String Matching

• Brute-force

$i-j+2$ $\qquad$ i

Text $a[N]$: a b c x y z a b c d a b c d a b c x y z

Pattern $P[M]$: a b c d a b x y

j

BruteSearch

```
i := 1 ; j := 1;
repeat
   if a[i] = P[j] then
         i := i+1;  j := j+1;
   else
         i := i-j+2;  j := 1;
until  j > M  or  i > N
```

$j > M \Downarrow$ $i-M$ (or sentinel)

$i > N \Downarrow$ Not Found

Too bad !

• 0 0 0 0 0 0 0 0 0 0 0 0 1 $\qquad$ $8*5+5$

0 0 0 0 1 $\qquad$ $\Rightarrow$ KMP

• 1 0 0 0 0 1 0 0 0 0 0 0

1 0 0 0 0 0 $\qquad$ $\Rightarrow$ Boyer-Moore

$T = MN$

Knuth-Morris-Pratt algorithm

i never backup

a b c x y z a b c d a b c d a b x y z

a b c d a b x y

Next[j]

j

KMPsearch

```
i:=1; j:=1;
repeat
    if j=0 or a[i]=P[j] then
        i:= i+1;  j:= j+1;
    else
        j := next[j];
until j>M or i>N;
```

$j>M \Rightarrow i-M$

$i>N \Rightarrow$ Not Found

a b c x y z
x y z — next[1]=0

a b c x y z
a x y — next[2]=1

a b c x y z
a b x y — next[3]=1

a a c x y z
a a x y — next[3]=2

- How to compute next[j] ?

x x x a b c d a b e x y z a b c d x x x x x x x

a b c d a b e x y z a b c d a b c m n p

Pattern

j

j=next[i]

next[i]

i i i i

next[i]: 0 1 1 1 1 2 3 1

next[i]=j
P[i] = P[j] $\Rightarrow$ next[i+1] = j+1

- **Init Next**

```
i:=1;  j:=0;  next[1]:=0;
repeat
    if j=0 or P[i]=P[j] then
        i:= i+1;  j:= j+1;  next[i]:= j;
    else
        j:= next[j]
until i>M
```

T = M+N

Hard-Wire "next" table into program

$a[N+1..N+M] := P[1..M]$

10100111

j	next[j]
1	0
2	1
3	1
4	2
5	3
6	1
7	2
8	2

```
    i:=0;
0:  i:=i+1;
1:  if a[i]<>'1' then goto 0;  i:=i+1;   ← Comp a[i] with P[1]=1
2:  if a[i]<>'0' then goto 1;  i:=i+1;   ←        "        P[2]=0
3:  if a[i]<>'1' then goto 1;  i:=i+1;   ←        "        P[3]=1
4:  if a[i]<>'0' then goto 2;  i:=i+1;
5:  if a[i]<>'0' then goto 3;  i:=i+1;   ←                 P[5]=0
6:  if a[i]<>'1' then goto 1;  i:=i+1;
7:  if a[i]<>'1' then goto 2;  i:=i+1;
8:  if a[i]<>'1' then goto 2;  i:=i+1;
    search:=i-8;
```

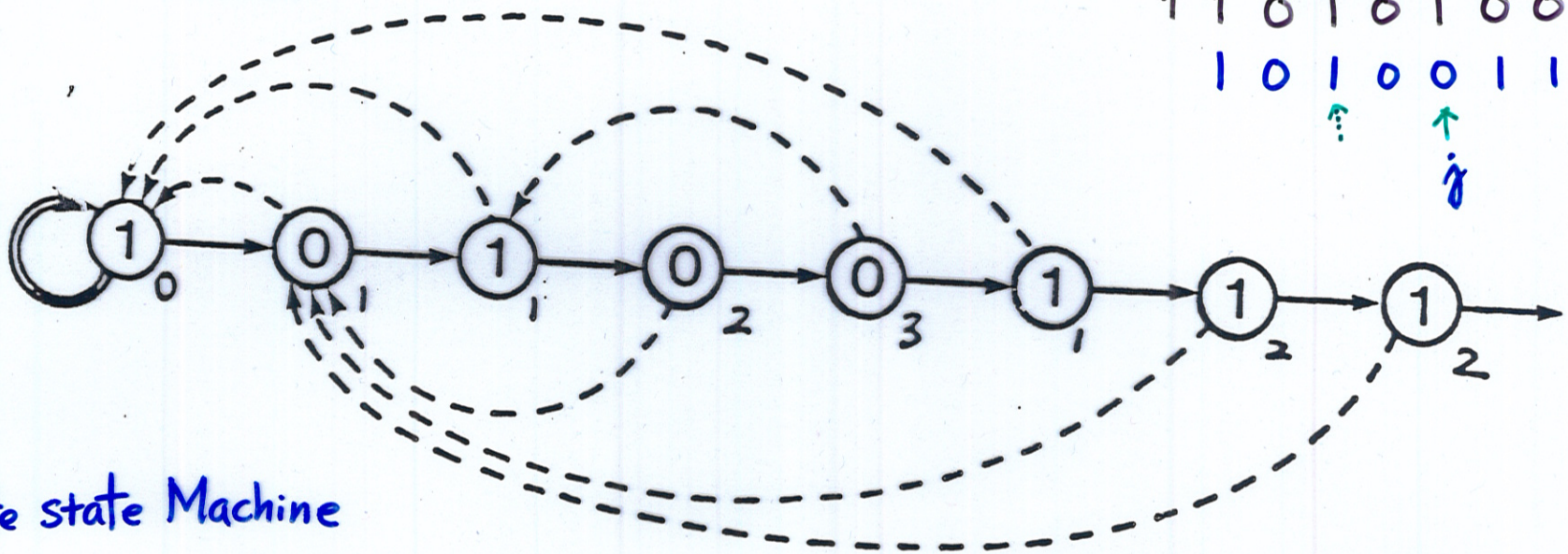
j = state = program counter

i

1 1 0 1 0 1 0 0 1 1 1 1 0-

1 0 1 0 0 1 1 1

j



Finite state Machine

Boyer-Moore algorithm

$T = M + N$

```
Text    : WHICH-FINALLY-HLATS-AT-THAT
Pattern : AT-THAT
          AT-THAT
            AT-THAT
              AT-THAT
                AT-THAT
```

$d_1[F] = 7$ $d_1[-] = 4$

$d_2[5] = 7$ $d_1[-] = 4$

BMsearch

```
i := M; j := M;
repeat
    if a[i] = p[j] then
        i := i-1; j := j-1;
    else if j = M then
        i := i + d1[a[i]];
    else
        i := i + d2[j]; j := M;
Until j<1 or i > N
```

$j<1 \Rightarrow i+1$

$i>N \Rightarrow$ Not Found

How to compute d_1, d_2 ?

i	1	2	3	4	5	6	7	8
$P[i]$	A	S	S	E	S	S	E	S
$d_2[i]$	15	14	13	7	11	10	3	

```
??? % SSES ????
ASSESSES
   ASSESSES
```

$\therefore d_2[4] = 7$

$d_1[S] = 0$

$d_1[E] = 1$

$d_1[A] = 7$

others $= 8$

$d_1[c] = M - \max_j (P[j] = c)$

```
???????E???
ASSESSES
```

Rabin-Karp algorithm

- encode string into integer.

a b d e a b c text

a b c Pattern

$d = 26$

$$[abc]_{26} = 1 \cdot 26^2 + 2 \cdot 26 + 3$$

$$[abd]_{26} = 1 \cdot 26^2 + \underline{2 \cdot 26 + 4}$$

$$[bde]_{26} = \underline{2 \cdot 26^2 + 4 \cdot 26} + 5$$

$$= \left([abd]_{26} - 1 \cdot 26^2\right) * 26 + 5$$

text : $a_1 a_2 \cdots \underline{a_i a_{i+1} \cdots a_{i+M-1}} \; a_{i+M} \cdots a_N$

Pattern : $\underline{p_1 p_2 \cdots p_M}$

$$x = [a_i \cdots a_{i+M-1}]$$

$$\Rightarrow \quad [a_{i+1} \cdots a_{i+M}] = \left(x - a_i d^{M-1}\right) * d + a_{i+M} \pmod q$$

$T = N + M$

File Compression

- reduce space (encoding) ⇒ reduce computing or communication time
- Impossible to compress every files.

• Run length

AAAABBBAABBBBBCDDDQM

- alphabets only ⇒ 4A3BAA5BC3DQM
- Mixed ⇒ QDABBBAAQEBCDDDQ~~A~~M

  Escape character (Q, 4)

  Q = Q

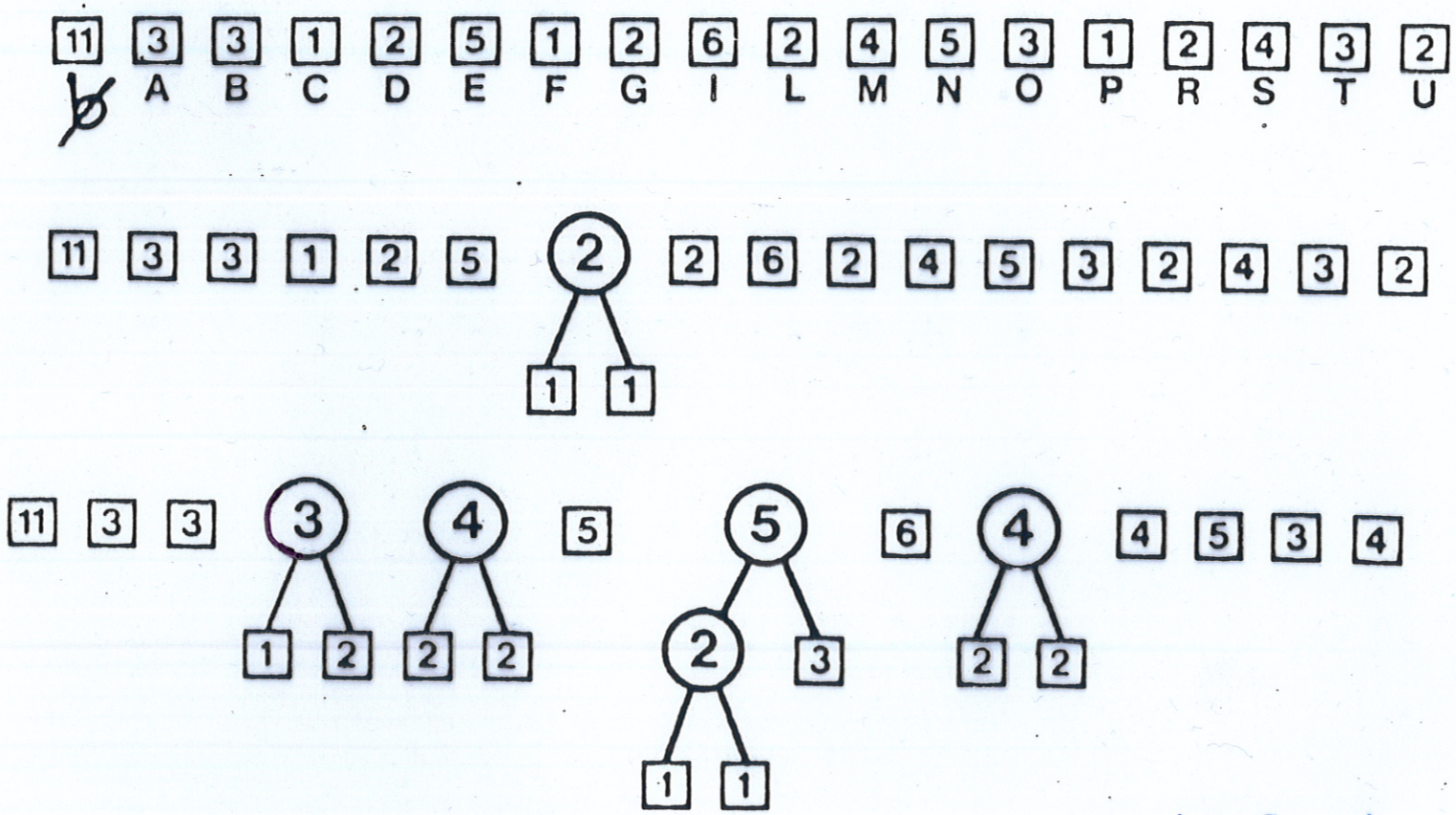
- Binary file : '0', '1'

51 bits ⇒ No EOL

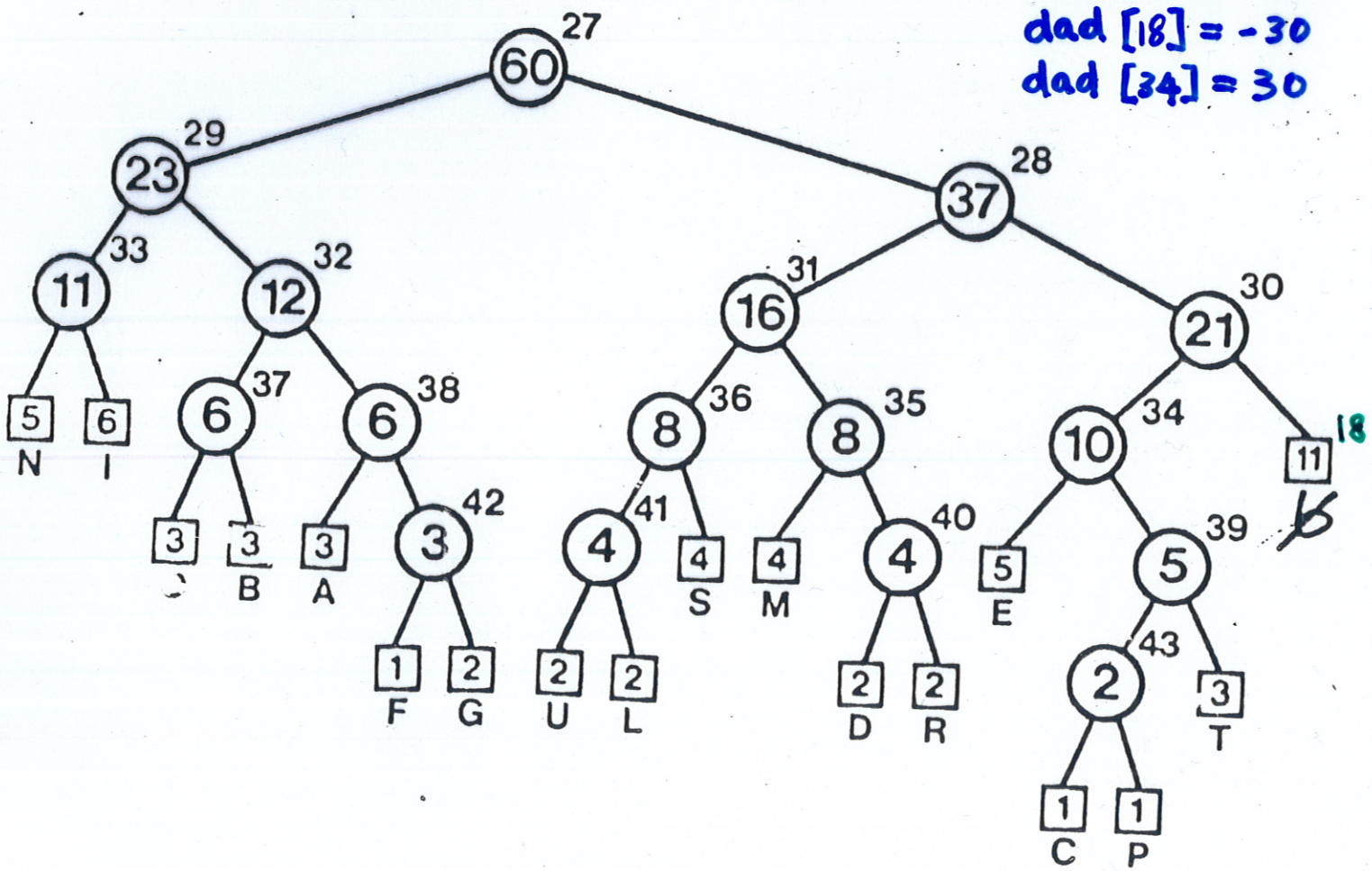
00000000000000000000000000000000111111111111111000000000	28 14 9
00000000000000000000000000000000111111111111111110000000	26 18 7
00000000000000000000000000000111111111111111111111110000	23 24 4
00000000000000000000000000001111111111111111111111111000	22 26 3
00000000000000000000000000011111111111111111111111111110	20 30 1
00000000000000000000000011111100000000000000000001111111	19 7 18 7
00000000000000000000000011111000000000000000000000011111	19 5 22 5
00000000000000000000000011100000000000000000000000000111	19 3 26 3
00000000000000000000000011100000000000000000000000000111	19 3 26 3
00000000000000000000000011100000000000000000000000000111	19 3 26 3
00000000000000000000000011100000000000000000000000000111	19 3 26 3
00000000000000000000000011110000000000000000000000001110	20 4 23 3 1
00000000000000000000000011100000000000000000000000111000	22 3 20 3 3
01111111111111111111111111111111111111111111111111111111	1 50
01111111111111111111111111111111111111111111111111111111	1 5
01111111111111111111111111111111111111111111111111111111	1 5
01111111111111111111111111111111111111111111111111111111	1 50
01111111111111111111111111111111111111111111111111111111	1 50
01100000000000000000000000000000000000000000000000000011	1 2 46 2

• Huffman Coding

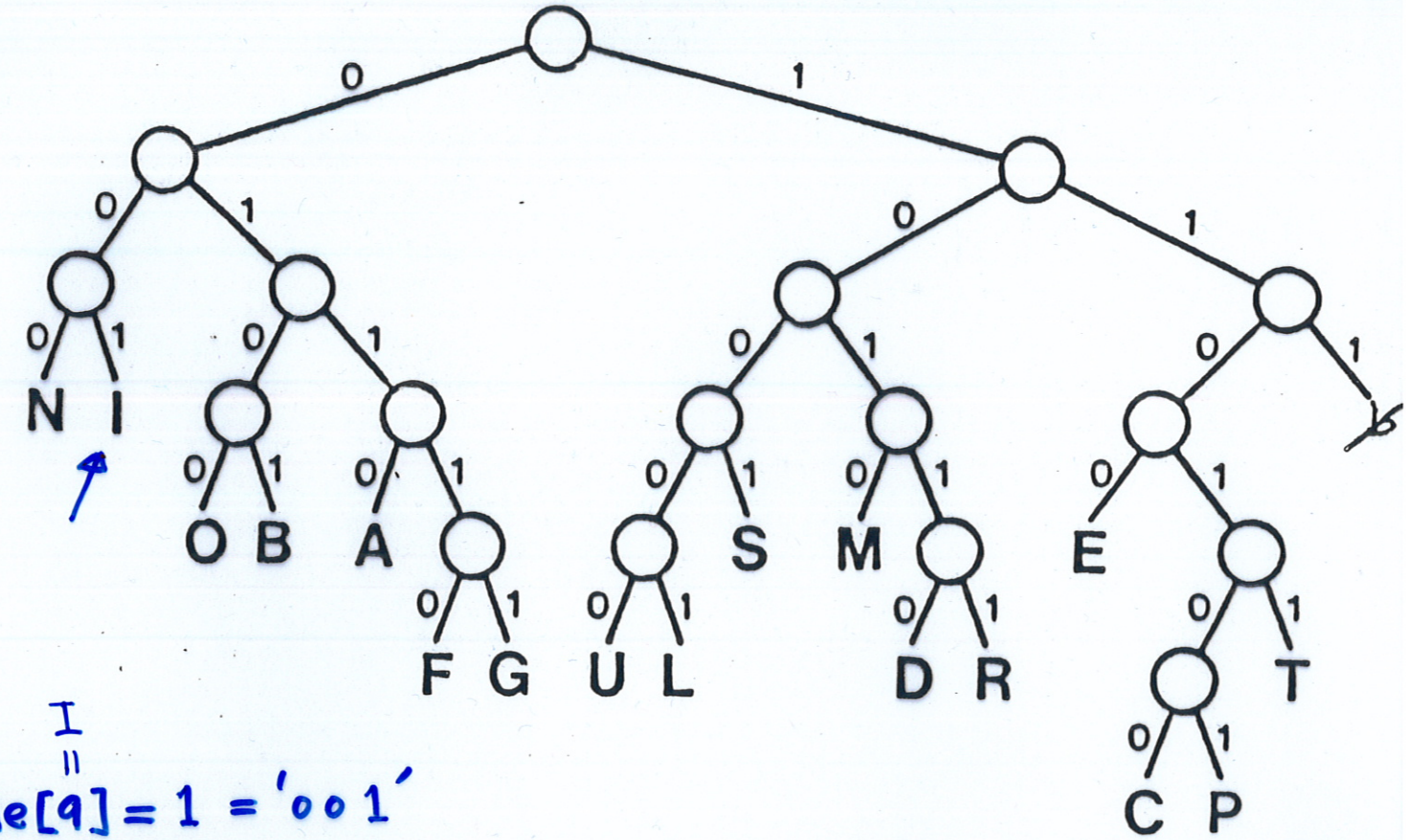
- encode frequent character with shorter string



count[30] = 21
dad[18] = -30
dad[34] = 30



- With priority queue



I
||
Code[9] = 1 = '001'
len[9] = 3

⇔ external nodes of a binary tree

- **Prefix code**: no sequence is a prefix of another
- **Optimum prefix code**: shortest encoded string

Example of non-prefix code

A: 10
B: 011
C: 01110

01110 = BA ?
C ?

- Dynamic Huffman Coding

Cryptology (密碼學)

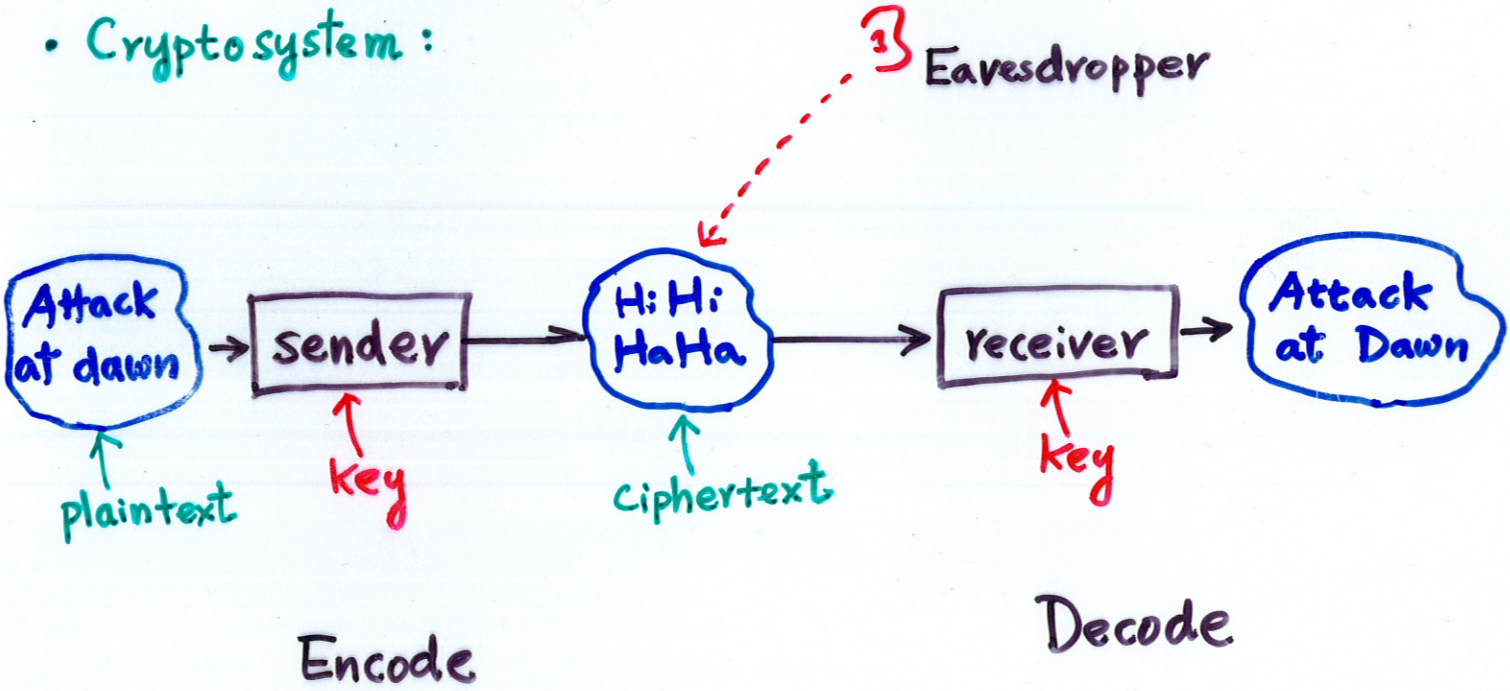
- **Cryptograph** : encode, encryption, (編碼), encipher

  Cryptoanalysis : decode, decryption, (解碼), decipher

- **Applications** : military, file system,

  electronic fund transfer

- **Cryptosystem** :



Simple methods

- **Caesar Cipher** ($K=3$)

$$F: N\text{th} \rightarrow (N+K)\text{th}$$

Plaintext: ATTACK AT DAWN $\quad K=1$

Ciphertext: BUUBDLABUAEBXO

- Even F: 1-1 : $\{A, \ldots, Z\} \rightarrow \{A, \ldots, Z\}$ is easy to break by frequency counting.

- **Vigenere cipher**

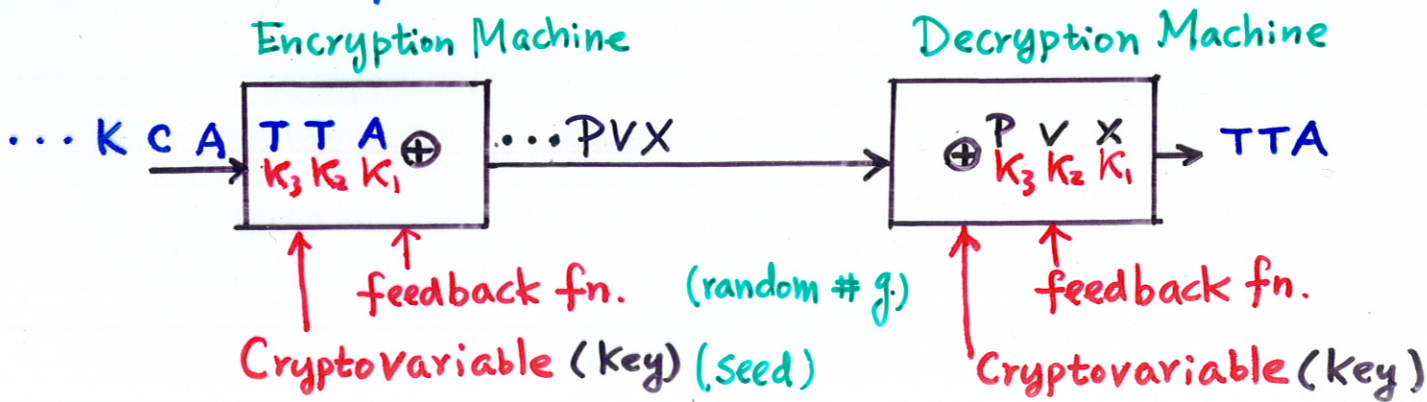
Key	:	A	B	C	A	B	C	A	B	C
Plaintext	:	A	T	T	A	C	K		A	T
Ciphertext	:	B	V	W	B	E	N	A	C	W

- **Vernam cipher (one time pad)**
  - $\forall A \in \text{Key} \hookrightarrow F_A$, 1-1, onto : $\{A, \ldots, Z\} \rightarrow \{A, \ldots, Z\}$
  - Key as long as plaintext
  - The Only Secure system

- **Binary file**

$$\text{Ciphertext} = \text{Key} \oplus \text{plaintext}$$

- **Encryption/Decryption Machine (Hardware)**



Number Theory

Theorem (Fermat): $(a, p) = 1$, p: prime $\Rightarrow a^{p-1} \equiv 1 \mod p$

Pf: (i) $\{1 \cdot a \bmod p,\ 2a \bmod p,\ \ldots,\ (p-1)a \bmod p\}$

$= \{1, 2, \ldots, p-1\}$

Example: $\{3 \bmod 5,\ 6 \bmod 5,\ 9 \bmod 5,\ 12 \bmod 5\}$

$= \{3, 1, 4, 2\}$

$\because\ xa \equiv ya \mod p \Leftrightarrow p \mid (x-y)a$

$\Leftrightarrow p \mid x-y$

$\Leftrightarrow x \equiv y \mod p$

(ii) $(p-1)!\, a^{p-1} \equiv (p-1)! \mod p$

$\Rightarrow a^{p-1} \equiv 1 \mod p$

Example: $3^{5-1} \equiv 1 \mod 5$

$4^{7-1} \equiv 1 \mod 7 \qquad (\because 4^3 = 64 \equiv 1 \mod 7)$

Theorem $(M, p) = (M, q) = 1 \Rightarrow M^{(p-1)(q-1)} \equiv 1 \mod \underset{pq}{\underset{\shortparallel}{N}}$

Pf: $M^{p-1} \equiv 1 \mod p$, $M^{q-1} \equiv 1 \mod q$ $\Rightarrow$ $M^{(p-1)(q-1)} \equiv 1 \mod p$, $M^{(p-1)(q-1)} \equiv 1 \mod q$

$\Rightarrow M^{(p-1)(q-1)} \equiv 1 \mod pq$

Theorem (Euler) $(a, N) = 1 \Rightarrow a^{\varphi(N)} \equiv 1 \pmod N$

$\varphi(N) :=$ # relative primes in $\{0, 1, 2, \cdots, N-1\}$

- **Public-Key systems**

  E : public key

  D : private key

  (1) $D(E(M)) = M$, $\forall$ message M. $\Rightarrow E: 1\text{-}1$.

  (2) All (D, E) distinct

  (3) $E \underset{M, E(M)}{\not\Rightarrow} D$ (Even, $M, E(M), \not\Rightarrow D$)

  (4) E, D are easy to compute.

- Proposed by Diffe, Helleman, 1976.
- Rivest-Shamir-Adleman (RSA) algorithm, 1978.

p, q prime

By primality test

$N = p \cdot q$

$N \not\Rightarrow p, q$ Factorization is Hard

$\varphi(N) = (p-1)(q-1)$

$(d, \varphi(N)) = 1 \Rightarrow ed + K\varphi(N) = 1$ (d: prime)

$E = (N, e)$ public Key

$D = (N, d)$ private Key

$$M \rightarrow \boxed{\text{sender}} \xrightarrow{M^e \bmod N} \boxed{\text{receiver}} \rightarrow (M^e)^d = M \bmod N$$

$E = (N, e)$ $\qquad$ $D = (N, d)$

$\because$ $\boxed{M^e}^d = M^{-K\varphi(N)+1}$

cipher

$= M^{-K(p-1)(q-1)+1}$

$\equiv M \bmod N$

Example: $p=3,\ q=7$

$N = 21$

$\varphi(N) = 12$

$d=5,\quad e=5 \qquad \because 5\cdot 5 - 2\cdot 12 = 1$

$M=3 \qquad M^e \bmod N = 3^5 \bmod 21 = 12$ ← Cipher

$12^d \bmod N = 12^5 \bmod 21 = 3$

$12 = 2\cdot 5 + \underline{2}$

$5 = 2\cdot \underline{2} + 1$

$\Rightarrow 1 = 5 - 2\cdot(12 - 2\cdot 5)$

$= 5\cdot 5 - 2\cdot 12$

Why RSA algorithm works? $(N, e, M^e \overset{?}{\Rightarrow} M) \bmod N$

$\overset{?}{\Rightarrow} d$

$\underset{e}{N} \overset{?}{\Longrightarrow} p, q\ ? \Rightarrow \varphi(N), d \Rightarrow M$

$O(e^{\sqrt{\ln n \ln\ln n}})$

$\because$ The only way to factorize N is to test primes $2, 3, 5, 7, \cdots, \sqrt{N}$.

N # digits	# operations Factorize N	Computing time with Cray
50	1.4×10^{10}	4 Hours
80	1.3×10^{13}	150 Days
100	2.3×10^{17}	74 Years
200	1.2×10^{23}	3.8×10^{9} Years

• Digital Signature

(an application of public key systems)

$$(E_i, D_i) \xrightarrow{M'} (E_j, D_j)$$

$$E_i\left(D_j\left(\underline{E_j\left(D_i(M)\right)}\right)\right) = M$$

M' = message i sends to j

Require: (1) $D(E(M)) = \underline{E(D(M))} = M$.

(2) i can't deny, j can't forge $D_i(M)$!!

• Knapsack problem

- Given $\{a_1, a_2, \dots, a_n\}$, K, $\exists\ a_{i_1} + a_{i_2} + \dots + a_{i_p} = K$?

$$\Leftrightarrow\ a_1 x_1 + a_2 x_2 + \dots + a_n x_n = K,\ x_i = 0, 1\ ?$$

- General $\{a_n\}$: NP-complete, No polynomial time algorithm.
- Super-increasing $\{a_1, a_2, \dots, a_n\}$: $\forall j = 2, \dots, n$, $a_1 + \dots + a_{j-1} < a_j$

  can be solved in $O(N)$ time.

Example: $(3, 5, 9, 20, 44)$

$3 < 5$

$3+5 < 9$

$3+5+9 < 20$

$3+5+9+20 < 44$

$$3x_1 + 5x_2 + 9x_3 + 20x_4 + 44x_5 = 69 \mod 89$$

$x_5 = 1$

$= 25$ $\quad x_4 = 1$

$= 5$ $\quad x_3 = 0,\ x_2 = 1$

$x_1 = 0$

$4 * 67 \equiv 1 \mod 89$

$$\Rightarrow *4 \uparrow \downarrow *67 \quad \begin{aligned} &3x_1 + 5\underline{x_2} + 9x_3 + 20\underline{x_4} + 44\underline{x_5} \equiv 69 \mod 89 \\ &23x_1 + 68\underline{x_2} + 69x_3 + 5\underline{x_4} + 11\underline{x_5} \equiv 84 \mod 89 \end{aligned}$$

• Knapsack cipher system (Hellman & Merkle, 1978)

public key : $(23, 68, 69, 5, 11)$, 89

private key : $(3, 5, 9, 20, 44)$, 4

plaintext : $M = (01011)_2$

ciphertext : $M' = 84$

Receiver
(i) $4 * 84 \equiv 69 \mod 89$
(ii) Solve x_i, obtain $(01011)_2$

• But Shamir (1983) showed, this method is breakable!

Computational Geometry

- **Objects:** point : (x, y)
  line, segment, array : (p, q)
  polygon : $p_0, p_1, \ldots, p_N, p_0$
  planar subdivision

- **Questions:** intersection — static / dynamic
  point-location
  decomposition

- **Applications:**
  - graphics,
  - CAD
  - VLSI
  - Image Processing
  - Pattern Recognition
  - Robotics

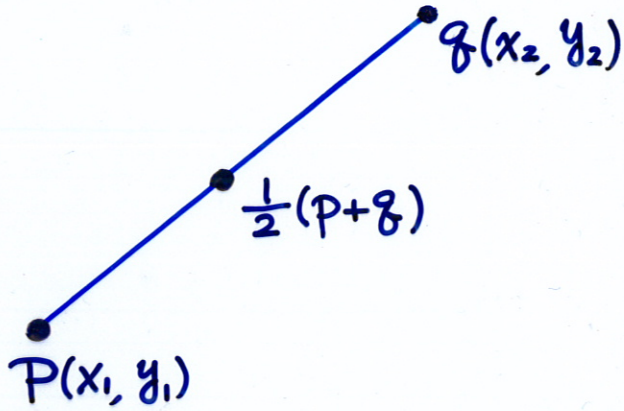
- **Paradigms:** Incremental Construction
  Plane sweep
  Locus
  Divide & Conquer
  Geometric Transformation
  Prune & Search
  Dynamization

Scan-conversion

- Geometric objects → Pixels, Bitmaps
  (line, area)

- **Draw(p, q)**

```
dot(p);
dot(q);
if |x1-x2|>1 or |y1-y2|>1 then
    draw(p, 1/2(p+q));
    draw(1/2(p+q), q);
```



- **Incremental algorithm**

$$m = \frac{y_2 - y_1}{x_2 - x_1}; \qquad b = y_1 - m x_1$$

$$y - y_1 = m(x - x_1)$$

$$y = mx + y_1 - mx_1$$
$$= mx + b$$

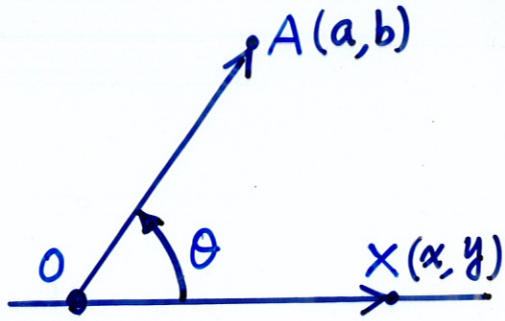
$x \to x+1$
$y \to y+m$

Program:

```
y := y1;
for x := x1 to x2 do
    dot(x, round(y));
    y := y+m;
```

Line Intersection

$$r'e^{i\theta} = \frac{a+ib}{x+iy} = \frac{(x-iy)(a+ib)}{x^2+y^2}$$

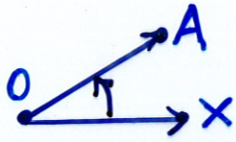
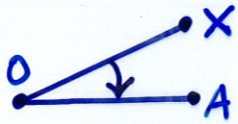
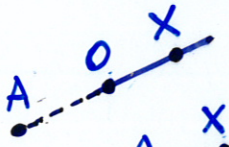
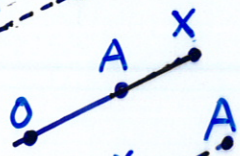
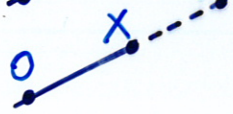


$\therefore\ r\sin\theta = bx - ay > 0$

$\Leftrightarrow\ 0 < \theta < 180^\circ$

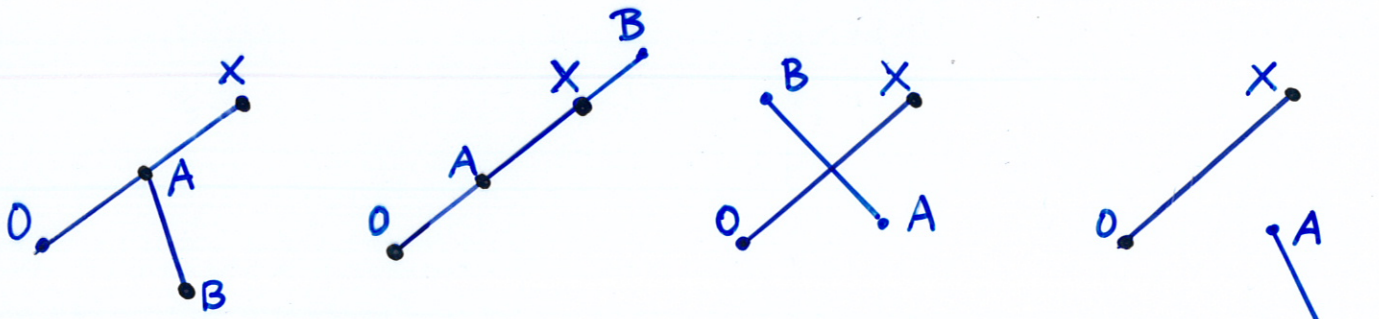
- Counter ClockWise

$CCW(O, X, A) =$

$+1$		$bx - ay > 0$
-1		$bx - ay < 0$
-1		$bx - ay = 0$ { $ax<0$ or $by<0$
0		$x^2+y^2 \geq a^2+b^2$
$+1$		else }

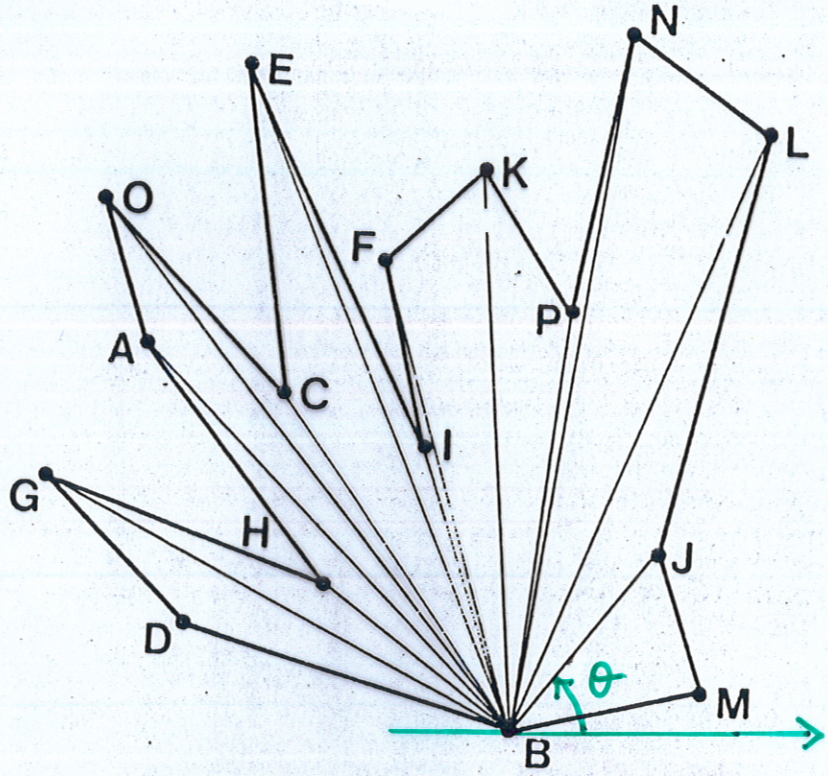
- $L_1 = (O, X)$
  $L_2 = (A, B)$ Segments

$L_1 \cap L_2 \neq \emptyset \Leftrightarrow CCW(O,X,A) * CCW(O,X,B) \leq 0$ AND
$CCW(A,B,O) * CCW(A,B,X) \leq 0$



$CCW(O,X,A) = 0$ $\qquad$ $CCW(O,X,A) = -1$

Simple closed path : thru every points, does not intersect itself



Method: order points according to θ to an anchor point.

- How to find $\theta = \tan^{-1}\frac{y}{x}$?

$\theta = \tan^{-1}\frac{y}{x}$ ↑ in (I) 象限.

$\Leftrightarrow \frac{y}{x}$ ↑ 〃

$\Leftrightarrow 0 \le \frac{y}{x+y} \le 1$ ↑ 〃

$\frac{b}{a} < \frac{d}{c} \Leftrightarrow \frac{b}{a+b} < \frac{d}{c+d}$

i.e., $\boxed{-1 \le \frac{y}{|x|+|y|} \le 1 \quad \begin{matrix} \nearrow & \text{I, IV} \\ \searrow & \text{II, III} \end{matrix}}$

Program

$t := \frac{y}{|x|+|y|}$; (0, $x=y=0$)

if $x < 0$ then (II, III)

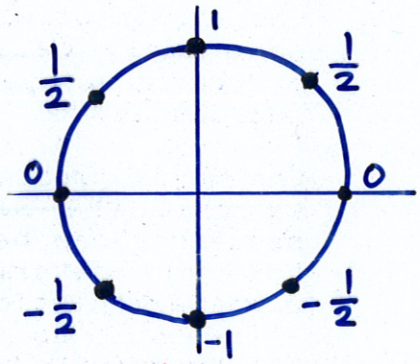
$t := 2-t$;

else if $y < 0$ then (IV)

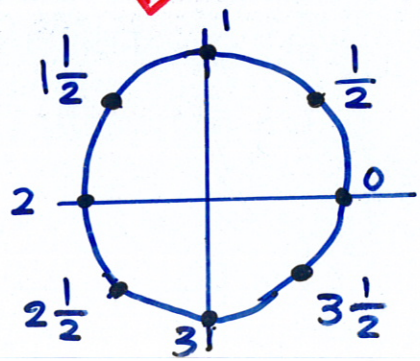
$t := 4+t$;

$0 \le t \le 4$

$0 \le 90t \le 360$

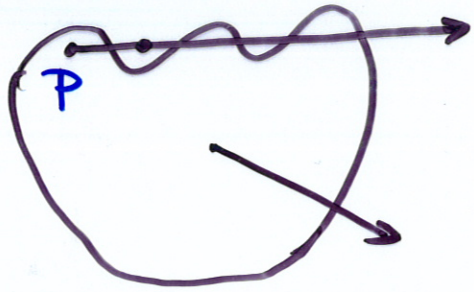


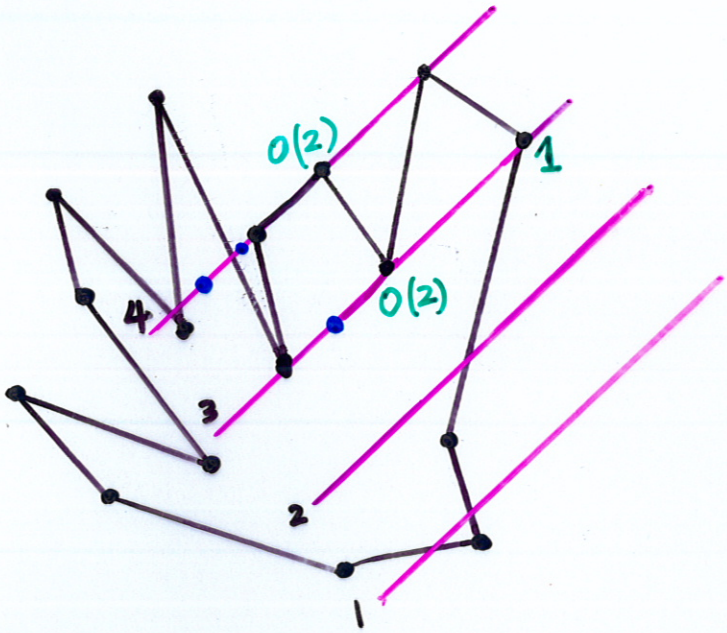
⇓



Inclusion in Polygon $P_0, P_1, \dots, P_N = P_0$

P in Polygon ?



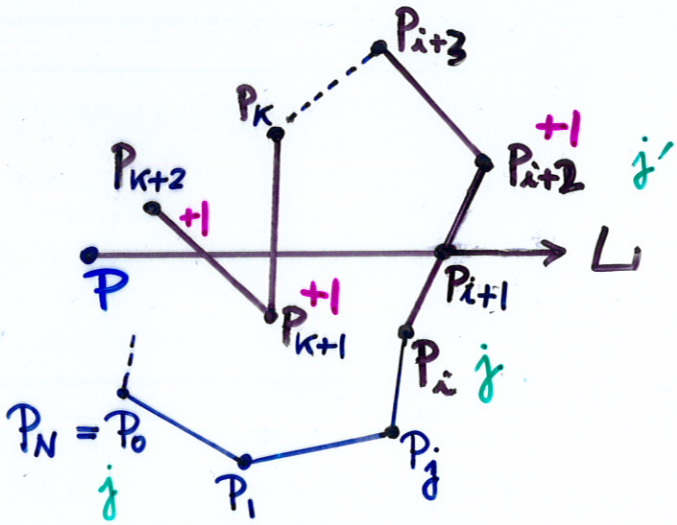


in $\Leftrightarrow$ odd intersection
out $\Leftrightarrow$ even

- Go from one side to another $\Rightarrow$ **count +1**

Program

```
j := 0;  count := 0;
for i := 1 to N do
    if P_i ∉ L then {
        if P_j P_i ∩ L ≠ ∅ then
            count := count + 1;
        j := i;
    }
```



- Choose P_1: min x among min y coordinates

$$P_1 \in L \Rightarrow \underset{\underset{P_N}{\parallel}}{P_0} \notin L$$

Bug: $P_0 \in L$, $P_1 \notin L$.

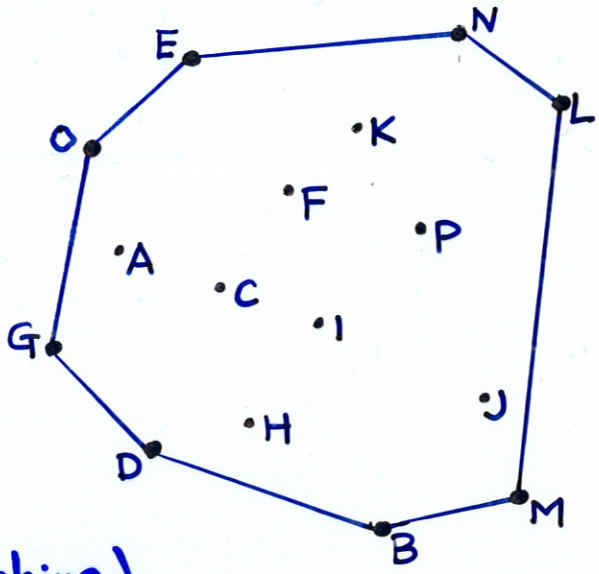
Convex Hull

Smallest convex polygon (curve) contains all points

Goal: find vertices define the Hull.

Properties of CH:

- all points on same side of edge in CH.
- points with $\begin{Bmatrix} \max \\ \min \end{Bmatrix} \begin{Bmatrix} x \\ y \end{Bmatrix}$ are vertices of CH.



Package Wrapping (Jarvis Marching)

p_0 : smallest y

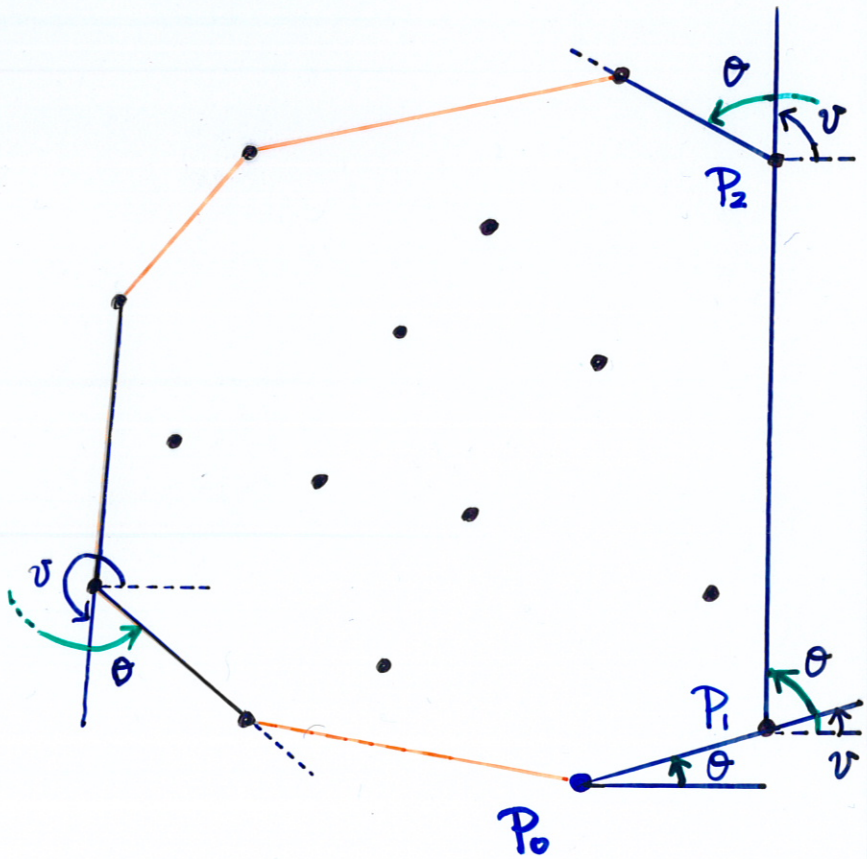
p_1 : min θ

p_2 : min θ, among $> v$

⋮

until $p_n = p_0$

$T = NM$

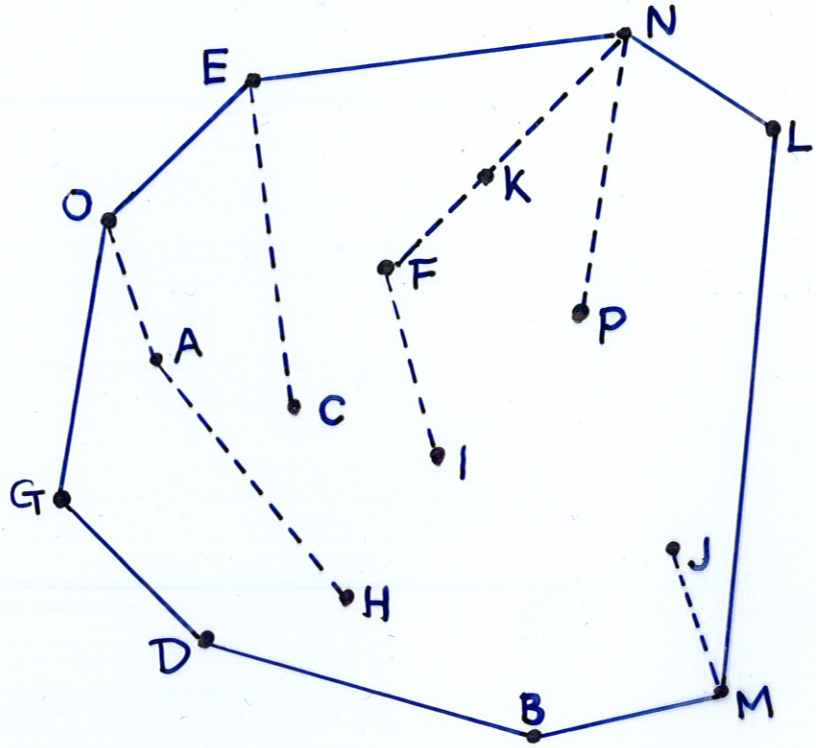


Graham Scan

- Choose 1 point, sorting according to θ
- Add 1 pt, modify to keep CH property : i.e., all points on same side of Hull edge. (No right-turn)

Example:

- Add K
- B, K different sides of $\overleftrightarrow{PN}$
- B, K same side of $\overleftrightarrow{NL}$
  $CCW(N, L, K) < 0$



$T = N \log N + O(N)$

BMJLNPKFIECOAHGD

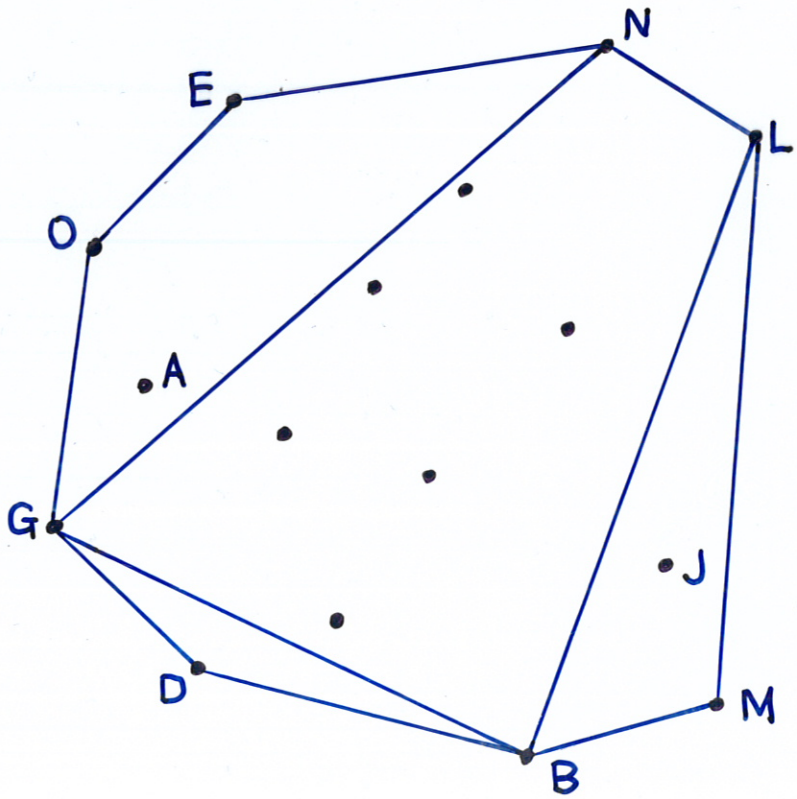
Interior Elimination

- Pick 4 points on the Hull, discard every point inside, then
- Use Graham scan.

4 points can be

(1) max, min, x
max, min, y

(2) max, min, $x+y$
max, min, $x-y$



Complexity

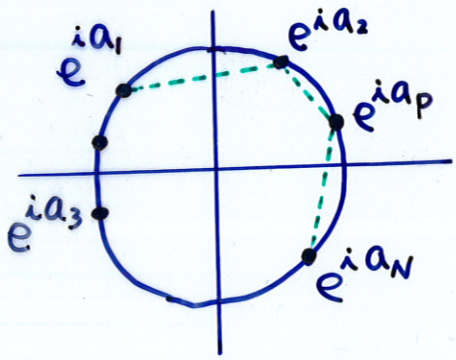
- Package Wrapping : NM
- Graham Scan : $N \log N$ (optimum)
- Interior Elimination : $N \log N$ (worst)
  N (in practice)

- Complexity of Convex Hull = Complexity of sorting = $N \log N$

proof (i) Sorting $\propto$ computing convex Hull

To sort $\{a_i\}$: (1) $S = \{e^{ia_i}\}$ or $\{(a_i, f(a_i))\}$, $f(x) = x^2$ (or e^x) ← convex

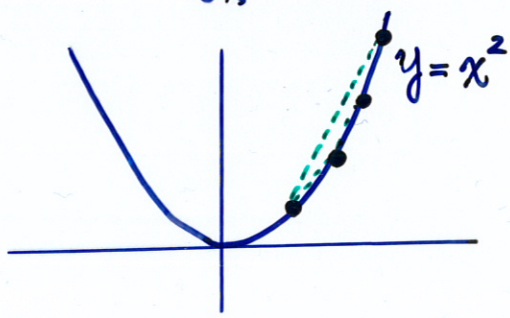
(2) $CH(S)$

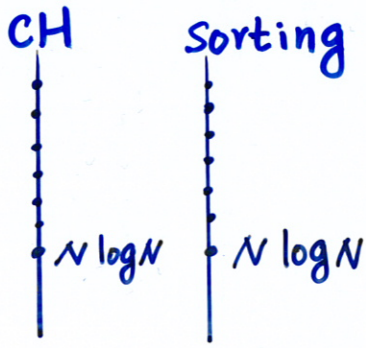


$\therefore$ Convex Hull $\geq N \log N$

(ii) $\exists$ $N \log N$ CH algorithm

OR





Range Searching (Multi-key searching)

- Find Record $= (x_1, \ldots, x_N)$, s.t., $x_1 \in [a_1, b_1]$
  $x_2 \in [a_2, b_2]$
  $\vdots$
  $x_N \in [a_N, b_N]$

  x_1: attribute

$\boxed{N = 1}$

- $a_1 = b_1$: internal searching
- Preprocessing
  Range-search
- **Method 1**

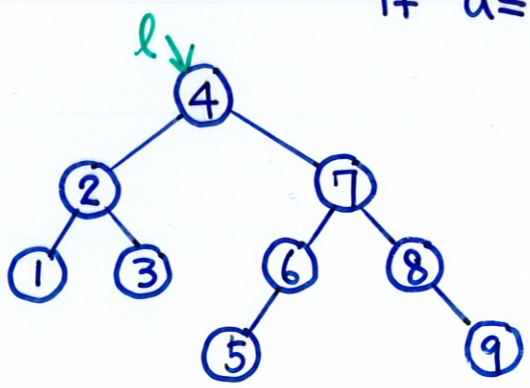
  Preprocessing : sorting $\quad N \log N$
  Range-search : binary search $\quad R + \log N$

  Keys: 1, 2, 3, 4, 5, 6, 7, 8, 9
  $Q = [2.5, 7]$

- **Method 2**

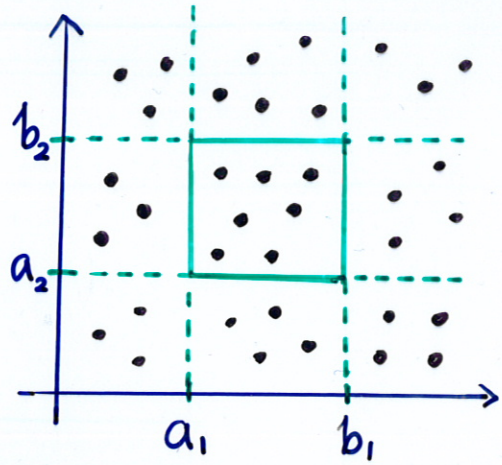
  Preprocessing : binary search tree $\quad N \log N$
  Range-search : Search(ℓ, [a, b]) $\quad R + \log N$

```
if a ≤ ℓ.key then search(ℓ.left, [a, b]);
if ℓ.key ≤ b then search(ℓ.right, [a, b]);
if a ≤ ℓ.key ≤ b then print(ℓ.key);
```



$Q = [2.5, 7]$

$\boxed{N=2}$ $Q=[a_1, b_1]\times[a_2, b_2]$



- **Sequential search**

  $\forall\ 1\le i\le N$, check $(x_i, y_i)\in Q$

- **Projection**

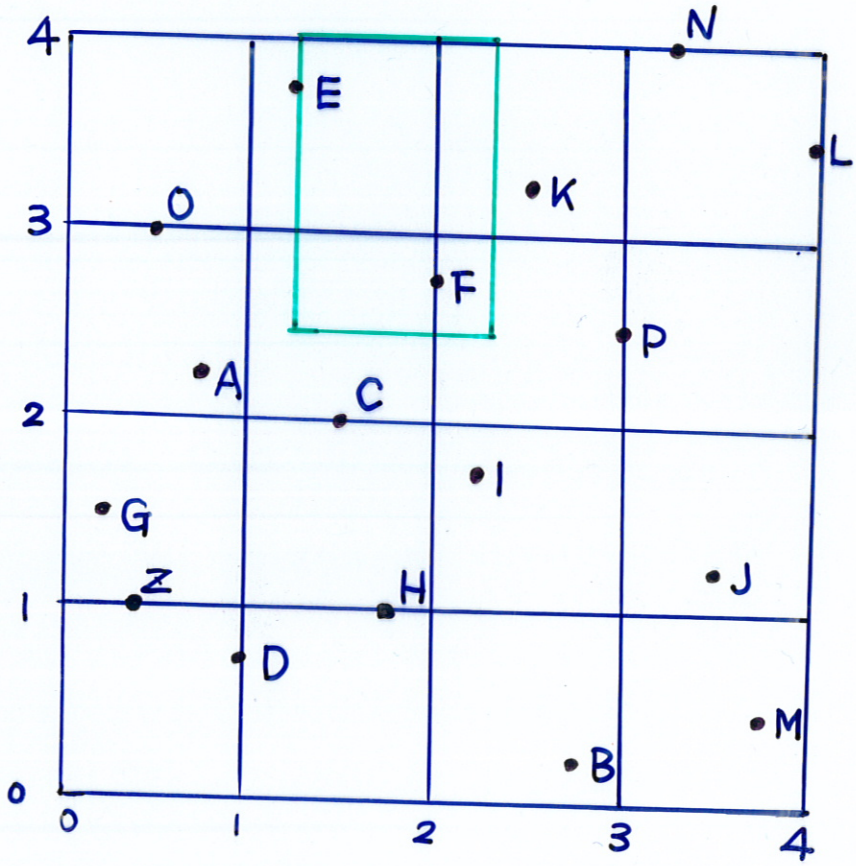
  (1) $\forall i$, check $x_i\in[a_1, b_1]$?

  (2) check $y_i\in[a_2, b_2]$? among those i pass (1)

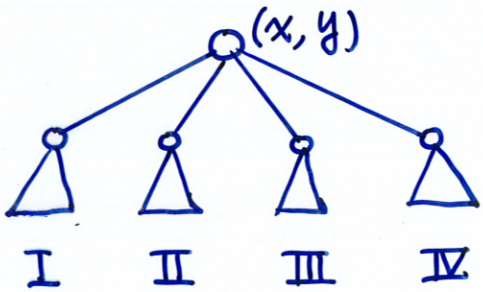
- **Grid method** (Atlas)

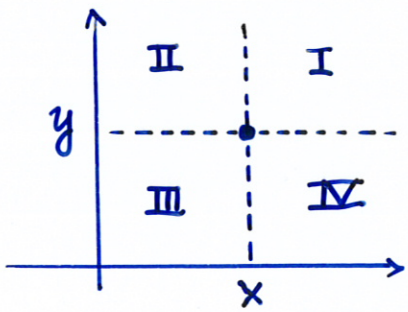
  Grid[0..4, 0..4]

  Grid[0,1] → G, Z

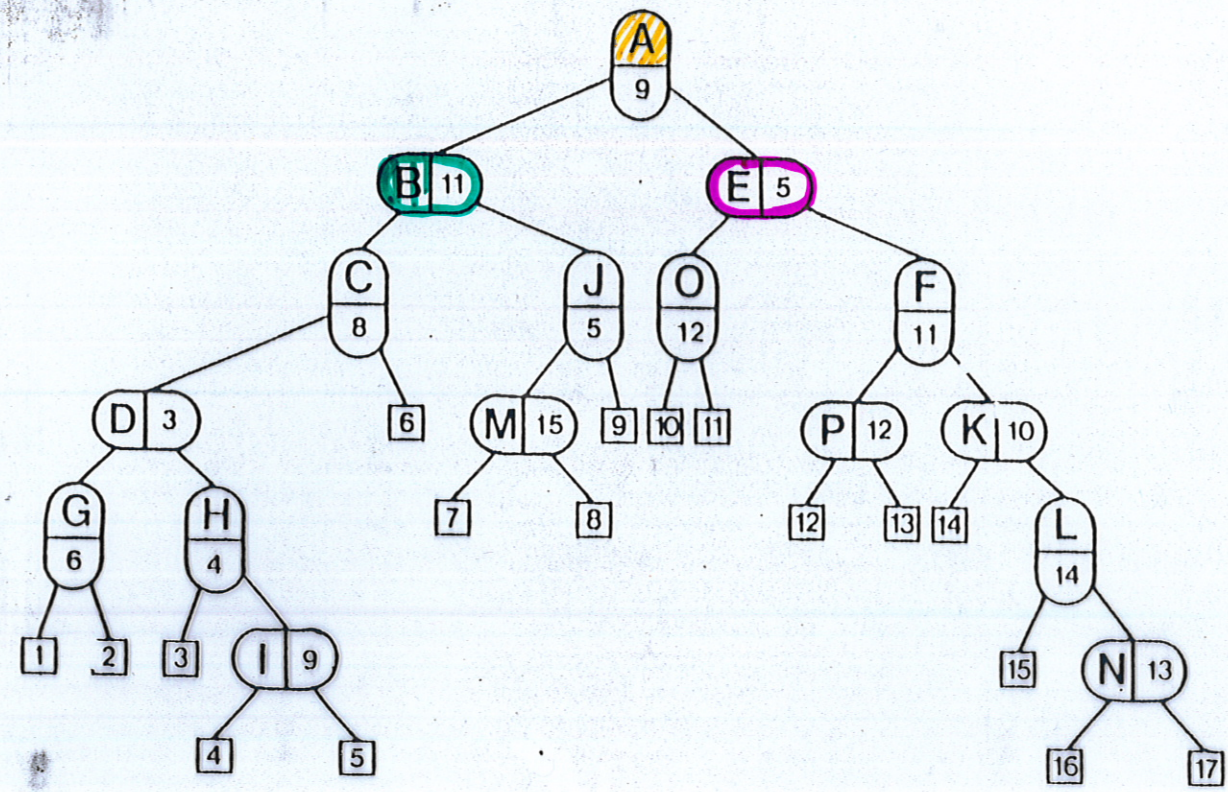


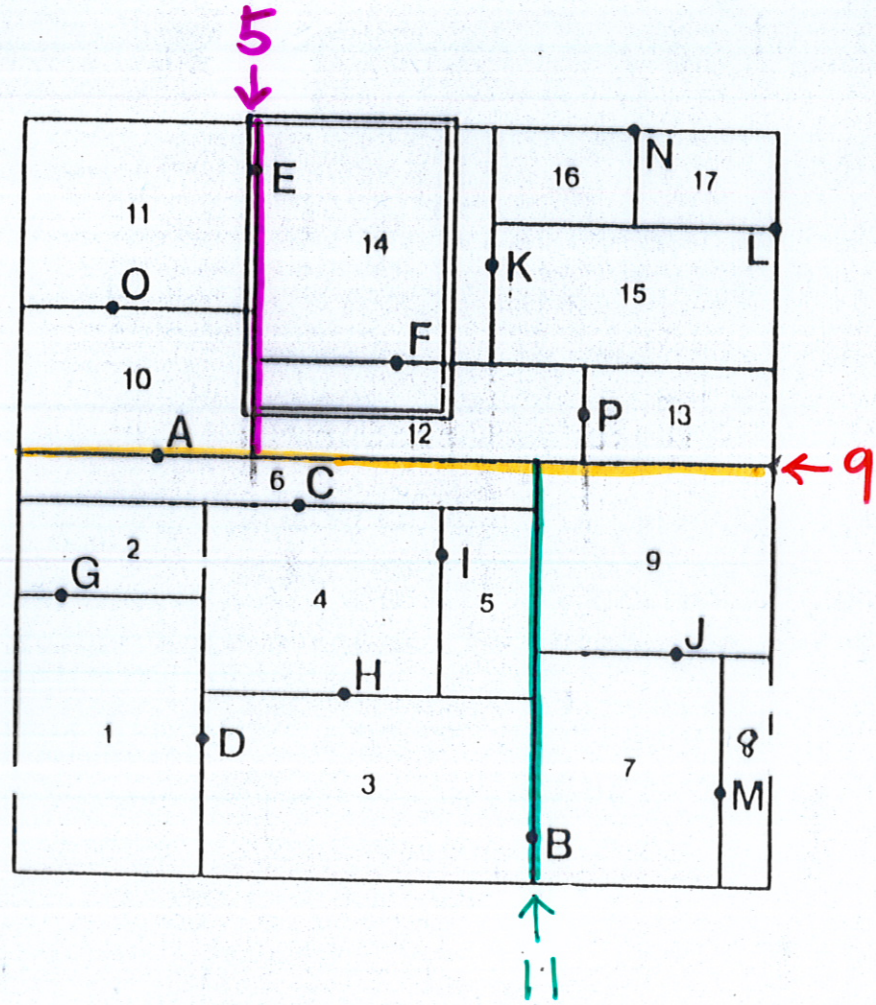
- **K-D tree**
- **Quad-tree**



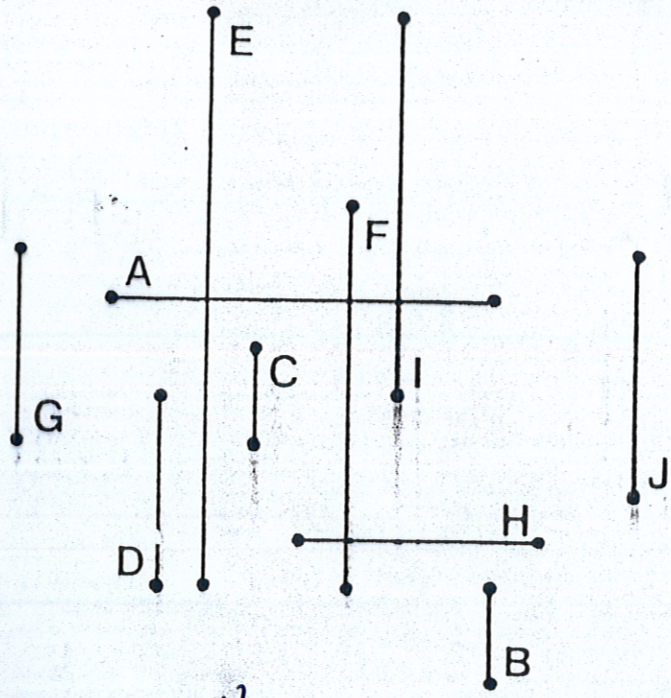


2-D tree


A 9
B 11
E 5
C 8
J 5
O 12
F 11
D 3
6
M 15
9
10
11
P 12
K 10
G 6
H 4
7
8
12
13
14
L 14
1
2
3
I 9
15
N 13
4
5
16
17


5
E
11
14
16
N
17
L
K
15
O
F
10
P
13
A
12
9
6
C
2
G
I
9
4
5
J
H
1
D
3
7
8
M
B
11

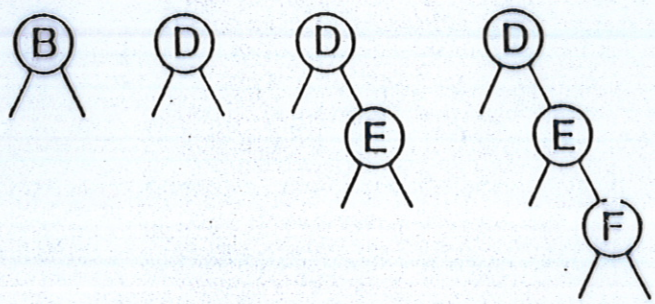
Geometric Intersection : Sweeping = Binary tree + range searching



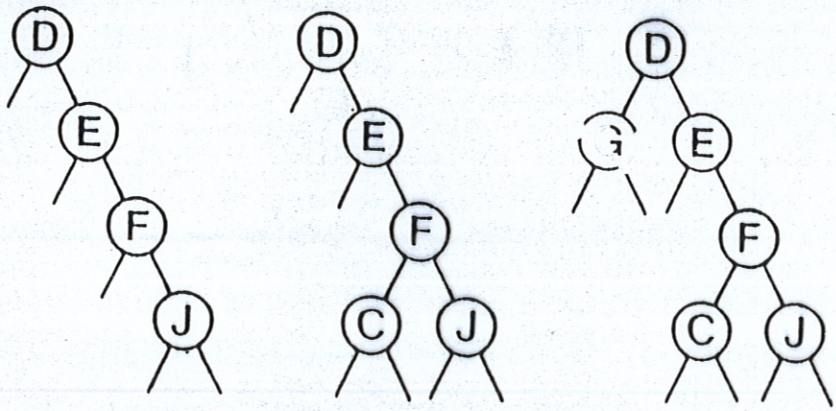
- Binary tree $\supset \{ I \cap \text{——} \neq \phi \}$

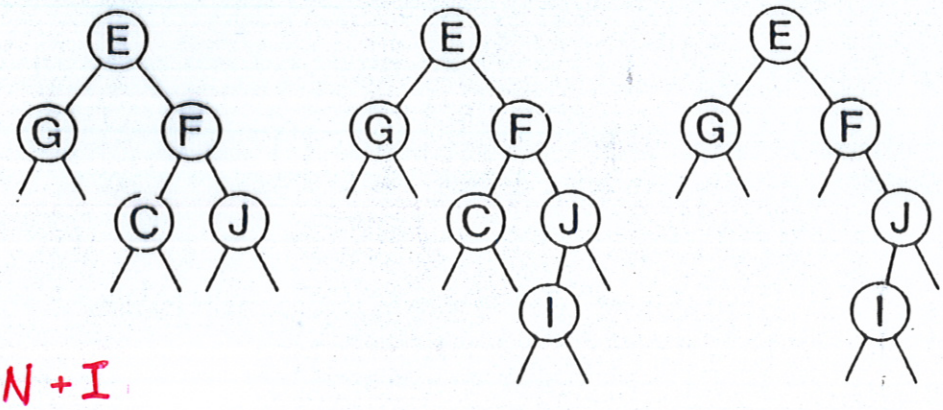
  Key is x coord.

B B̈ D E F H̄ J C G
Ḋ I Ċ Ā Ġ Ḟ J̇ Ė Ī



range search H



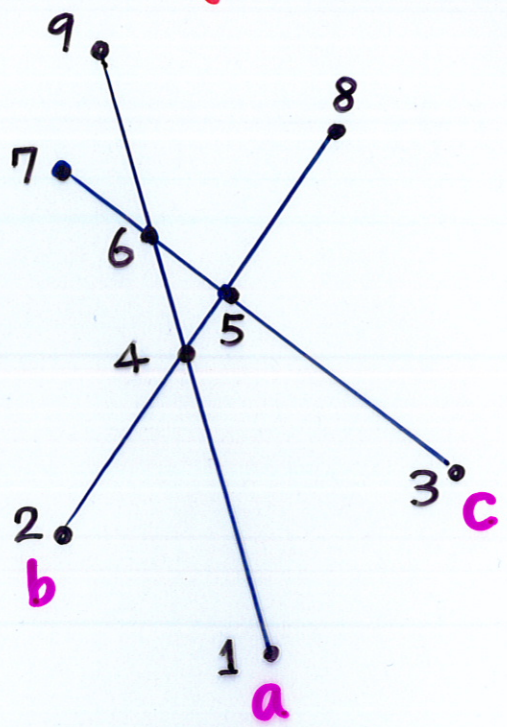


$T = N \log N + I$

search A

- ⇒ Insert
- ⇒ Delete
- ⇒ Range Search

General Segment Intersection



x-structure	y-structure (priority queue)
a, b	8, 9
c, a, b	7, 8, 9
a, c, b	6, 7, 8, 9
a, b, c	5, 6, 7, 8, 9
b, a, c	4, 6, 7, 8, 9
b, a	3, 4, 7, 8, 9
a	2, 3, 7, 8, 9
	1, 2, 3, 7, 8, 9

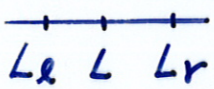
Property: Intersecting lines must be adjacent in x-structure some time

Algorithm

While y-structure $\neq \phi$ do {

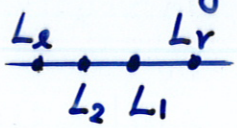
delete P from y-structure;

$P = \text{Low}(L) \Rightarrow$ (1) Insert(L, x);

(2) 

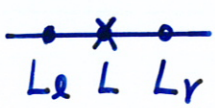
Insert$(L \cap L_l, y)$
Insert$(L \cap L_r, y)$ if exist

$P = L_1 \cap L_2 \Rightarrow$ (1) Interchange L_1, L_2 in x-structure

(2) 

Insert$(L_l \cap L_2, y)$ if above
Insert$(L_1 \cap L_r, y)$ sweep L

$P = \text{High}(L) \Rightarrow$ (1) Delete(L, x);

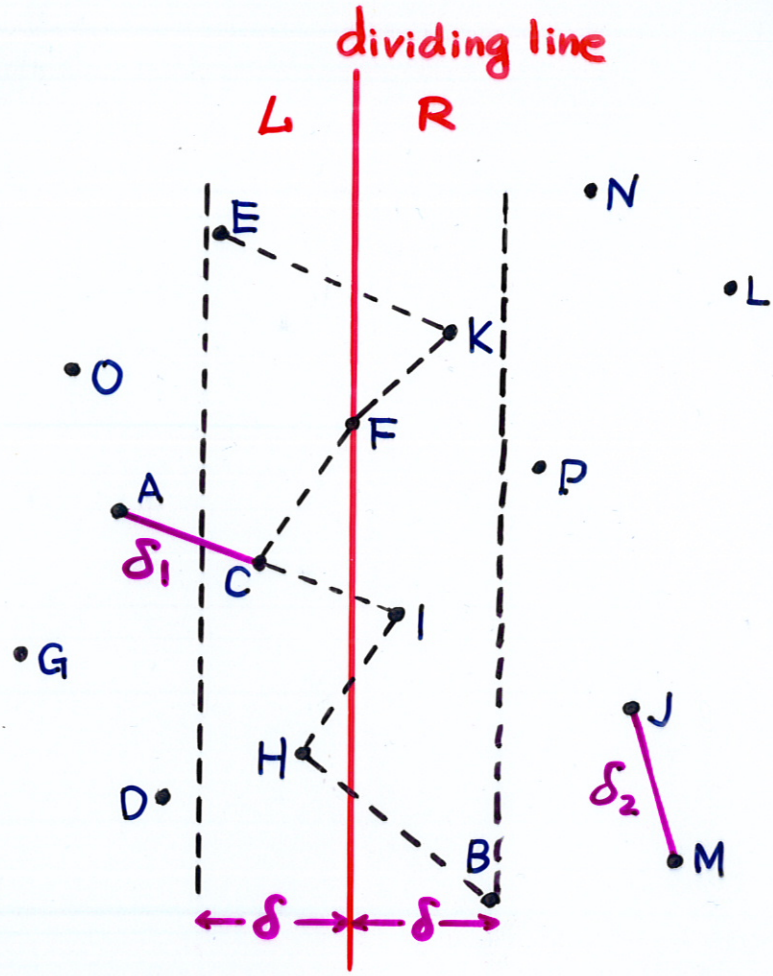


Insert$(L_l \cap L_r, y)$ "

}

$T = N \log N + I \log N$

Closest Points



Algorithm (divide & conquer)

(1) Find closest pairs $\begin{matrix}\delta_1\\ \delta_2\end{matrix}$ of $\begin{matrix}L\\ R\end{matrix}$ recursively

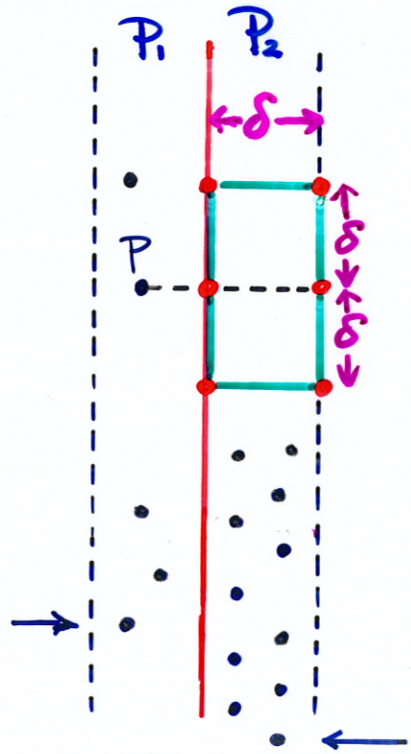
(2) $\delta = \min(\delta_1, \delta_2)$

(3) Project $\begin{matrix}P_1\\ P_2\end{matrix} \rightarrow \begin{matrix}P_1^*\\ P_2^*\end{matrix}$ sorted by y-coord.

(4) $\forall p \in P_1^*$, scan ≤ 6 pts in P_2^* (within δ)

Get δ_ℓ

(5) $\delta_s = \min(\delta, \delta_\ell)$



Presort y coordinate

$$\Rightarrow T(N) = 2T\left(\frac{N}{2}\right) + O(N)$$

$$= O(N \log N)$$

Voronoi Diagram

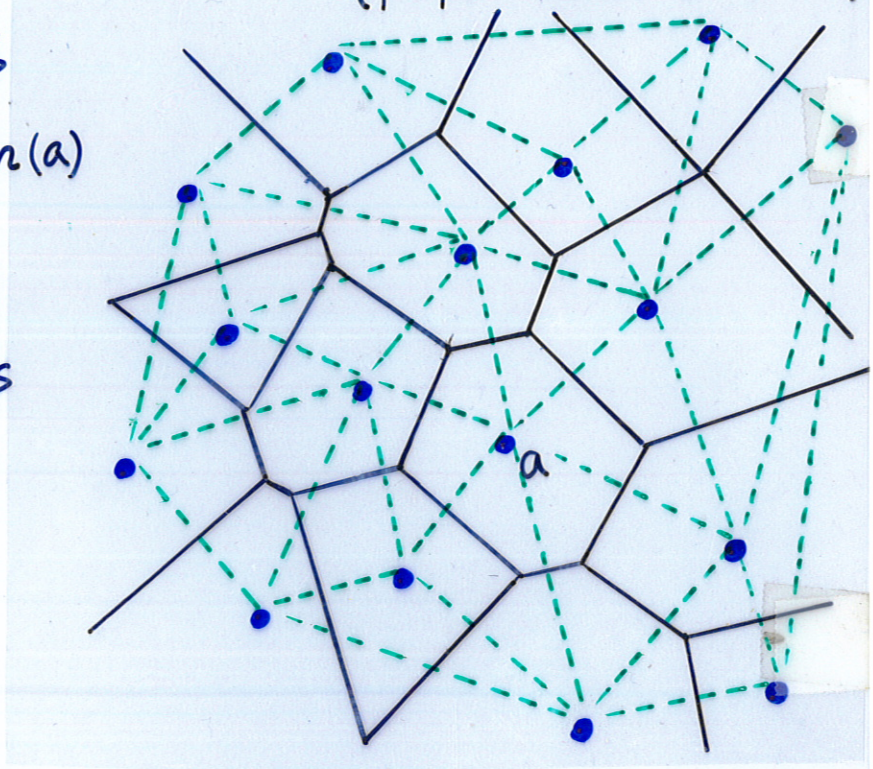
Given $a_1, \cdots, a_N$

- Voronoi polygon (a) := all points closer to a than other a_i's. (perpendicular bisector)

  verteices $\leq 2N-4$

  edges $\leq 3N-6$

- Voronoi diagram := $\bigcup_a$ polygon (a)



- Voronoi Dual
  (Delaunay triangulation)
  triangulation of all points
- degree (vertex) = 3 (外心)
- Convex Hull $\subset$ Dual
- Polygon (a) unbounded
  $\Leftrightarrow$ a $\in$ convex Hull

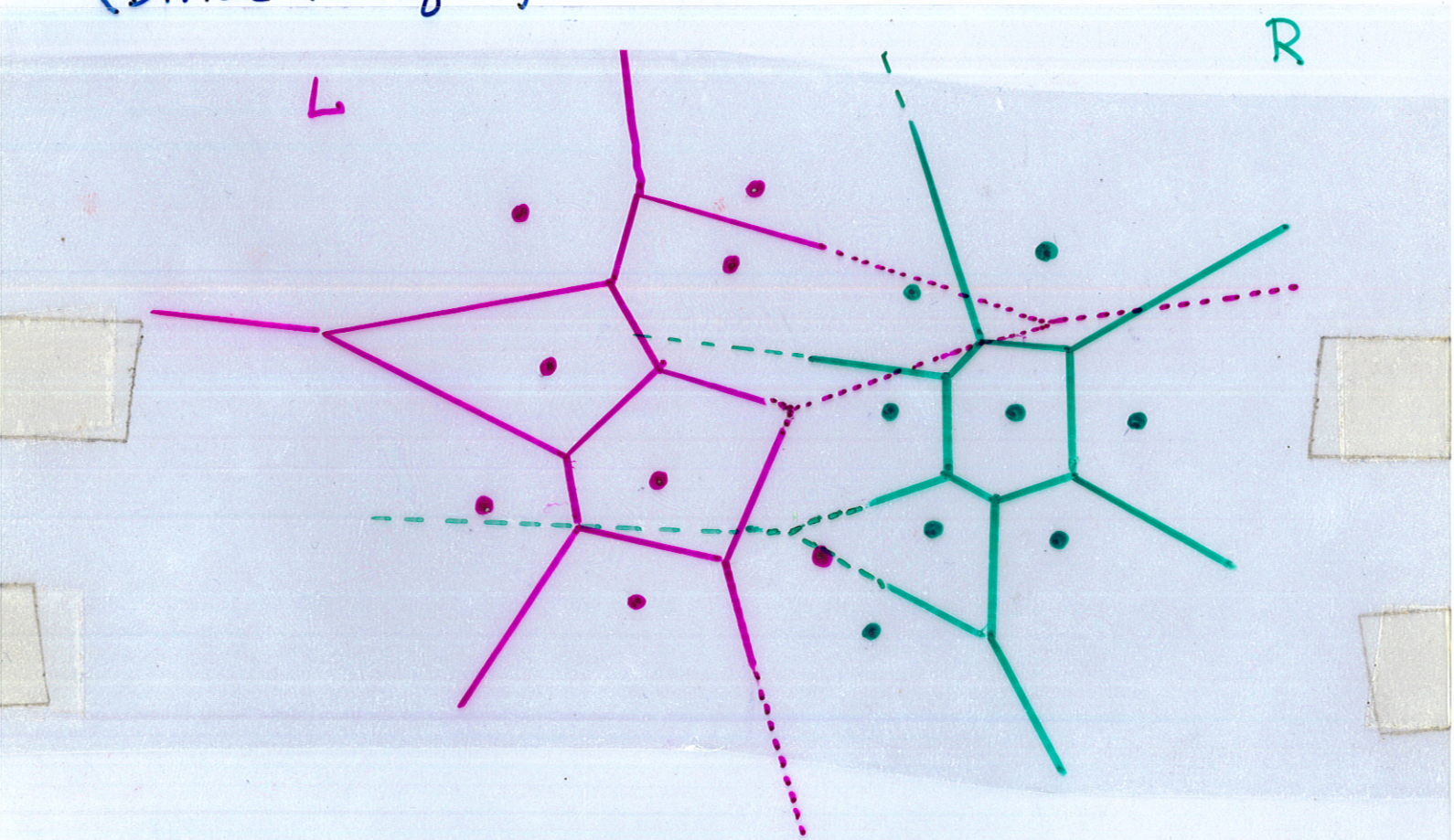
- **Applications**
  - closest pair : find closest pair of a_i's ($\subset$ Dual)
  - all nearest neighbor : find nearest neighbor of a_i ($\subset$ Dual)
  - nearest neighbor : find nearest a_i of a point P ($P \in$ poly(a_i))
  - computing convex Hull : ($\subset$ Dual)
  - Euclidean Minimum Spanning Tree : ($\subset$ Dual)
    find a tree connects all a_i's with minimum distance.

- Can compute Voronoi Diagram in $O(N \log N)$

  $\Rightarrow$ All can be solved in $O(N \log N)$

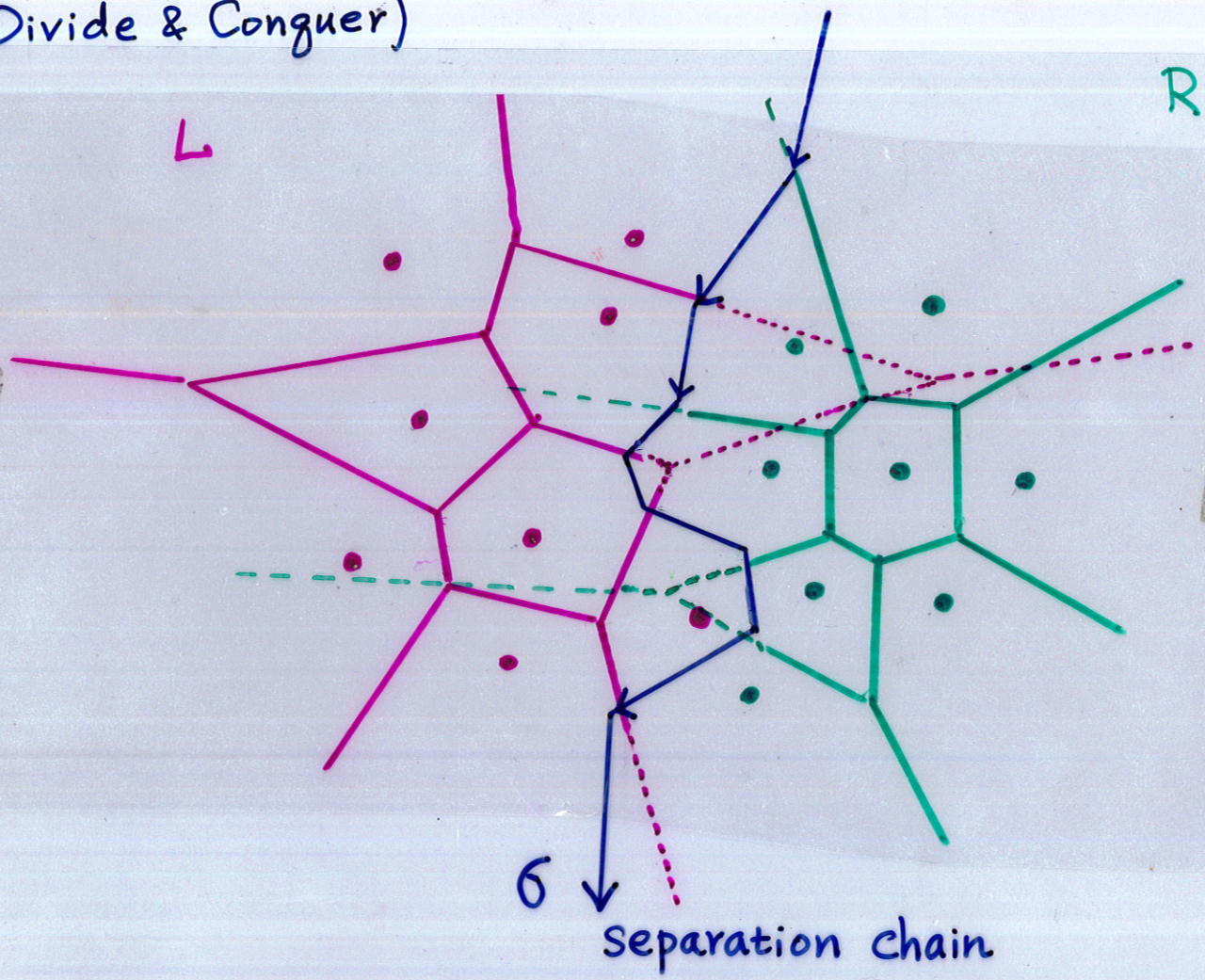
Compute Voronoi Diagram in $O(N \log N)$ time

(Divide & Conquer)

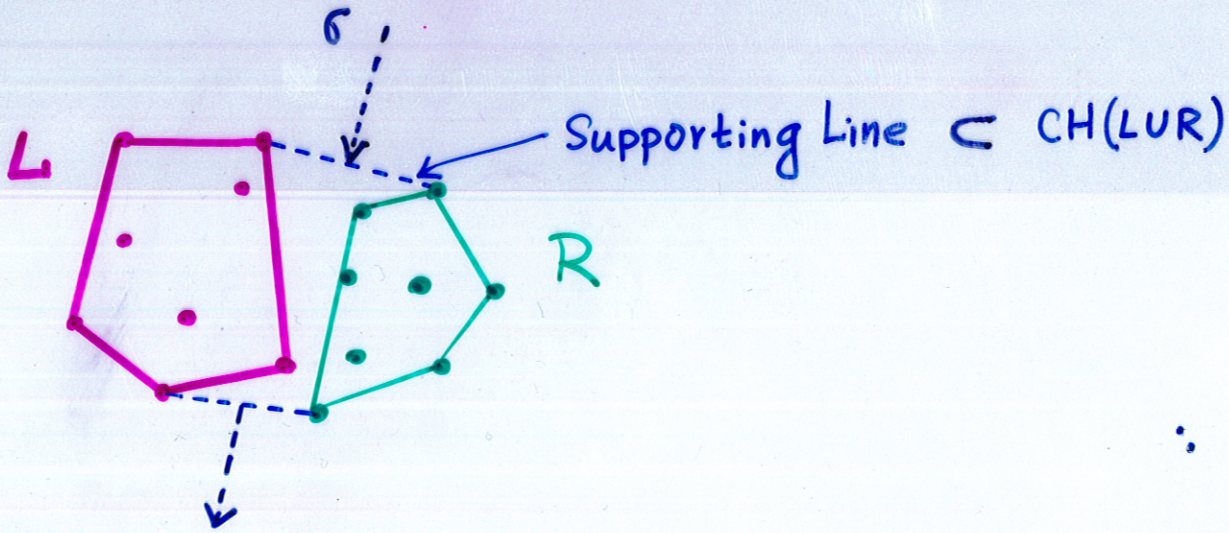


Compute Voronoi Diagram in $O(N \log N)$ time

(Divide & Conquer)



$$VD = \left(VD(L) \cap L(\sigma)\right) \cup \sigma \cup \left(VD(R) \cap R(\sigma)\right)$$

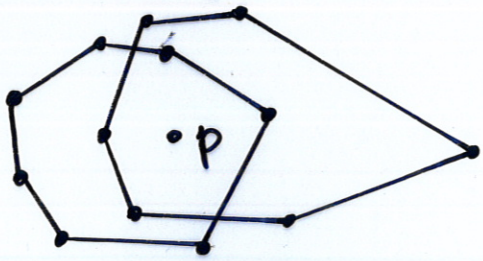


- Compute Supporting Lines $T = O(N)$

(1) Compute $CH(L)$; $CH(R)$

(2) Find interior point P of $CH(L)$, (3点を内心)

(3) P interior point of $CH(R)$?



(4) Merge vertices of $CH(L)$; $CH(R)$ / $CH(R^*)$

R^*

(5) Graham Scan

Property: P: interior point of CH

$\Leftrightarrow$ Angles of vertices of CH ↗

- Algorithm for convex hull

$$T(N) = 2T\left(\frac{N}{2}\right) + O(N)$$
$$= O(N \log N)$$