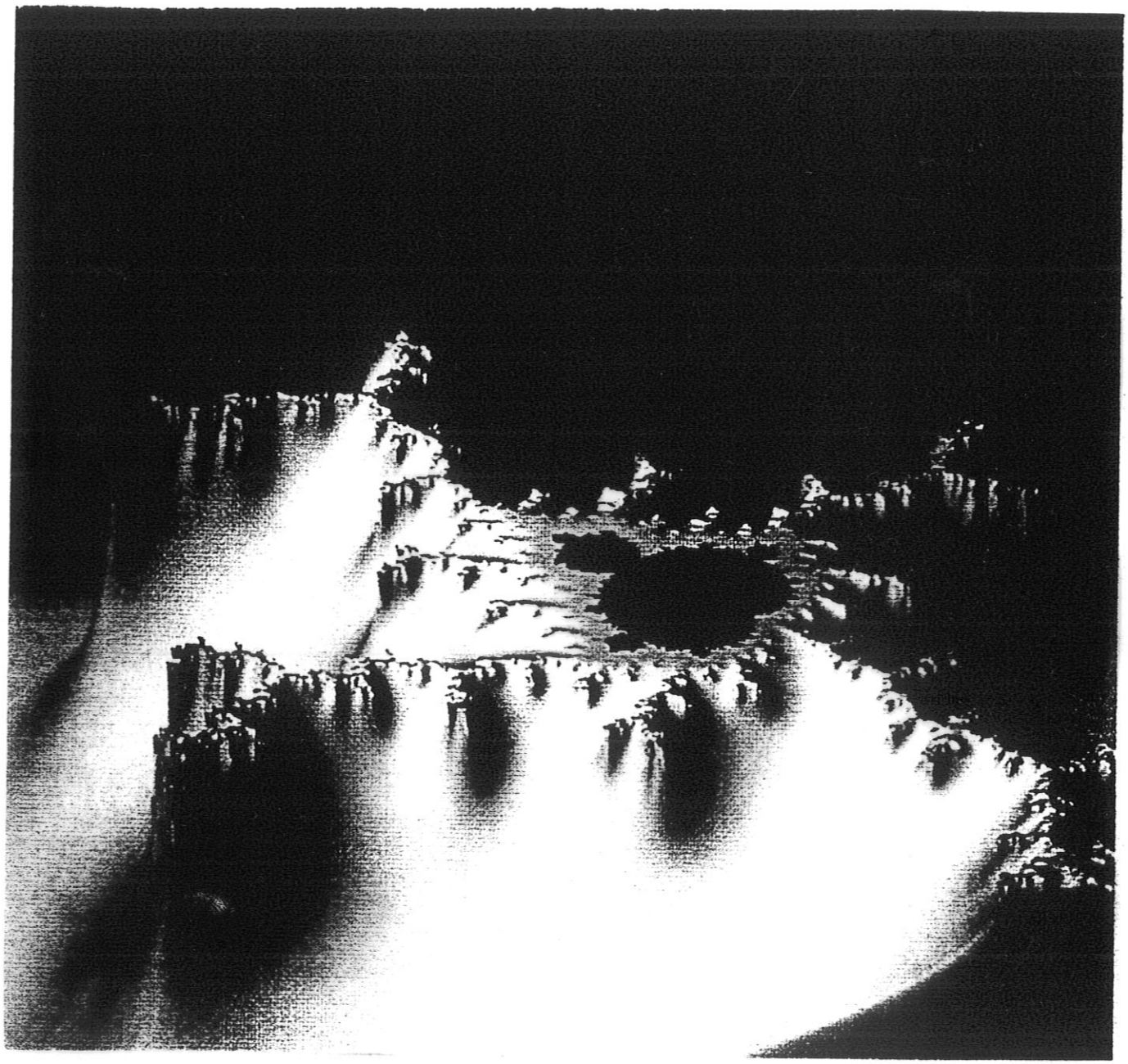
10

11

12

13

Q



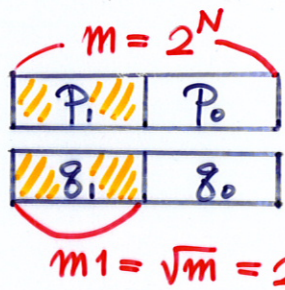
Random Number Generator

- Applications : random sampling, decision making, simulation, ...
- Random number sequence
  - genuine — impossible with computers
  - pseudo-random — some good properties of r.n.
  - quasi-random —
- Linear Congruential method :
  - $\begin{cases} a_0 = \text{seed} \\ a_i = (a_{i-1} * b + 1) \bmod m \end{cases}$

  a_0 : random seed
  m : 2^α, 10^α
  b : 1 digit less than m
  ... x 2 1
  ↑ even
  - Avoid overflow
    - $a_i = \text{mult}(a_{i-1}, b) + 1 \bmod m$

$$\text{mult}(p, q) = \Big(\big((p_0 * q_1 + p_1 * q_0) \bmod m1\big) * m1 + p_0 q_0\Big) \bmod m$$

$p_1 = p \text{ div } m1, \quad p_0 = p \bmod m1$
$q_1 = q \text{ div } m1, \quad q_0 = q \bmod m1$

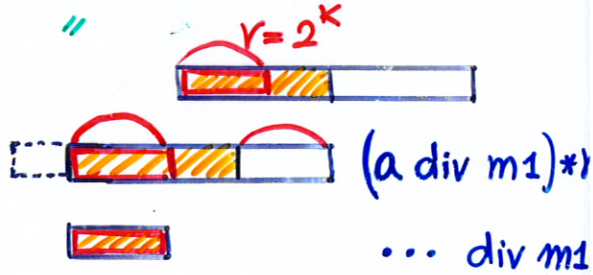


- Example : $m = 10^8$, $m1 = 10^4$

$p = 10^4 p_1 + p_0$
$q = 10^4 q_1 + q_0$ $\Rightarrow pq = 10^8 p_1 q_1 + (p_0 q_1 + p_1 q_0)\, 10^4 + p_0 q_0$

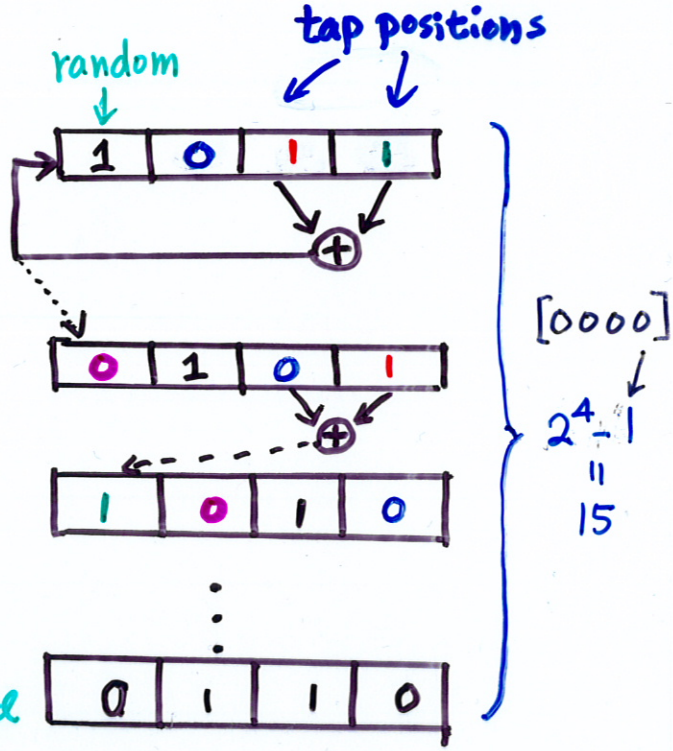
- random number $\in [0, r-1]$ $\quad r = 2^k \le m1$

$$\begin{cases} a = \text{mult}(a, b) + 1 \bmod m \\ \quad (x)\ a \bmod r & : \text{right } K \text{ bits} \\ \quad (x)\ a * r \text{ div } m & : \text{left } \text{〃} \\ \big((a \text{ div } m1) * r\big) \text{ div } m1 \end{cases}$$



- random number $\in [0, 1)$: $\frac{a}{m}$

- **Additive Congruential method**
  - **Linear feedback shift register (LFSR)**



  - **Software Implementation:**

$$a[k] := \left(a[k-b] \underset{\oplus}{+} a[k-c]\right) \bmod m$$

$b = 24$

$c = 55$

  - Need a table size 55 initially generated by Linear Congruential

a_{55} $[a_{54} \cdots \underline{a_{31}} \cdots a_2\ a_1\ \underline{a_0}]$

$[a_{54} \cdots a_{31} \cdots a_2\ a_1\ a_{55}]$ $\quad a_0 \leftarrow a_{55} := (a_0 + a_{31}) \bmod m$

$[a_{54} \cdots a_2\ a_{56}\ a_0]$ $\quad a_1 \leftarrow a_{56} := (a_1 + a_{32}) \bmod m$

$\vdots$

  - $a[j] := \left(a[j] + a[(j+31) \bmod 55]\right) \bmod m$

- **Implementation Note:**
  - 2 generators: one generates the table, another picks #

• Testing Randomness

• χ^2-test :

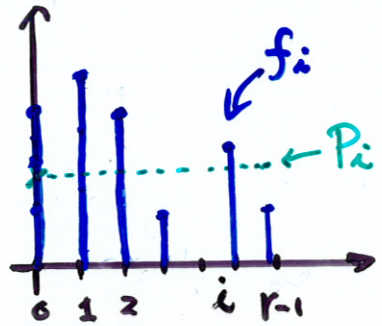
$\# \in [0, r-1]$

$$\chi^2 = \sum_{0 \le i < r} \frac{(f_i - P_i)^2}{P_i}$$

$$= \sum_i \frac{f_i^2}{P_i} - 2f_i + P_i$$

$$= \left[\sum_i \frac{r}{N} f_i^2\right] - 2N + N$$

$$= \frac{r}{N} \sum_{0 \le i < r} f_i^2 - N$$



- $P_i = \frac{N}{r}$
- $f_i = (\#\ r.n. = i)$
- Totally N r.n.
  i.e., $f_0 + \cdots + f_{r-1} = N$

$\chi^2 - r \le 2\sqrt{r}$ probably "Random" !

$>$ No !!

• Spectral-test :

⋮

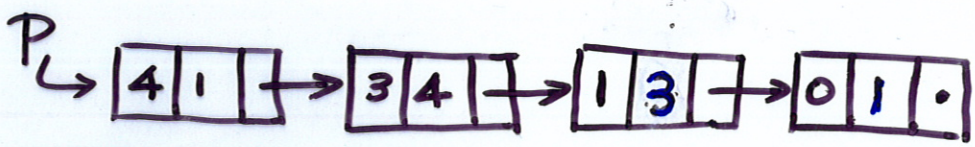
Polynomials & Matrices

- **Polynomial**: $P(x) = x^4 + 4x^3 + 3x + 1$

  $P_i x^i$ ↕ $P[i]$

  array: $P[0..4] = [1, 3, 0, 4, 1]$

  linked list:



- **Matrix**:

  array: $M[i, j] \leftrightarrow M_{ij}$

  Doubly linked list: (Sparse matrix)

Polynomials $p(x) = a_n x^n + \cdots + a_1 x + a_0$

$$\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{bmatrix} \xrightarrow{p(x)} \begin{bmatrix} y_1 = p(x_1) \\ y_2 = p(x_2) \\ \vdots \\ y_N = p(x_N) \end{bmatrix}$$

- Compute y: evaluation

  p: interpolation

  x: find root : Newton's method, ...

- **Evaluation** $y = p(x) = 2x^4 + 3x^3 - 6x^2 + 2x + 1$

- **Brute-force**:

  $2x^4$, $3x^3$, $(-6)x^2$, ... "*": $4+3+2+1 = \frac{N}{2}(N+1)$

  $(2x^4+3x^3)+$ .... "+": $4 = N$

  Problems: x^3, x^4 重复计算.

- **Horner's rule**:

$$p(x) = x\cdot\Big(x\big(x\cdot(x\cdot 2+3)-6\big)+2\Big)+1 \quad : 4\text{"*"} + 4\text{"+"}$$

- **FFT** (Fast Fourier Transforms in Chap. 41)

  can evaluate at N points with $N \log N$ "*"

- x^N only: (Horner's rule: $(N-1)$ "*")

$$x^{55} = x^{[1\,1\,0\,1\,1\,1]_2}$$

$$= x^{32}\cdot x^{16}\cdot x^4\cdot x^2\cdot x^1 \qquad L \leftarrow R$$

$$= x^{2^5+2^4+0\cdot 2^3+2^2+2^1+1} \qquad \text{Horner's rule}$$

$$= x^{((((1\cdot 2+1)\cdot 2+0)2+1)2+1)2+1} \qquad L \rightarrow R$$

$$= \Big(\big(\big(\big(\big((1^2\cdot x)^2\cdot x\big)^2\cdot 1\big)^2\cdot x\big)^2\cdot x\big)^2\cdot x$$

$x^{a\cdot b} = (x^a)^b$

$x^{a+b} = x^a \cdot x^b$

$55 = [1, 1, 0, 1, 1, 1]_2$

$Y = 1$

x

x^3

x^7

x^7

x^{23}

x^{55}

$Y = Y\cdot x^{32}$

$Y = Y\cdot x^{16}$

$Y = Y\cdot 1$,

$Y = Y\cdot x^4$,

$Y = Y\cdot x^2$;

$Y = Y\cdot x$,

$Y = 1$

Interpolation : $p(x) = ?$

such that $p(x_1) = y_1, \ldots, p(x_N) = y_N$

- **Lagrange** method :

$$p(x) = \sum_{1 \le j \le N} y_j \underbrace{\frac{(x-x_1)\cdots(\hat{x-x_j})\cdots(x-x_N)}{(x_j-x_1)\cdots(\hat{x_j-x_j})\cdots(x_j-x_N)}}_{\begin{cases} = 0 & x \neq x_j \\ 1 & x = x_j \end{cases}}$$

Example : $p(1)=3,\ p(2)=7,\ p(3)=13.$

$$p(x) = 3 \cdot \frac{(x-2)(x-3)}{(1-2)(1-3)} + 7 \cdot \frac{(x-1)(x-3)}{(2-1)(2-3)} + 13 \cdot \frac{(x-1)(x-2)}{(3-1)(3-2)}$$

$$= x^2 + x + 1$$

$$p(1) = 3 \cdot 1 + 7 \cdot 0 + 13 \cdot 0 = 3$$

$$p(2) = 3 \cdot 0 + 7 \cdot 1 + 13 \cdot 0 = 7$$

$$p(3) = 3 \cdot 0 + 7 \cdot 0 + 13 \cdot 1 = 13$$

- $O(N^2)$ operations

Multiplication

$$p(x) = p_0 + p_1 x + \cdots + p_{N-1} x^{N-1}$$
$$q(x) = q_0 + q_1 x + \cdots + q_{N-1} x^{N-1}$$

$p(x) \cdot q(x) = ?$

- **Brute-force:** N^2 "$*$"

- **Divide + Conquer**

$$p(x) = \left(p_0 + p_1 x + \cdots + p_{\frac{N}{2}-1} x^{\frac{N}{2}-1}\right) + x^{\frac{N}{2}}\left(p_{\frac{N}{2}} + p_{\frac{N}{2}+1} x + \cdots p_{N-1} x^{\frac{N}{2}-1}\right)$$
$$= p_e(x) + x^{\frac{N}{2}} p_h(x)$$
$$q(x) = q_e(x) + x^{\frac{N}{2}} q_h(x)$$

$$\Rightarrow p(x) \cdot q(x) = p_e \cdot q_e + \underbrace{\left(p_e \cdot q_h + p_h \cdot q_e\right)}_{\| \; (p_e + p_h)\cdot(q_e + q_h) - p_e q_e - p_h q_h} x^{\frac{N}{2}} + \left(p_h \cdot q_h\right) x^N$$

- $M(N) :=$ # "$*$" for $p(x) \cdot q(x)$

$$M(N) = 3\,M\left(\tfrac{N}{2}\right) \qquad N = 2^n$$
$$= 3\,M(2^{n-1})$$
$$= 3^2 M(2^{n-2})$$
$$\vdots$$
$$= 3^n M(2^0)$$
$$= 3^n = \left(2^{\log_2 3}\right)^n = \left(2^n\right)^{\log_2 3}$$
$$= N^{1.58}$$

Example: $p(x) = 1 + x + 3x^2 - 4x^3 = \underbrace{(1+x)} + x^2 \underbrace{(3-4x)} = p_e + x^2 p_h$

$q(x) = 1 + 2x - 5x^2 - 3x^3 = \underbrace{(1+2x)} + x^2 \underbrace{(-5-3x)} = q_e + x^2 q_h$

Large numbers

- $p = 0120\ 2001\ 0311\ 0001\ 2000\ 0401\ 2314$

$$p(x) = x^{26} + 2x^{25} + 2x^{23} + \cdots + x^4 + 2x^3 + 3x^2 + x + 4 \Big|_{x=10}$$

$$P(x) = 120x^6 + 2001x^5 + 311x^4 + x^3 + 2000x^2 + 401x + 2314 \Big|_{x=10000}$$

- $p \cdot q = p(x) \cdot q(x) \Big|_{x=10}$

$= P(x) \cdot Q(x) \Big|_{x=10000}$

Example: $p = 4385$ $\quad q = 6857$

$p(x) = 4x^3 + 3x^2 + 8x + 5$ $\quad q(x) = 6x^3 + 8x^2 + 5x + 7$

$p(x) \cdot q(x) = 24x^6 + 50x^5 + 92x^4 + 137x^3 + 101x^2 + 81x + 35$

		60	106	147	109	84
$p \cdot q =$	30	0	6	7	9	45

- $(p \cdot q) \bmod N \quad (p^k \bmod N)$

Theorem: $(x+y) \bmod N \equiv x \bmod N + y \bmod N \pmod N$

$(x * y) \bmod N \equiv (x \bmod N) * (y \bmod N) \pmod N$

$$p \bmod N = 120(x^6 \bmod N) + 2001(x^5 \bmod N) + 311(x^4 \bmod N) + (x^3 \bmod N) + 2000(x^2 \bmod N) + 401(x \bmod N) + 2314 \pmod N$$

Strassen's algorithm

- $\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$

$c_{11} = a_{11}b_{11} + a_{12}b_{21}$
c_{12} :
c_{21} :
c_{22} :

$m_1 = (a_{12} - a_{22})(b_{21} + b_{22})$
$m_2 = (a_{11} + a_{22})(b_{11} + b_{22})$
$m_3 = (a_{11} - a_{21})(b_{11} + b_{12})$
$m_4 = (a_{11} + a_{12})\, b_{22}$
$m_5 = a_{11}\,(b_{12} - b_{22})$
$m_6 = a_{22}\,(b_{21} - b_{11})$
$m_7 = (a_{21} + a_{22})\, b_{11}$

$\Rightarrow$ $c_{11} = m_1 + m_2 - m_4 + m_6$
$c_{12} = m_4 + m_5$
$c_{21} = m_6 + m_7$
$c_{22} = m_2 - m_3 + m_5 - m_7$

- Compute $[a_{ij}]_{N\times N} \cdot [b_{ij}]_{N\times N} = [c_{ij}]_{N\times N}$

  - Brute-force: $c_{ij} = a_{i1}\cdot b_{1j} + a_{i2}\cdot b_{2j} + \cdots + a_{iN} b_{Nj}$
    $1 \le i, j \le N$

    N^3 integer "*"

  - Strassen's algorithm:

    $C_{11} = A_{11}B_{11} + A_{12}B_{21}$
    :

$$\left[\begin{array}{c|c} A_{11} & A_{12} \\ \hline A_{21} & A_{22} \end{array}\right] \left[\begin{array}{c|c} B_{11} & B_{12} \\ \hline B_{21} & B_{22} \end{array}\right]_{N\times N} = \left[\begin{array}{c|c} C_{11} & C_{12} \\ \hline C_{21} & C_{22} \end{array}\right]_{N\times N}$$

    $M_1, M_2, \cdots, M_7, \cdots$

    $M(N) = 7\, M\left(\frac{N}{2}\right) = \cdots$
    $= N^{\lg 7} = N^{2.81}$

Gaussian Elimination

- Solve simultaneous equations

$$\begin{cases} x_1 + 3x_2 - 4x_3 = 8 \\ x_1 + x_2 - 2x_3 = 2 \\ -x_1 - 2x_2 + 5x_3 = -1 \end{cases} \Rightarrow \begin{pmatrix} 1 & 3 & -4 \\ 1 & 1 & -2 \\ -1 & -2 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 8 \\ 2 \\ -1 \end{pmatrix}$$

(1) Forward Elimination (triangulation)

$$\begin{aligned} &x_1 + 3x_2 - 4x_3 = 8 \\ \rightarrow\ &-2x_2 + 2x_3 = -6 \quad (-1,\ +1) \\ &x_2 + x_3 = 7 \end{aligned} \rightarrow \begin{aligned} &x_1 + 3x_2 - 4x_3 = 8 \\ &-2x_2 + 2x_3 = -6 \\ &2x_3 = 4 \end{aligned}$$

(2)

a) Backward Substitution

$$x_3 = \frac{4}{2} = 2$$

$$x_2 = -\frac{1}{2}(-6 - 2 \cdot 2) = 5$$

$$x_1 = (8 - (-4) \cdot 2 - 3 \cdot 5) = 1$$

b) Gauss-Jordan Reduction

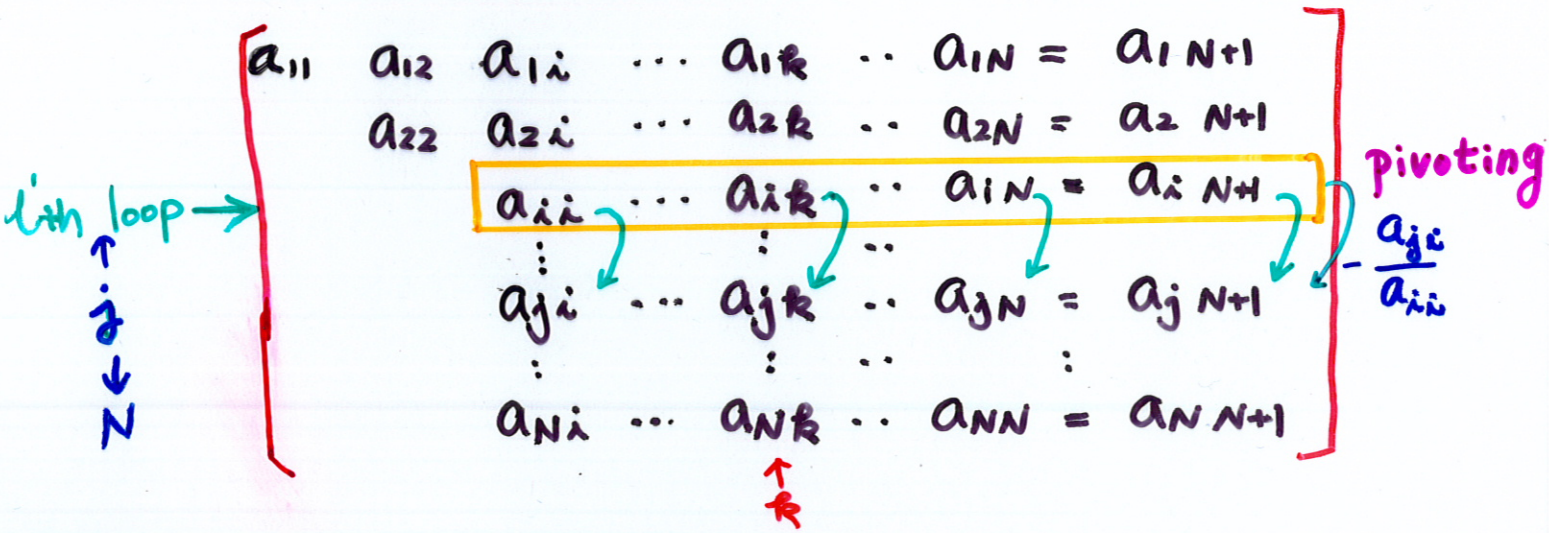
$$\begin{aligned} &x_1 + 3x_2 = 16 \\ \rightarrow\ &-2x_2 = -10 \\ &2x_3 = 4 \end{aligned} \rightarrow \begin{aligned} &x_1 = 1 \\ &-2x_2 = -10 \\ &2x_3 = 4 \end{aligned}$$

$$\rightarrow \quad x_1 = 1, \quad x_2 = 5, \quad x_3 = 2$$

General method

$$\begin{pmatrix} a_{11} & \cdots & a_{1N} \\ a_{21} & \cdots & a_{2N} \\ \vdots & & \\ a_{N1} & \cdots & a_{NN} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_N \end{pmatrix} \leftrightarrow (a_{ij})_{N \times (N+1)}$$

Forward Elimination:



Program

$i = 1$ to N ← (*)

$j = i+1$ to N

$K = N+1$ to i

$$a_{jk} = a_{jk} - \frac{a_{ji}}{a_{ii}} a_{ik}$$

- $T = O(N^3)$
- To avoid overflow

  choose j s.t. $\max_{i \le j \le N} |a_{ji}| \neq 0$ (*)

- $a_{ii} = a_{i+1,i} = \cdots = a_{N,i} = 0$

  $a_{ii} = a_{i,i+1} = \cdots = a_{i,N+1} = 0$ (相依) ⇒ dependent, ∞ Solutions

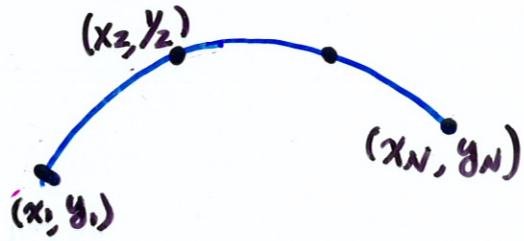
  row = 0, except last one $(a_{i,N+1})$ ⇒ Contradict (矛盾), NO "

- Complexity $AX = B$ = Complexity $A \cdot B$ = Complexity A^{-1} = ...

Curve Fitting

- Given $(x_1, y_1), (x_2, y_2), \cdots, (x_N, y_N)$ N points
  Find Nice-looking curve thru these points

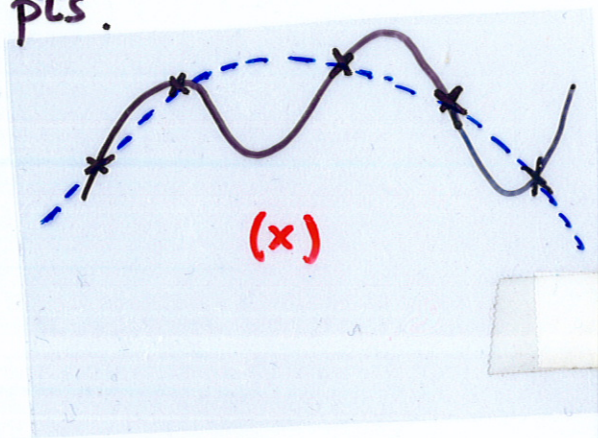
  i.e., $\begin{cases} f(x_i) = y_i & 1 \le i \le N \\ \text{reasonable} & \text{at other points} \end{cases}$



- Polynomial Interpolation

  1 polynomial of degree $N-1$ matches pts.

  - Complicated, fluctuated, slow



- Spline Interpolation
  - from mechanics
  - piece together $N-1$ cubic curves together smoothly

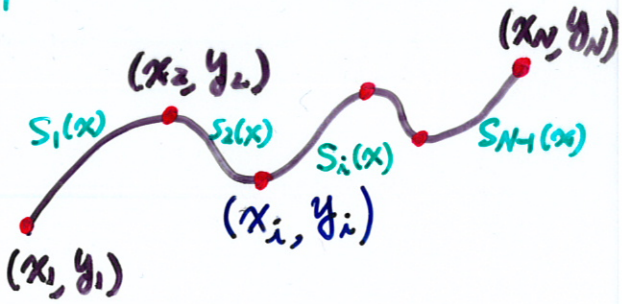
$$S_i(x) = a_i x^3 + b_i x^2 + c_i x + d_i \qquad 1 \le i \le N-1$$

(1) $S_i(x_i) = y_i$
$S_i(x_{i+1}) = y_{i+1}$ $\quad i = 1, 2, \cdots, N-1$

(2) $S'_{i-1}(x_i) = S'_i(x_i) \quad i = 2, \cdots, N-1$

(3) $S''_{i-1}(x_i) = S''_i(x_i) \quad i = 2, \cdots, N-1$

$S''_1(x_1) = 0$
$S''_{N-1}(x_N) = 0$



$4N-4$ unknowns
$4N-4$ eqns

Gaussian Elimination can do

• Better approach

$$t = \frac{x - x_i}{x_{i+1} - x_i}$$

• Theorem

Curve $S_i(t)$

$$\begin{cases} x(t) = (x_{i+1} - x_i)\,t + x_i \\ y(t) = t\,y_{i+1} + (1-t)\,y_i + \frac{1}{6}(x_{i+1} - x_i)^2 \cdot \left[(t^3 - t)\,P_{i+1} + \left((1-t)^3 - (1-t)\right)P_i\right] \end{cases} \qquad 0 \le t \le 1$$

$y(t) = S_i(t)$

$$\Rightarrow (1) \begin{cases} t=0, & x = x_i, \quad y = y_i \\ t=1, & x = x_{i+1}, \quad y = y_{i+1} \end{cases}$$

$$(2) \begin{cases} t=0, & S_i''(x_i) = \frac{d^2y}{dx^2} = P_i \quad \left(= S_{i-1}''(x_i)\right) \\ t=1, & S_i''(x_{i+1}) = P_{i+1} \end{cases}$$

Proof:

$$\frac{dS_i}{dx} = \frac{dt}{dx}\frac{dS_i}{dt} = \frac{y_{i+1} - y_i}{x_{i+1} - x_i} + \frac{1}{6}(x_{i+1} - x_i)\left[(3t^2 - 1)P_{i+1} + (-3(1-t)^2 + 1)P_i\right]$$

$\frac{dt}{dx} = \frac{1}{x_{i+1} - x_i}$; $\frac{y_{i+1} - y_i}{x_{i+1} - x_i} = z_i$

$$(3) \begin{cases} t=0, & S_i'(x_i) = z_i + \frac{1}{6}(x_{i+1} - x_i)\left[-P_{i+1} - 2P_i\right] \\ t=1, & S_i'(x_{i+1}) = z_i + \frac{1}{6}(x_{i+1} - x_i)\left[2P_{i+1} + P_i\right] \end{cases}$$

$$\frac{d^2S_i}{dx^2} = t\,P_{i+1} + (1-t)\,P_i \qquad \begin{cases} t=0, & S_i''(x_i) = P_i \\ t=1, & S_i''(x_{i+1}) = P_{i+1} \end{cases}$$

• Spline curve

$P_1 = P_N = 0$

$S_{i-1}'(1) = S_i'(0) \qquad i = 2, \dots, N-1$

$$\Rightarrow z_{i-1} + \frac{1}{6}(x_i - x_{i-1})\left[2P_i + P_{i-1}\right] = z_i + \frac{1}{6}(x_{i+1} - x_i)\left[-P_{i+1} - 2P_i\right]$$

$$\Rightarrow \underbrace{6(z_i - z_{i-1})}_{w_i} = \underbrace{(x_i - x_{i-1})}_{u_{i-1}}P_{i-1} + 2\underbrace{(x_{i+1} - x_{i-1})}_{d_i}P_i + \underbrace{(x_{i+1} - x_i)}_{u_i}P_{i+1}$$

• $N = 7$, P_i satify

$$\begin{bmatrix} d_2 & u_2 & & & & \\ u_2 & d_3 & u_3 & & & \\ & u_3 & d_4 & u_4 & & \\ & & u_4 & d_5 & u_5 & \\ & & & u_5 & d_6 & \end{bmatrix} \begin{bmatrix} P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \end{bmatrix} = \begin{bmatrix} w_2 \\ w_3 \\ w_4 \\ w_5 \\ w_6 \end{bmatrix} \begin{matrix} i=2 \\ i=3 \\ i=4 \\ \\ i=6 \end{matrix}$$

$P_1 = P_7 = 0$

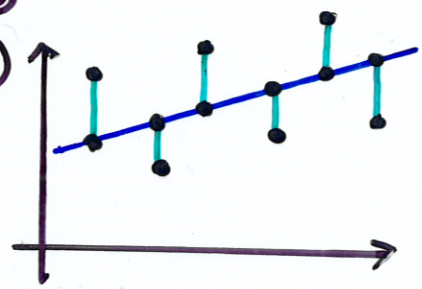
• Solve P_i

Plug in $S_i(t)$ $\quad T = O(N)$

Least Square Method

Problem: Find c_1, c_2 s.t. $f(x) = c_1 + c_2 \frac{1}{x}$ best fits points (x)

$(1.0, 2.5)$ $(2.0, 1.53)$ $(4.0, 1.26)$
$(5.0, 1.21)$ $(8.0, 1.13)$ $(10.0, 1.1)$



i.e., minimize

$$E(c_1, c_2) = \sum_{1 \le j \le 6} \left(f(x_j) - y_j \right)^2$$

3

Solution: $f(x) = c_1 f_1(x) + c_2 f_2(x)$ $\quad f_1(x) = 1$, $\quad f_2(x) = \frac{1}{x}$

$$E = \left(c_1 f_1(x_1) + c_2 f_2(x_1) - y_1 \right)^2 + \left(c_1 f_1(x_2) + c_2 f_2(x_2) - y_2 \right)^2 + \left(c_1 f_1(x_3) + c_2 f_2(x_3) - y_3 \right)^2$$

$$\vec{f}_i = \begin{pmatrix} f_i(x_1) \\ f_i(x_2) \\ f_i(x_3) \end{pmatrix}$$

$$\vec{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$$

$$\frac{\partial E}{\partial c_1} = 0 = 2 \left(c_1 f_1(x_1) + c_2 f_2(x_1) - y_1 \right) \cdot f_1(x_1) + 2 \left(c_1 f_1(x_2) + c_2 f_2(x_2) - y_2 \right) \cdot f_1(x_2) + 2 \left(c_1 f_1(x_3) + c_2 f_2(x_3) - y_3 \right) \cdot f_1(x_3)$$

$$= 2 \left(c_1 \vec{f}_1 + c_2 \vec{f}_2 - \vec{y} \right) \vec{f}_1$$

i.e., $c_1 \vec{f}_1 \cdot \vec{f}_1 + c_2 \vec{f}_1 \cdot \vec{f}_2 = \vec{y} \cdot \vec{f}_1$

OR $\begin{pmatrix} \vec{f}_1 \cdot \vec{f}_1 & \vec{f}_1 \cdot \vec{f}_2 \\ \vec{f}_2 \cdot \vec{f}_1 & \vec{f}_2 \cdot \vec{f}_2 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} \vec{y} \cdot \vec{f}_1 \\ \vec{y} \cdot \vec{f}_2 \end{pmatrix}$

$\frac{\partial E}{\partial c_2} = 0 \Rightarrow c_1 \vec{f}_2 \cdot \vec{f}_1 + c_2 \vec{f}_2 \cdot \vec{f}_2 = \vec{y} \cdot \vec{f}_2$

$\vec{f}_1 = (1.0, 1.0, 1.0, 1.0, 1.0, 1.0)$ $\quad f_1(x) = 1$

$\vec{f}_2 = (1.0, 0.5, 0.25, 0.2, 0.125, 0.1)$ $\quad f_2(x) = \frac{1}{x}$

$\vec{y} = (2.5, 1.53, \cdots 1.1)$

$$\begin{pmatrix} 6.0 & 2.175 \\ 2.175 & 1.398 \end{pmatrix} \begin{pmatrix} c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} 8.280 \\ 3.623 \end{pmatrix}$$

$c_1 = 0.998$

$c_2 = 1.054$

In General:

Given $(x_1, y_1), \cdots, (x_N, y_N)$

Find $c_1, c_2, \cdots, c_M$ such that

$$E = \sum_{1 \le j \le N} \left(f(x_j) - y_j \right)^2 \quad \text{is minimized}$$

$$f(x) = c_1 f_1(x) + c_2 f_2(x) + \cdots + c_M f_M(x).$$

Method:

$$\forall\, k \quad \frac{\partial E}{\partial c_k} = 2 \sum_j \left(\sum_i c_i f_i(x_j) - y_j \right) f_k(x_j) = 0$$

$$\Rightarrow \sum_i c_i \, \mathbf{f}_k \cdot \mathbf{f}_i = \mathbf{f}_k \cdot \mathbf{y}$$

i.e. $$\left(a_{ij} \right) \left(c_j \right) = \left(b_j \right)$$

$$a_{ij} = \mathbf{f}_i \cdot \mathbf{f}_j$$

$$b_j = \mathbf{y} \cdot \mathbf{f}_j$$

Integration

- **Symbolic** computation
  - $P_0, P_1, P_2, \ldots, P_n \rightarrow 0, P_0, \frac{P_1}{2}, \frac{P_2}{3}, \ldots, \frac{P_n}{n+1}$

    $\left(\int P_0 + P_1 x + \cdots + P_n x^n \, dx = P_0 x + \frac{P_1}{2} x^2 + \cdots + \frac{P_n}{n+1} x^{n+1} \right)$
  - Heuristic methods (A.I.) : $\int u dv = u \cdot v - \int v du$
  - **MACSYMA** package (Maple, Mathematica)
    - Can do integration, differentiation, solve equations, O.D.E, P.D.E., factorization,
      ⋮
    - Example : Integrate $[1 + x] \rightarrow x + \frac{1}{2} x^2 + C$

- **Numerical** integration
  - Assume : given x, can compute $f(x)$
  - Goal : Compute $\int_a^b f(x) dx$

    Example $\int_0^{0.5} 1 + x \rightarrow 0.625$

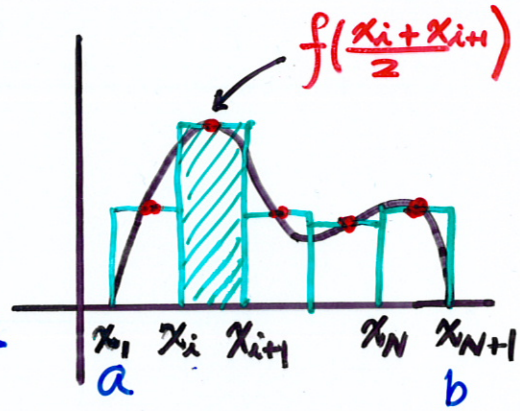
Simple Quadrature

- **Rectangle** method (constant)

$$r = \sum_{1 \le i \le N} (x_{i+1} - x_i) f\left(\frac{x_i + x_{i+1}}{2}\right)$$

$$= \sum_{1 \le i \le N} w \cdot f\left(a + wi - \frac{w}{2}\right)$$

$w = \frac{b-a}{N}$



Error:

$$f(x) = f(c) + f'(c)(x-c) + \frac{f''(c)}{2}(x-c)^2 + \cdots$$

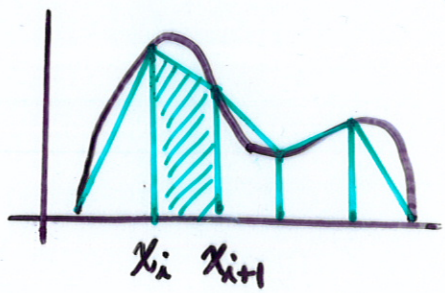
$$\int_a^b f(x)dx = f(c)(b-a) + f'(c)\frac{(x-c)^2}{2}\Big|_a^b + \frac{f''(c)}{2}\frac{(x-c)^3}{3}\Big|_a^b + \cdots$$

$$= r + 0 + w^3 e_3 + w^5 e_5 + \cdots$$

$w^3 e_3 = f''(c) \pm ?$

- **Trapezoid** method (linear)

$$t = \sum_{1 \le i \le N} (x_{i+1} - x_i) \frac{f(x_i) + f(x_{i+1})}{2}$$



Error:

$$\int_a^b f(x)dx = t - 2w^3 e_3 - 4w^5 e_5 + \cdots$$

- **Simpson's** method (quadratic)

$$S = \frac{2}{3}r + \frac{1}{3}t$$

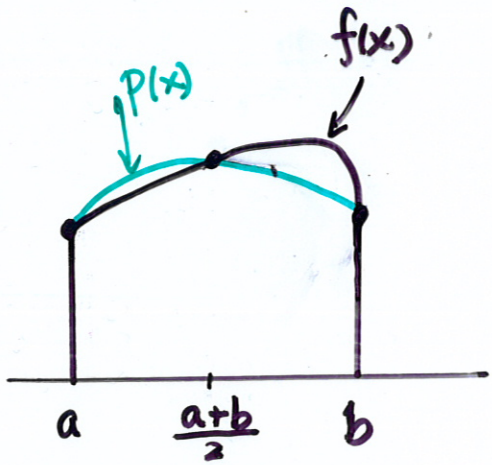
$$= \sum_i (x_{i+1} - x_i)\frac{1}{3}\left[2f\left(\frac{x_i + x_{i+1}}{2}\right) + \frac{f(x_i) + f(x_{i+1})}{2}\right]$$

$$= \sum_i \frac{x_{i+1} - x_i}{6}\left[f(x_i) + 4f\left(\frac{x_i + x_{i+1}}{2}\right) + f(x_{i+1})\right]$$

Error: $\int_a^b f(x)dx = \frac{1}{3}\left(2r + t - 2w^5 e_5 + \cdots\right)$

- **Spline** method (cubic)

- Why Simpson's method Quadratic ?



Exercise :

Given $\begin{cases} f(x) \\ p(x) = \alpha + \beta x + \gamma x^2 \end{cases}$

such that : $f(a) = p(a)$

$f(b) = p(b)$

$f(\frac{a+b}{2}) = p(\frac{a+b}{2})$

Prove that

$$\int_a^b p(x)dx = \frac{b-a}{6}\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right]$$

- Real Statistics

$$\int_1^2 \frac{1}{x}dx = \ln 2 = 0.693147180599$$

	N=10	N=100	N=1000
Rectangle	0.6928353604100 (-3)	0.693144055 6283	0.693147149 3
Trapezoid	0.693771403 1754 (+6)	0.693153430 4818	0.693147243 05
Simpson's	0.693147374665 |	0.693147180579 5	0.693147180559 9

Adaptive method

- Method 1

```
Adapt(a, b)
  if |Rect(a, b, 10) - Trap(a, b, 10)| < tolerance then
      adapt := Rect(a, b, 10);
  else
      adapt := adapt(a, (a+b)/2) + adapt((a+b)/2, b);
```

- Method 2

```
Adapt(a, b)
  if |Simp(a, b, 10) - Simp(a, b, 5)| < tolerance then
      adapt := Simp(a, b, 10);
  else
      adapt := adapt(a, (a+b)/2) + adapt((a+b)/2, b);
```