Cpts 515 Lesson 1

Theory Course, Online Course —— Focus!

1. Watch Videos and take Notes
2. Do HWs
3. You will be fine!

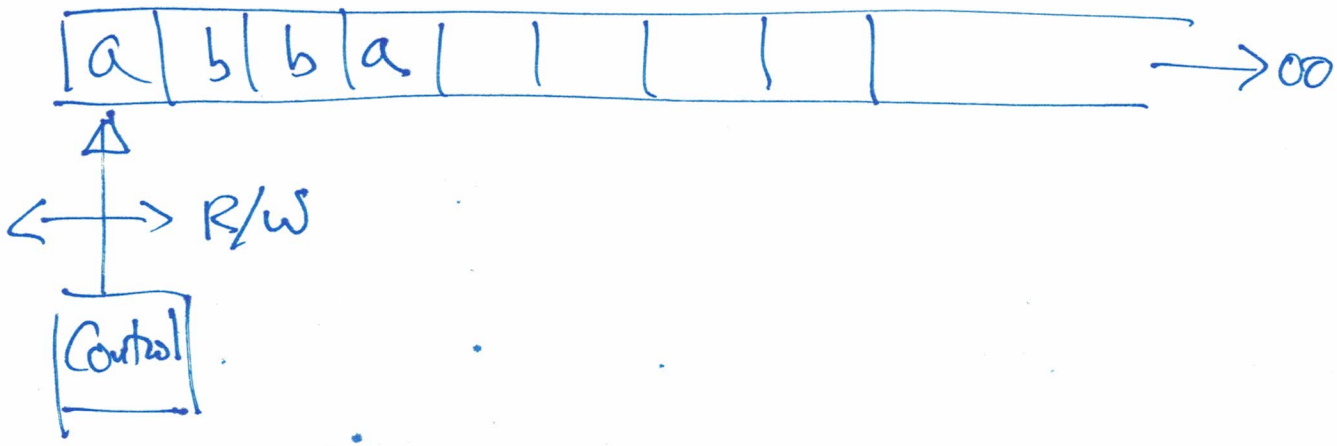
What is an alg ?

"steps", "procedure", ...

David Hilbert 1900

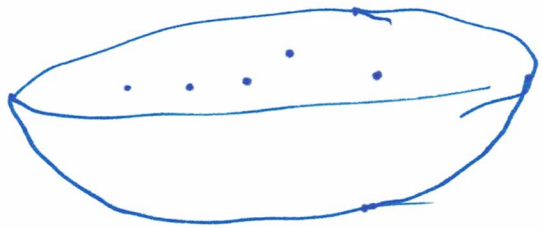
Hilbert tenth problem

$p(x_1, \ldots, x_n) = 0$ / $2x^2y - 3yz + 5 = 0$

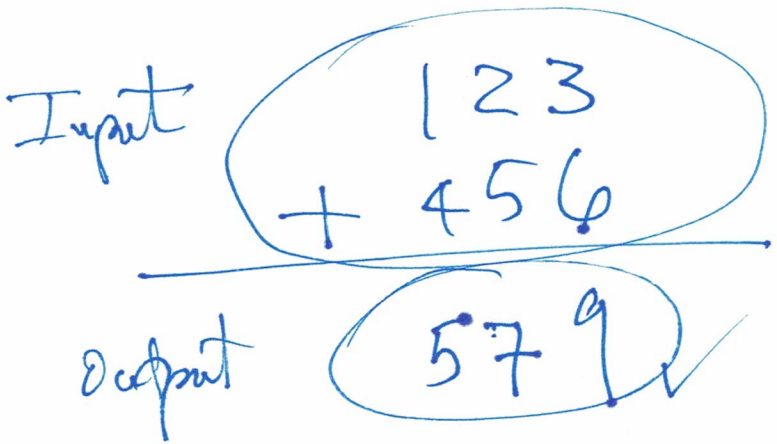


Alg = TM that always halts.

$$\begin{array}{r} 123 \\ +\ 456 \\ \hline 579 \end{array}$$



Alan Turing / Computation



1). local read/write

2). Inf. memory

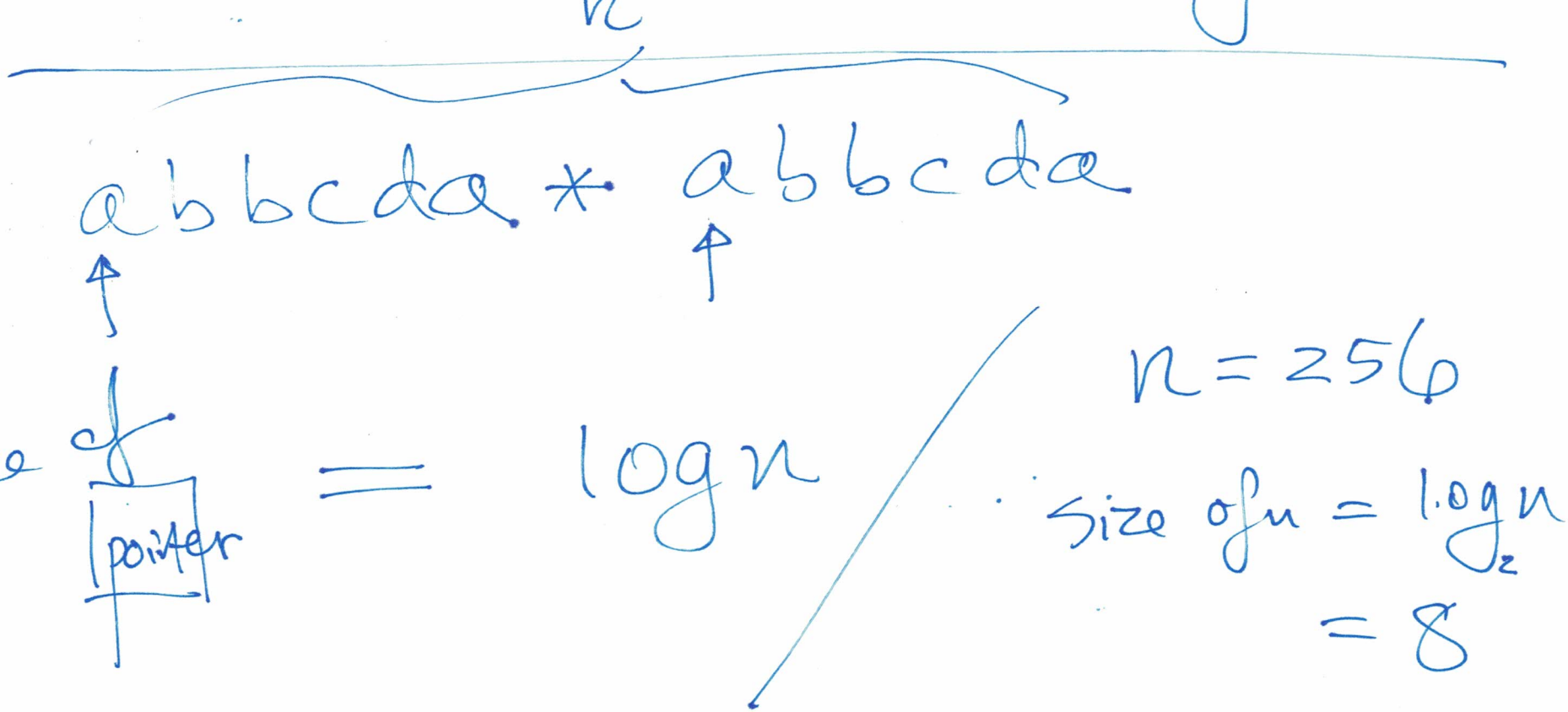
3). Finite # of rules

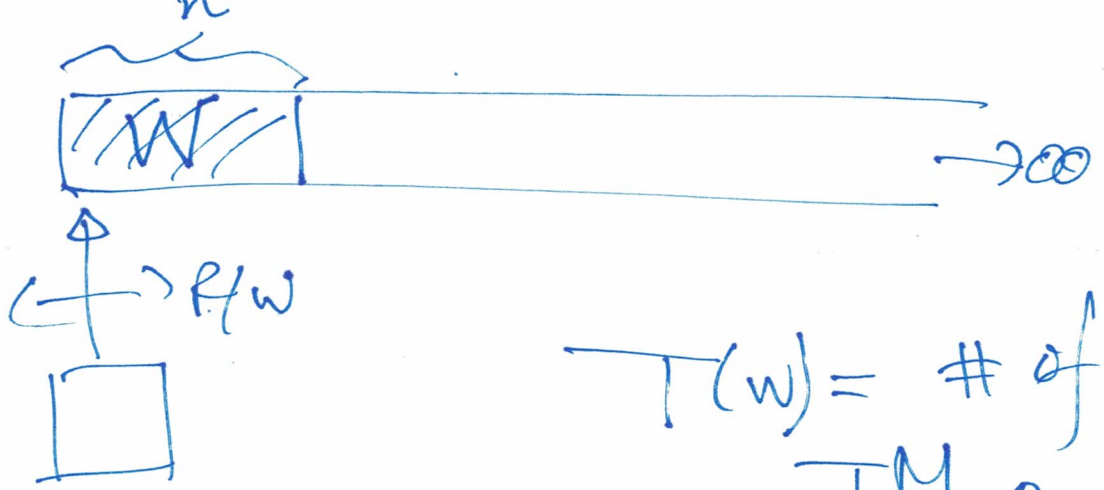
↑

local

Time complexity : ① function of input size
② NOT in sec, year, ...

Space complexity : input doesn't take memory !
if its readable only.





$T(w)$ = # of moves of TM on input w before it halts.

$T(n) =$

$$\max_{|w|=n} T(w).$$

worst case time complexity.

$$T(n) = \operatorname{avg}_{|w|=n} T(w)$$

average-case time complexity.

```
int Fib(int k){
    ⋮
}
```

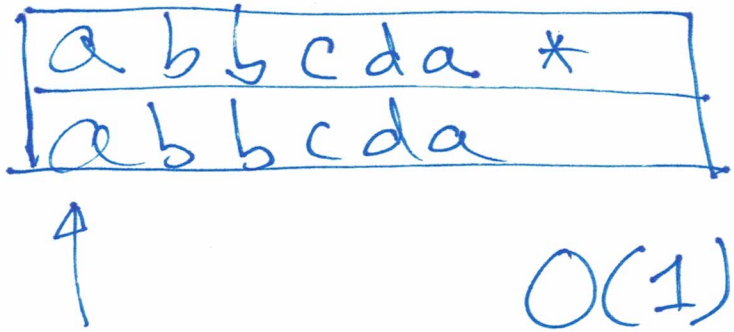
} Fibonacci function.

Input size = # of bits needed to encode k.

$= \log k$.

$$S(n) = \max_{|w|=n} S(w).$$

Wirth —— Pascal



$O(1)$ = Space complexity.

$O(n^2)$

$T(n) = O(n^3)$: There is a $c > 0$,

For almost all $n > 0$, we have

$$T(n) \leq c \cdot n^2 .$$

"for all n except for finitely many"

$\theta(n^2) = 3n^2 - 12n + 5 = O(n^2)$.

$\theta(n) = 3n - 12 = O(n^2)$.

$3n^3 + 5 \not\in O(n^2)$

4. $4 \cdot 2^{3n^2} = 2^{O(n^2)}$

Proof. Find $c > 0$ s.t.

$$4 \cdot 2^{3n^2} = 2^{3n^2+2} \leq 2^{c \cdot n^2}$$

for almost all n.

Take $c = 4$.

$4 \cdot 2^{3n^2} \neq O(2^{n^2})$.

$T(n) = O(n^2)$

A	B
$3n^2+12$	$3000n^2+12$
$O(n^2)$	$O(n^2)$

$O(n^2) = O(n^2)$

Why? Blum's speed-up theorem.

32-bit architecture
64-bit ...
128-bit ...

Recurrence — Recursive Function

$$\begin{cases} f(n+1) = f(n) + 1 \\ f(0) = 0 \end{cases}$$

$f(n) = n$ ← explicit

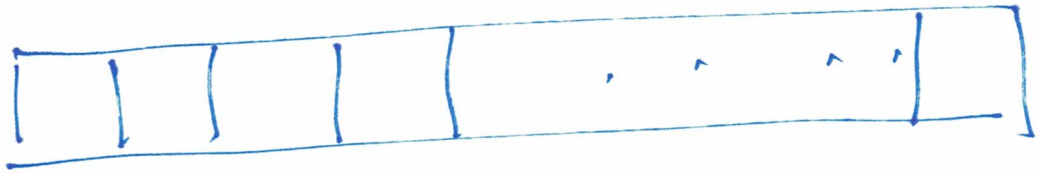
Given: $\begin{cases} f(n+2) = f(n+1) + f(n) \\ f(0) = f(1) = 1 \end{cases}$

Want: $f(n) \sim O(?)$

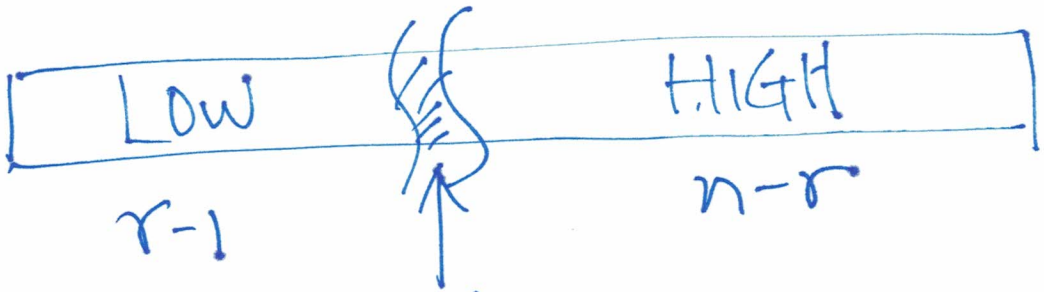
Quick-sort

Naive sorting: $O(n^2)$ —— bubble sort, insert sort, ...

Hoare. $O(n \log n)$



(1) partition: $O(n)$



(2). QS on LOW

(3). QS on HIGH.

Worst-case time complexity of QS.

(1). Write a formula

$$T_W(n) = \max_{1 \le r \le n} \left(T_W(r-1) + T_W(n-r) + O(n) \right)$$

- $T_W(r-1)$: QS on LOW (2)
- $T_W(n-r)$: QS on HIGH (3)
- $O(n)$: partition (1)

(2). Guess $T_W(n) = O(n^2)$. Want c s.t. $T_W(n) \le c \cdot n^2$. (a given)

(3). check:

$$T_W(n) = \max_{1 \le r \le n} \left(T_W(r-1) + T_W(n-r) + a \cdot n \right)$$

$$\le \max_{1 \le r \le n} \left(\underbrace{c \cdot (r-1)^2 + c \cdot (n-r)^2 + a \cdot n}_{F(r)} \right)$$

$$\leq c(n-1)^2 + a \cdot n$$

$$= c(n^2 - 2n + 1) + a \cdot n$$

$$\leq cn^2 \quad \text{when } c \text{ large for all } n.$$

max T(r)
1 ≤ r ≤ n
T
r
r=1
r=n

$$F(r) = c(r-1)^2 + c\cdot(n-r)^2 + a\cdot n$$

$$F'(r) = 2c(r-1) - 2c(n-r) = 0$$

$$r \approx \frac{n}{2}.$$

$$\max_{1 \le r \le n} F(r) = \max\left(\cancel{F(0)}\ F(1),\ F(n),\ F\left(\frac{n}{2}\right)\right)$$

$$= \max\left(c(n-1)^2,\ c(n-1)^2,\ 2c\left(\frac{n}{2}\right)^2\right)$$

$$= c(n-1)^2$$

Avg-case time complexity of QS.

(1). Write formula

$$T_{AVG}(n) = \sum_{r=1}^{n} \frac{1}{n}\left(\underbrace{T_{AVG}(r-1)}_{\text{QS on Low}} + \underbrace{T_{AVG}(n-r)}_{\text{QS on High}} + O(n) \right)$$

(2). Guess $T_{AVG}(n) = O(n \log n)$. Want c:

$$T_{AVG}(n) \le c n \log n.$$

(3). Check.

$$T_{AVG}(n) = \sum_{r=1}^{n} \frac{1}{n} \left(T_{AVG}(r-1) + T_{AVG}(n-r) + a \cdot n \right)$$

$$\le \frac{1}{n} \sum_{r=1}^{n} \left(c(r-1)\log(r-1) + c(n-r)\log(n-r) + a \cdot n \right)$$

$$\log(n!) = \sum_{r=1}^{n} \log(r-1) + \sum_{r=1}^{n} \log(r-1)$$

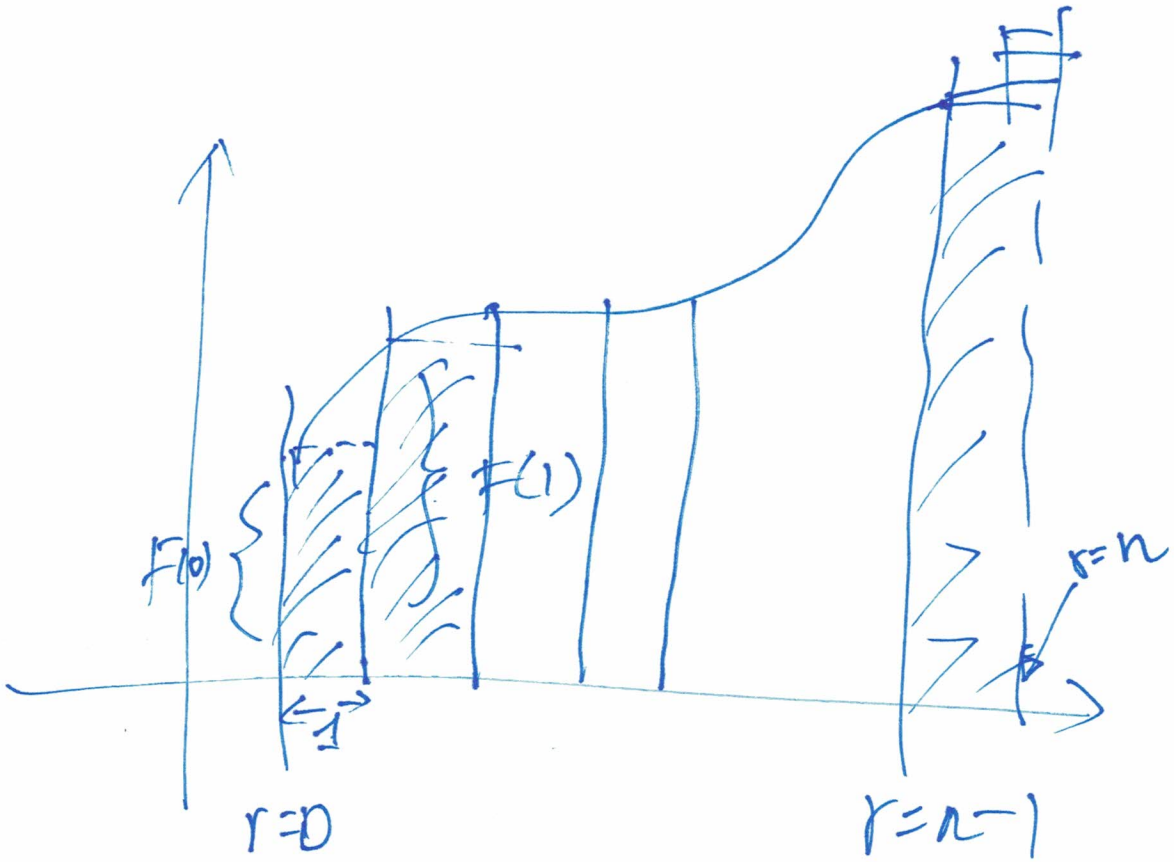
$$= \sum_{r=1}^{n} \log(r-1)$$

$$+ n.n$$

$$= \sum_{r=1}^{n} 2\log(r-1)$$

$$+ n.n$$

$$T(n-1)$$



$$\sum_{r=1}^{n} F(r-1) \leq \int_{0}^{n} F(r)\, dr.$$

$$\int x \ln x \, dx = \frac{x^2}{2} \ln x - \frac{x^2}{4} + C$$

$$\log x = \frac{\ln x}{\ln 2}$$

$$\int x \log x \, dx = \frac{1}{\ln 2} \int x \ln x \, dx$$

$$= \frac{x^2}{2} \log x - \frac{x^2}{4 \ln 2} + C$$

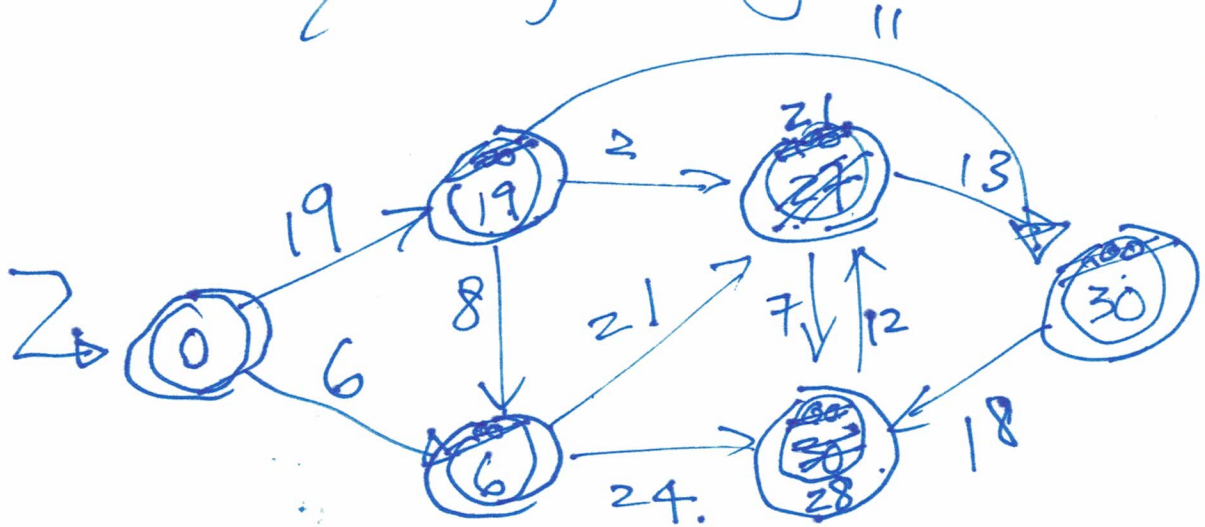
$$\leq \frac{2c}{n} \int_0^n x \log x \, dx + a \cdot n$$

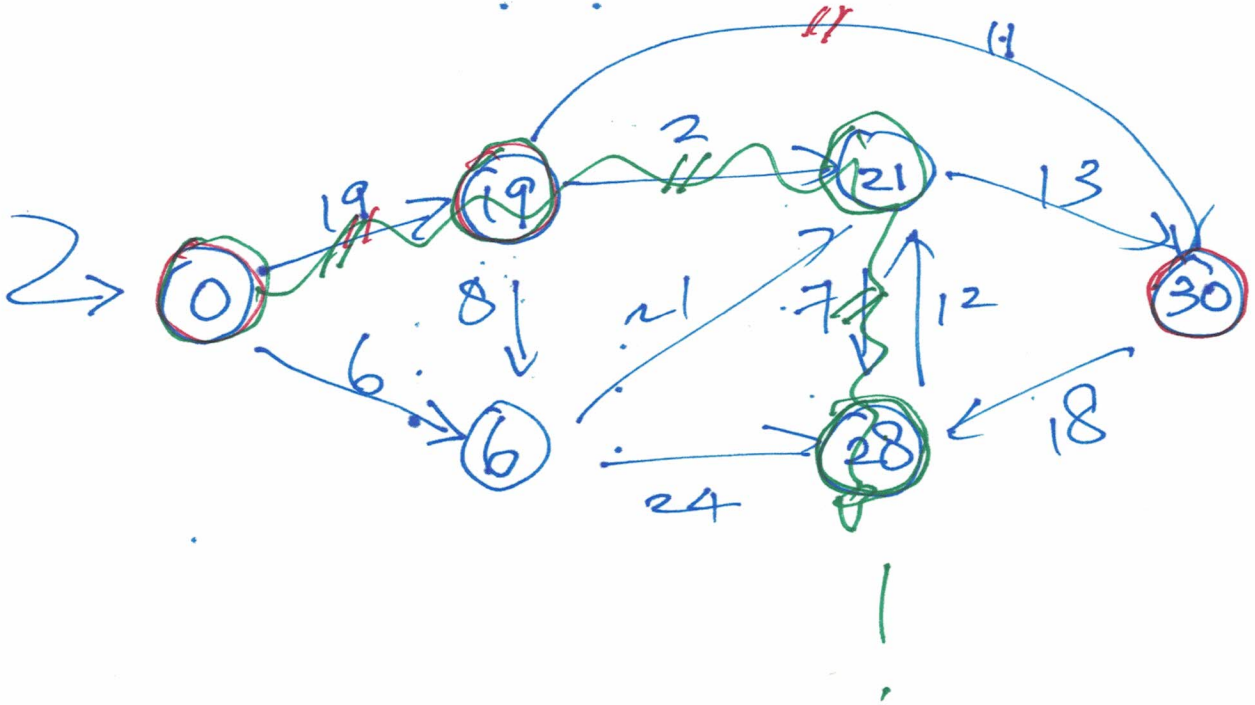
$$= \frac{2c}{n} \left(\frac{n^2}{2} \log n - \frac{n^2}{4 \ln 2} \right) + a \cdot n$$

$$\leq c\, n \log n \quad \text{for large } c.$$

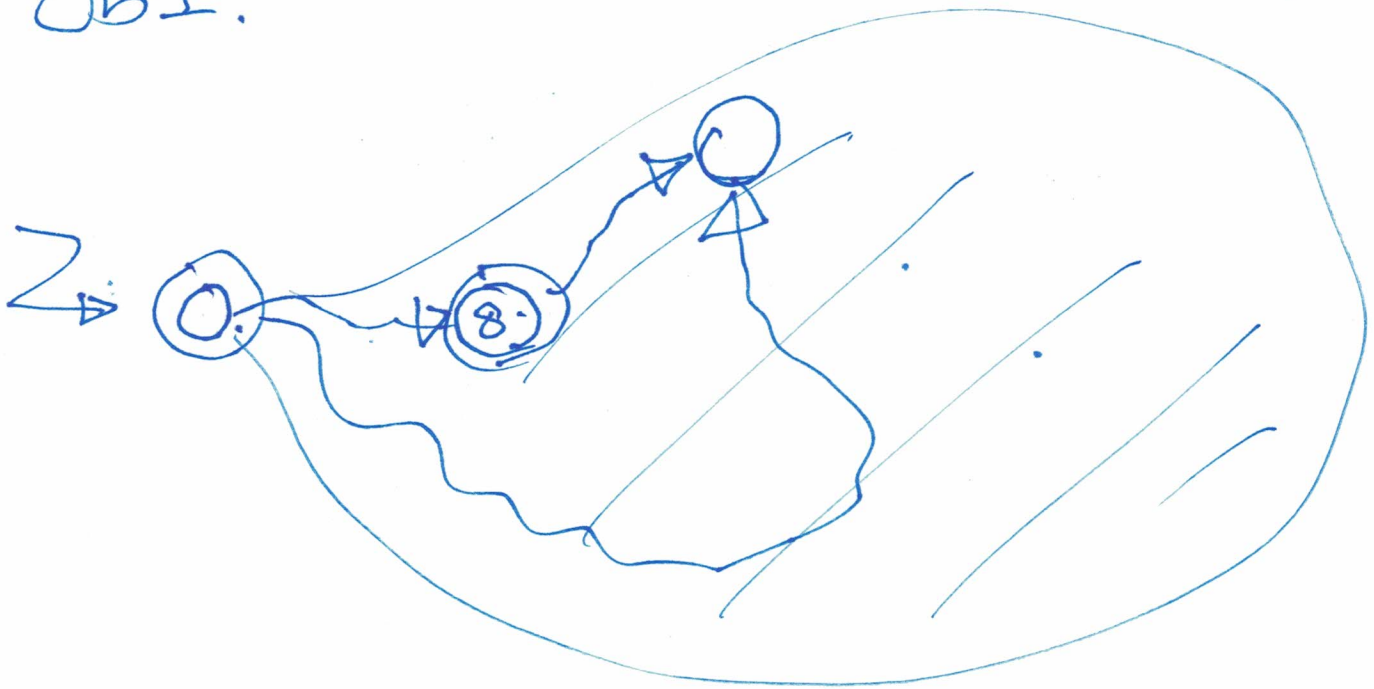
Shortest path / Djikstra Alg.

Weighted graph

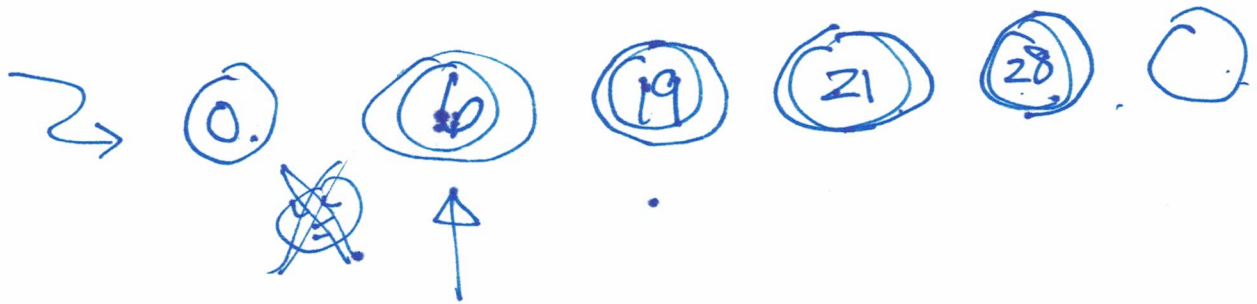




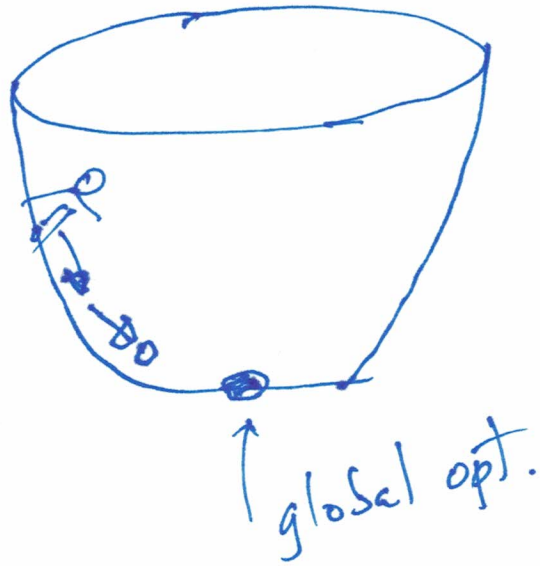
ob1.



ob2.



Greedy. Divide & Conquer. Dynamic Programing.



Dynamic Programming: LCS Alg

α = a b b a c a d a b ..... fish

β = c b a d a b d a ..... human

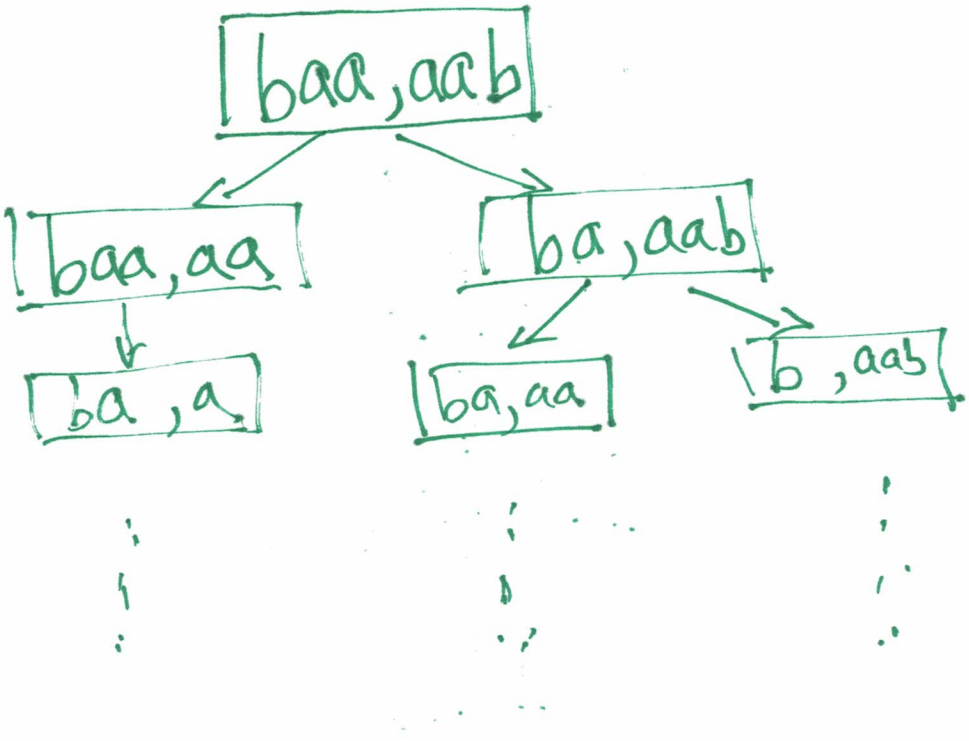
A - T - C - G Alignment

Naive: Enumerate all subseq's of α and of β

Pick longest common subseq.

$|\alpha| = n$, $|\beta| = m$. ⟶ Goal: $O(nm)$.

aba, aba
ab, ab
a, a
λ, λ



Exp time!

$O(2^n)$

Alg ← Study the problem.

$LCS(\alpha, \beta)$ has some properties:

(1). $LCS(Aa, Ba) = LCS(A, B)\,a$

(2). $LCS(Aa, Bb) =$ (longer of $LCS(Aa, B)$ and $LCS(A, Bb)$)
$a \neq b$

(3). $LCS(A, \lambda) = LCS(\lambda, B) = \lambda$.
(λ is empty string, NULL).

We have the alg!

A_1, B_1

A_2, B_2

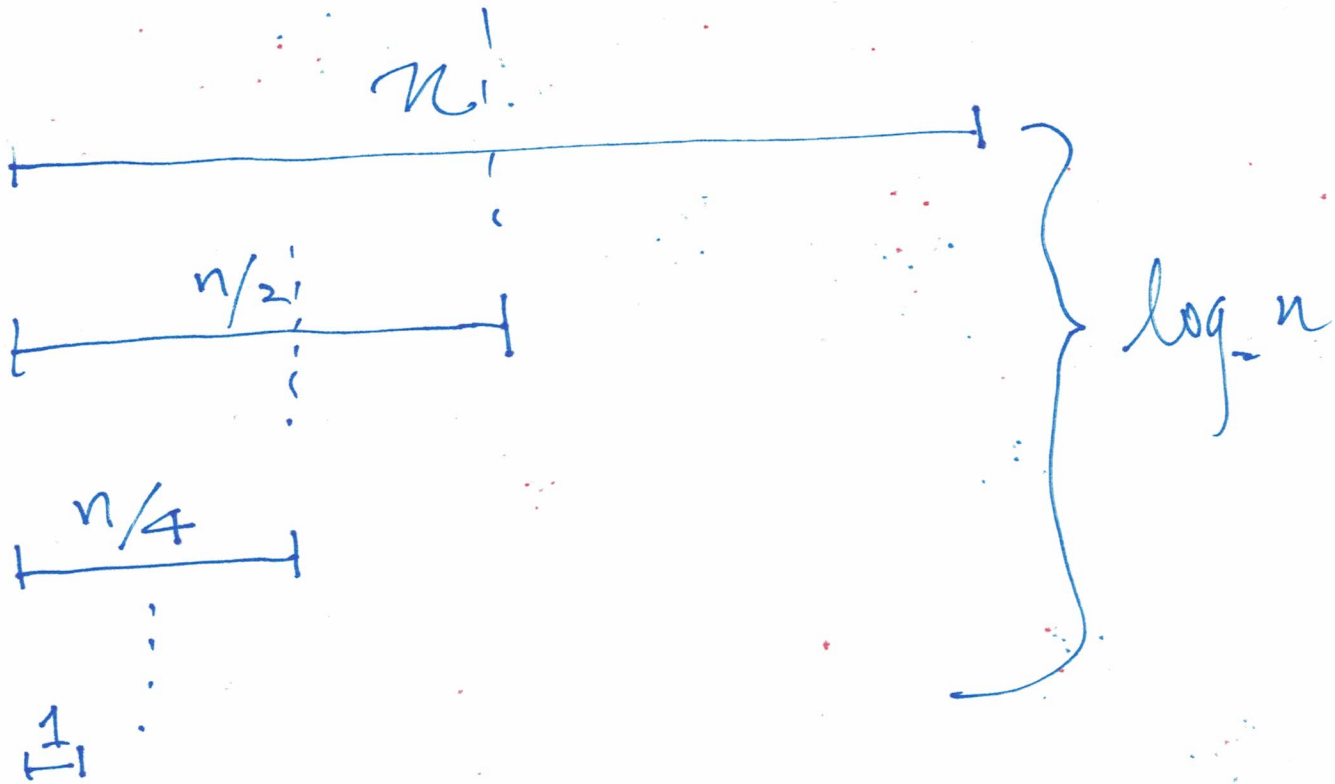
A_a, B

A_a, B_b

a
a
b
a
m
a
s
t
r
i
A

Divide & Conquer

$\log_2 n$



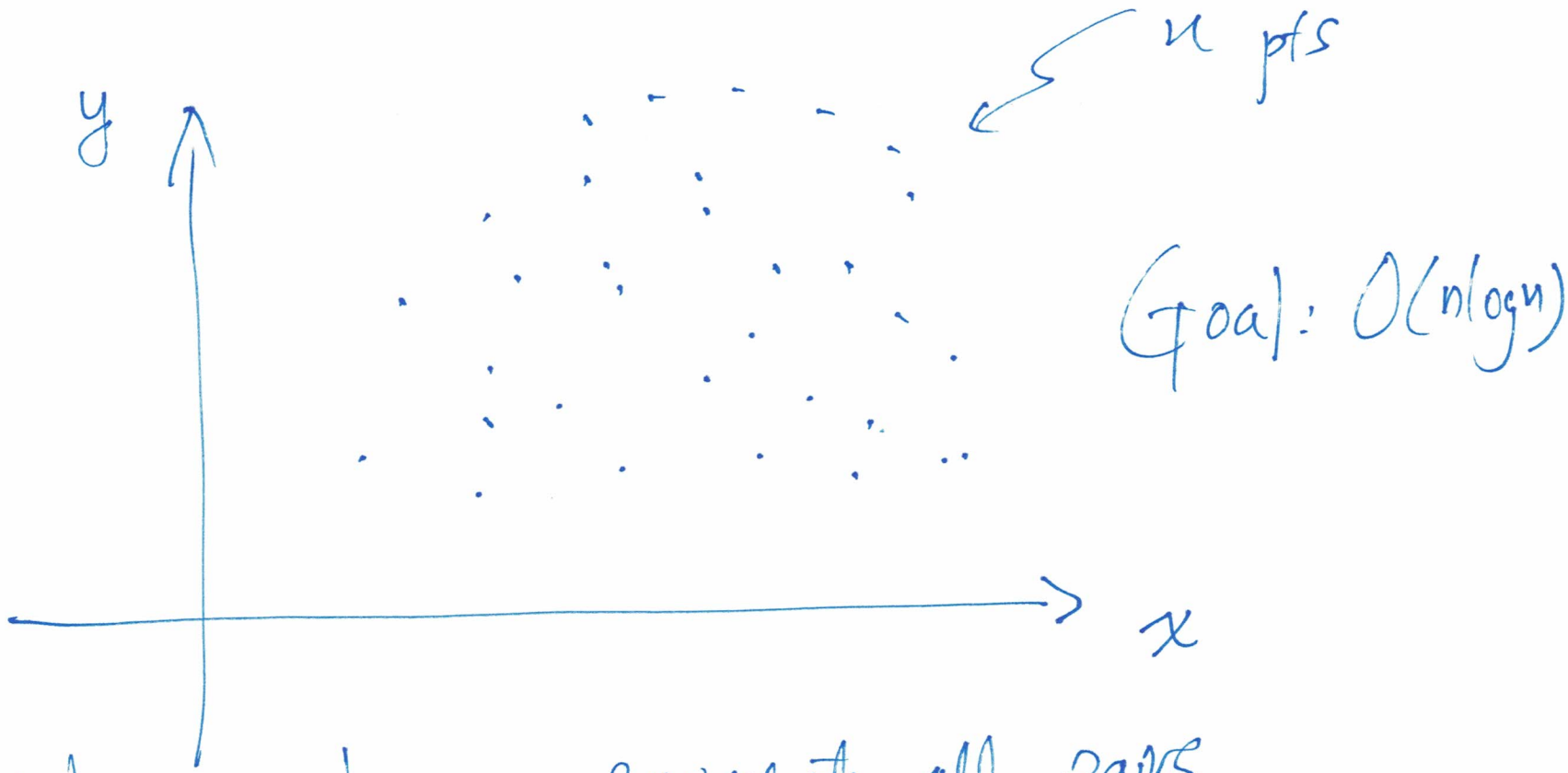
Sub1
Sub2

$$d(\vec{x}, \vec{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

where $\vec{x} = (x_1, x_2)$

$\vec{y} = (y_1, y_2)$

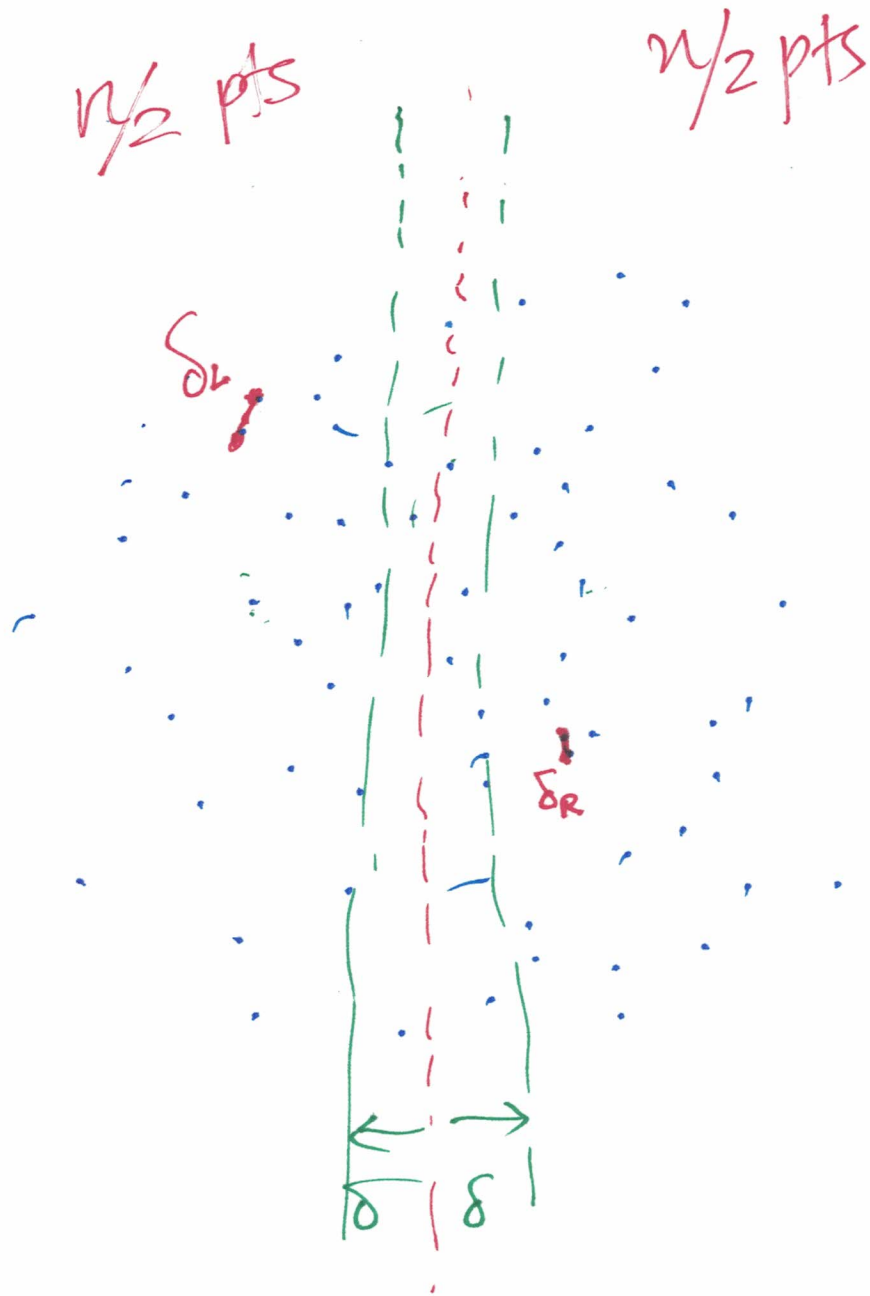
Closest-Pair



Goal: $O(n \log n)$

Naive alg: $O(n^2)$

- enumerate all pairs $\Sigma\ O(n^2)$
- Compute distance between two pts in each pair
- pick smallest.



$\delta = \min$ of δ_L and δ_R.

d
d
impossible

Global closest pair = either

δ_L, δ_R, or

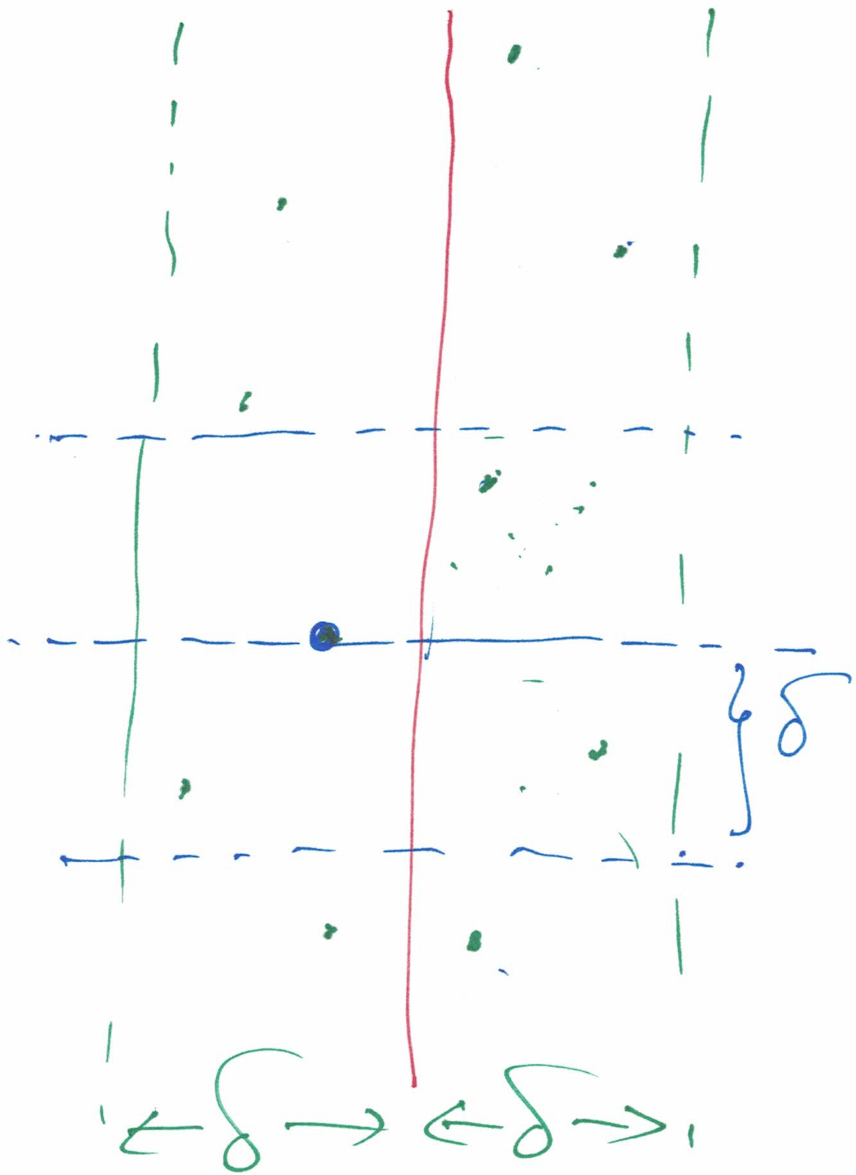
the closest pair between a pt on the left and a pt on the right.

$O(n)$

$n/2 \cdot 8$

$= 4 \cdot n$

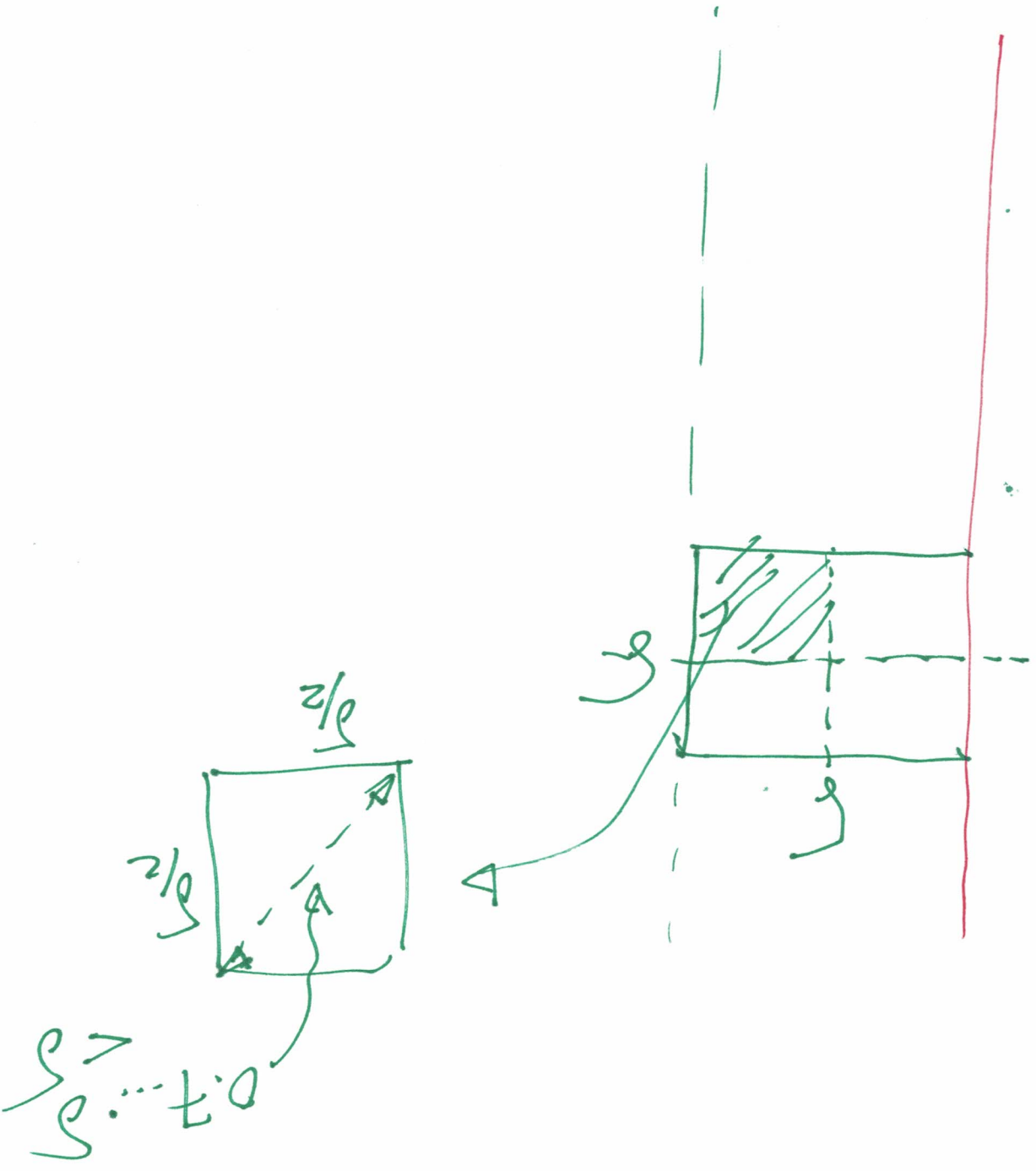
pairs.



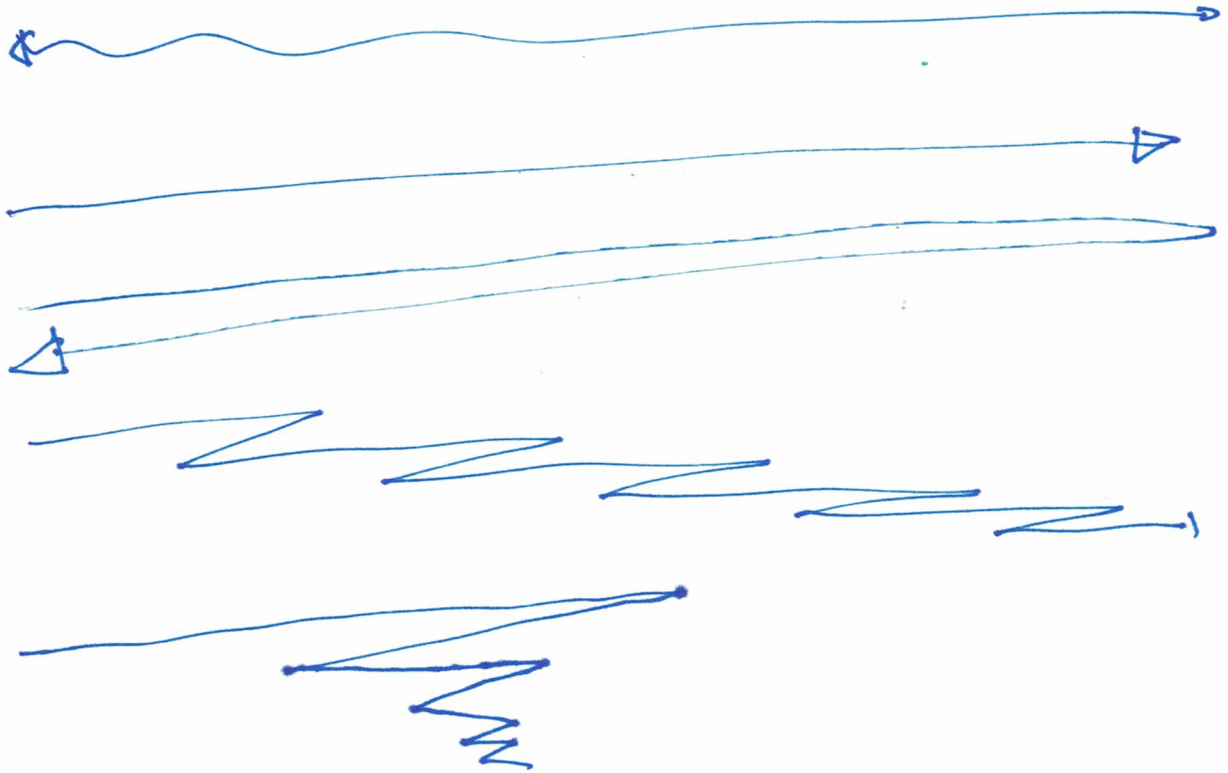
pts are sparse

$n/2$ on the left strip.

each such pt, you can pick at most 8 pts on the right


δ/2
δ/2
0.7 ... δ

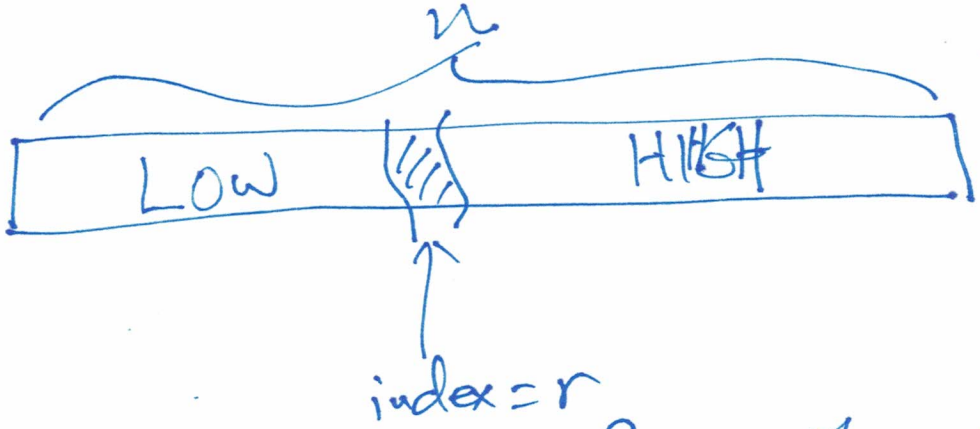
Lesson 3.



Linear-time selection.

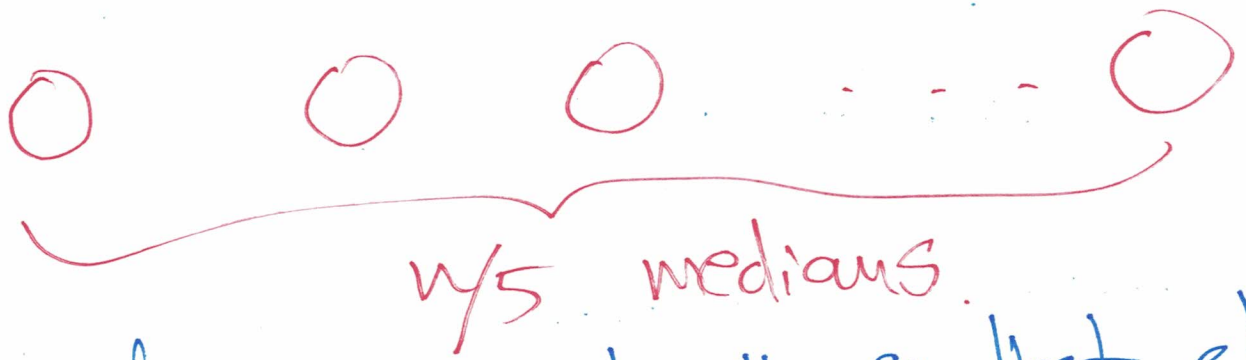
Quick-select —— avg case $O(n)$.

Look for: a linear time $O(n)$ selection alg.



Select i-th smallest # from the given n #'s.

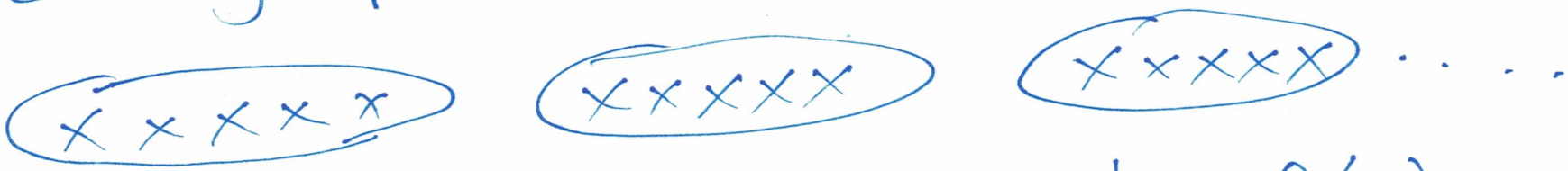
(3). Each group, after sorted, has a median. Totally, we have $n/5$ medians.



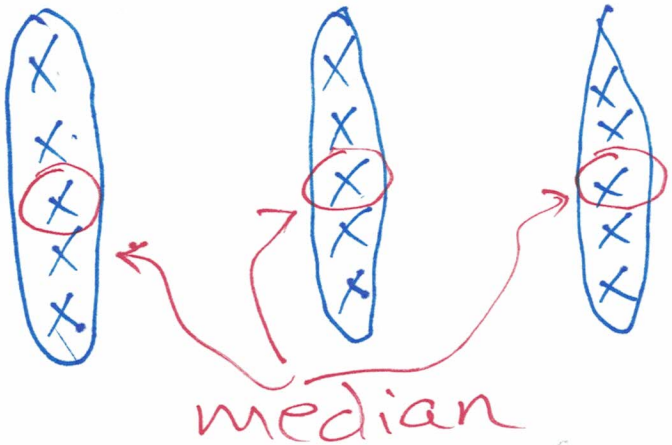
(4). Select the $n/10$-th smallest element from the $n/5$ medians.

(5). Swap MM with the fist element in the original array of n numbers.

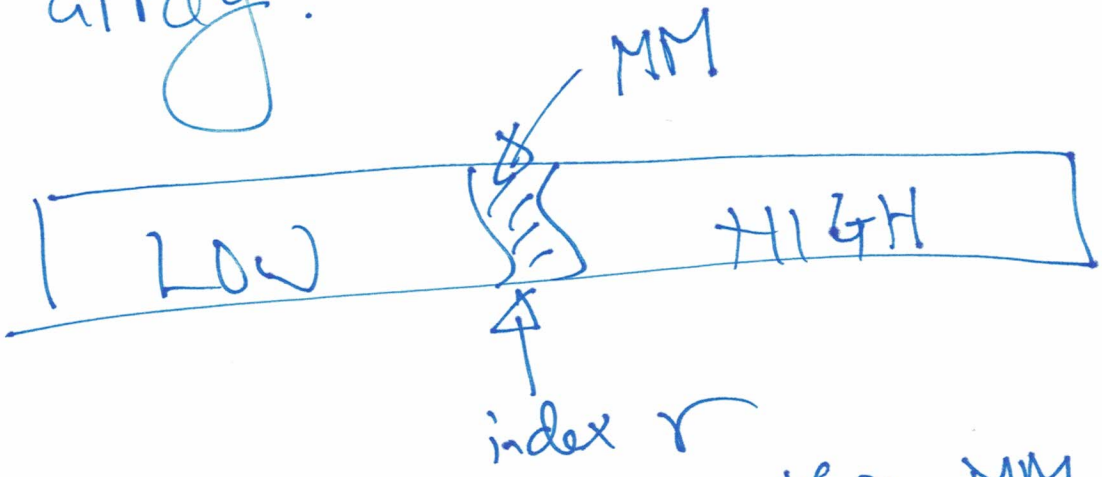
(1). Cut the n numbers into $n/5$ groups each group has 5 numbers.



(2). Sort each group. ← takes $O(n)$ time



(6). I have, after the swaps, an updated original array. I do partition on this array.



(7). if $i == r$, return this MM as the i-th smallest;

if $i < r$, run: select i-th smallest from LOW

if $i > r$, run: select $(i-r)$-th smallest from HIGH.

$$|LOW| \geq \frac{3}{10}$$

$$|HIGH| \geq \frac{3}{10}$$

$$|HIGH| \leq \frac{7}{10}$$

$$T_W \left(\max(|LOW|, |HIGH|) \right) \geq \frac{7}{10}$$

$$\therefore T_W \geq \frac{7}{10}$$

$$\therefore T_W \geq 1$$

$$\therefore T_W \geq 1$$

$$T_W(n) = T_W\left(\frac{7n}{10}\right) + T_W\left(\frac{n}{5}\right) + O(n)$$

Guess $T_W(n) = O(n) \le c \cdot n$ for some c.

Check:

$$T_W(n) = T_W\left(\frac{7n}{10}\right) + T_W\left(\frac{n}{5}\right) + a \cdot n$$

$$\le c \cdot \frac{7n}{10} + c \cdot \frac{n}{5} + a \cdot n$$

$$= c \cdot \frac{9}{10} \cdot n + a \cdot n$$

$$\le c \cdot n \quad \text{for a large } c.$$

< 1 !!!

$$T_w(n) = \underbrace{??}_{\text{step(7)}} + \underbrace{T_w(n/5)}_{\text{step(4)}} + O(n)$$

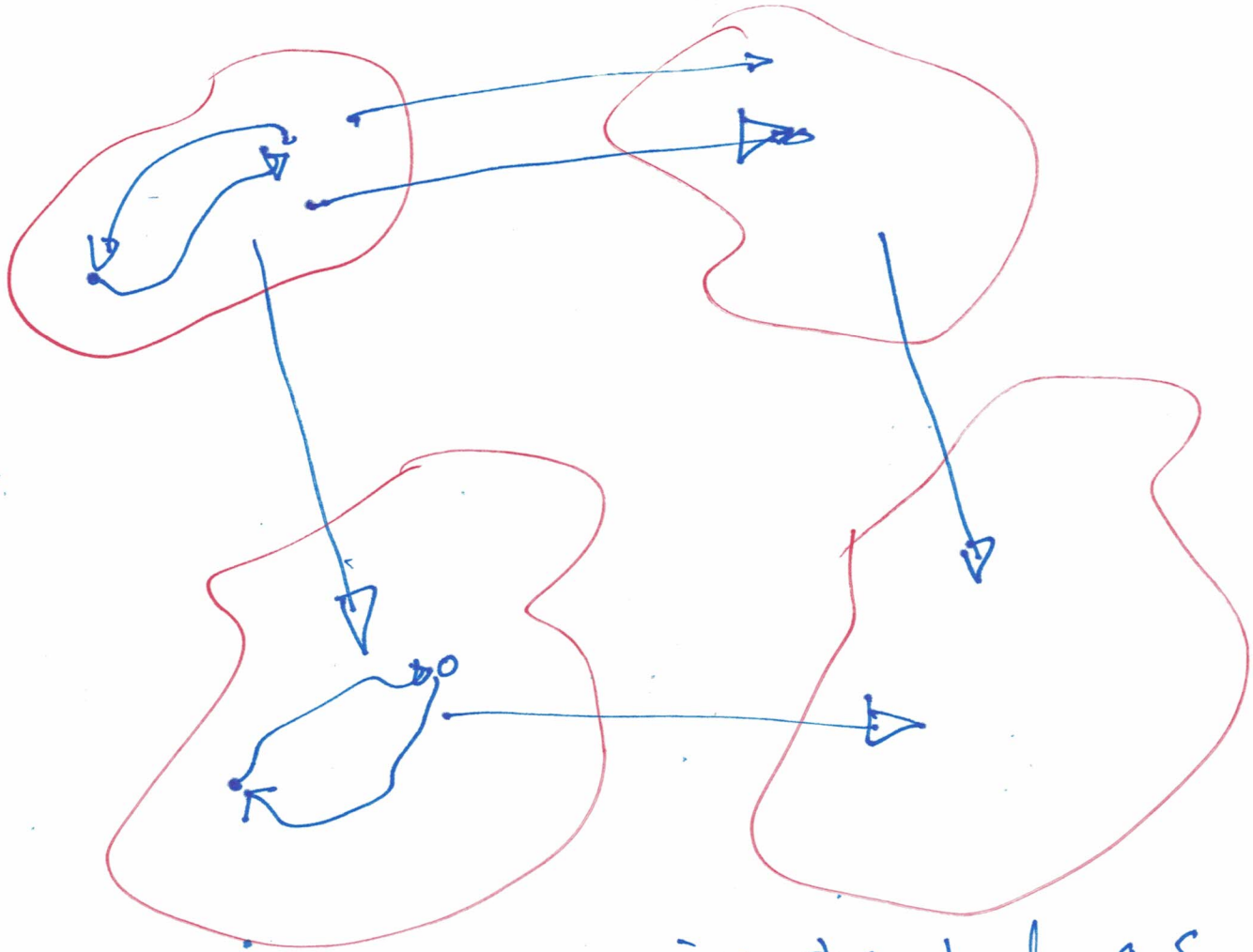
$?? = \max$ of $T_w(|LOW|), T_w(|HIGH|)$ .... .

How many medians $\leq MM$?

$n/10$

How many #'s in a group $\leq$ the median of the group ?

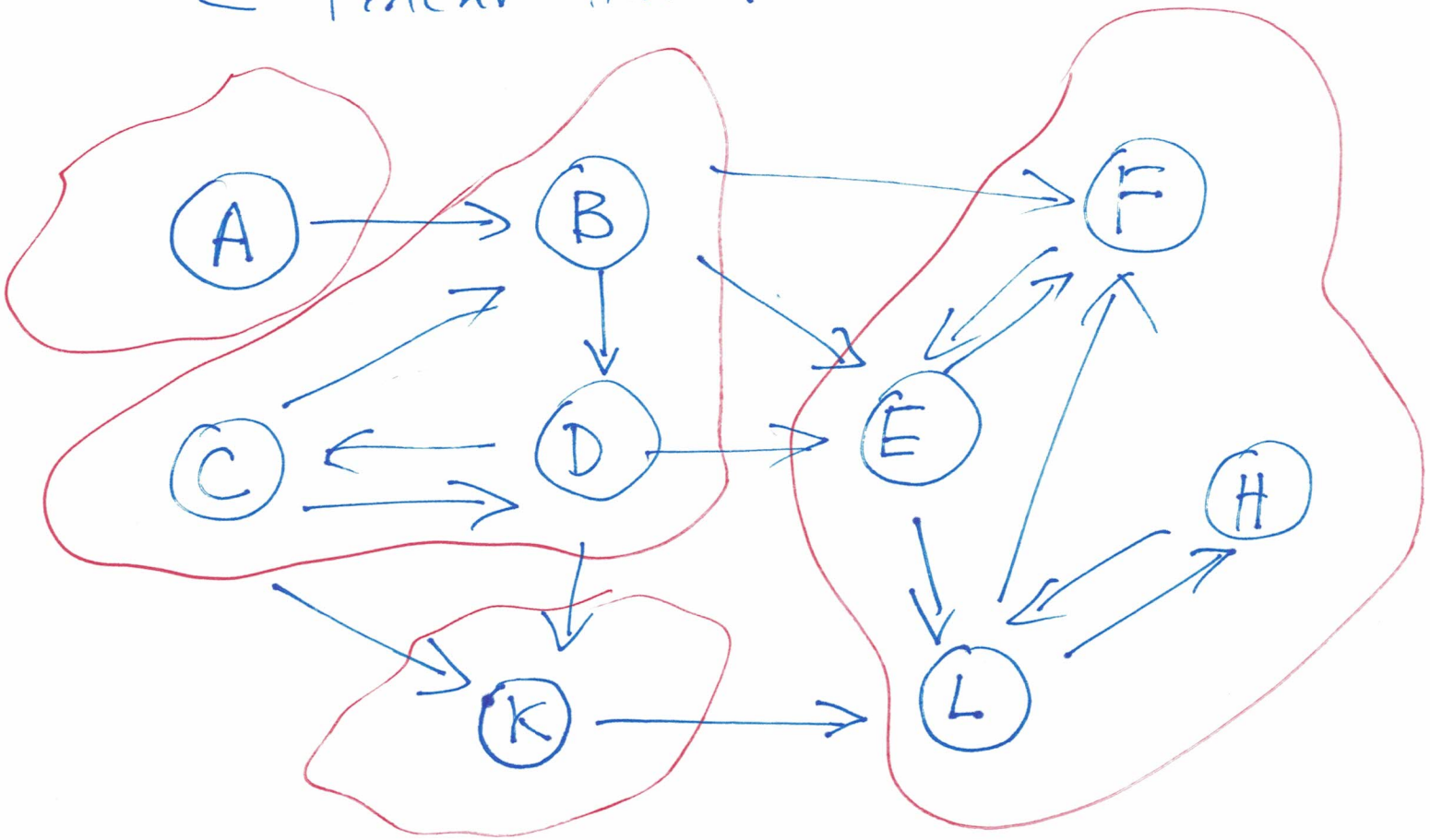
We have at least $3 \cdot \frac{n}{10}$ #'s in the original array $\leq MM$.



if each SCC is treated as a big node, the SCC's form a DAG of the big nodes.

SCC-alg / Tarjan Alg.

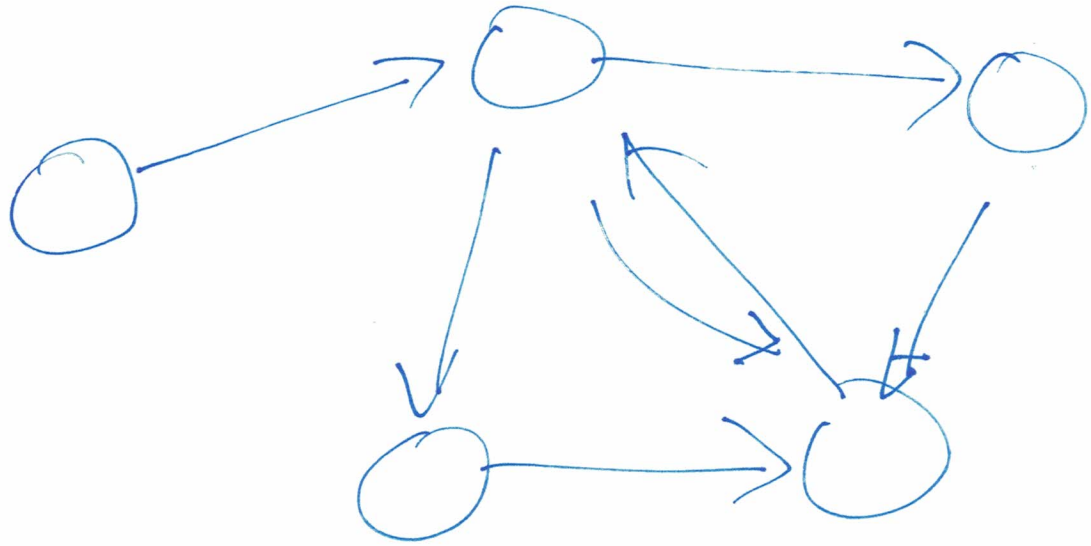
↑ linear-time.



A
9:16
B
9:14
C
9:15
D
9:12
E
9:09
F
9:08
G
9:06
H
9:07
I
9:11
K
9:11
9:13
9:01
9:10
9:02
9:03
9:05
9:04

A
q:16
C
q:15
K
q:11
D
q:12
B
q:14
L
q:06
E
q:09
F
q:08
H
q:07

A
C
K
D
B
M
L
H
F



a walk can contain loops.

a simple walk ← no loops.

"Loop analysis".

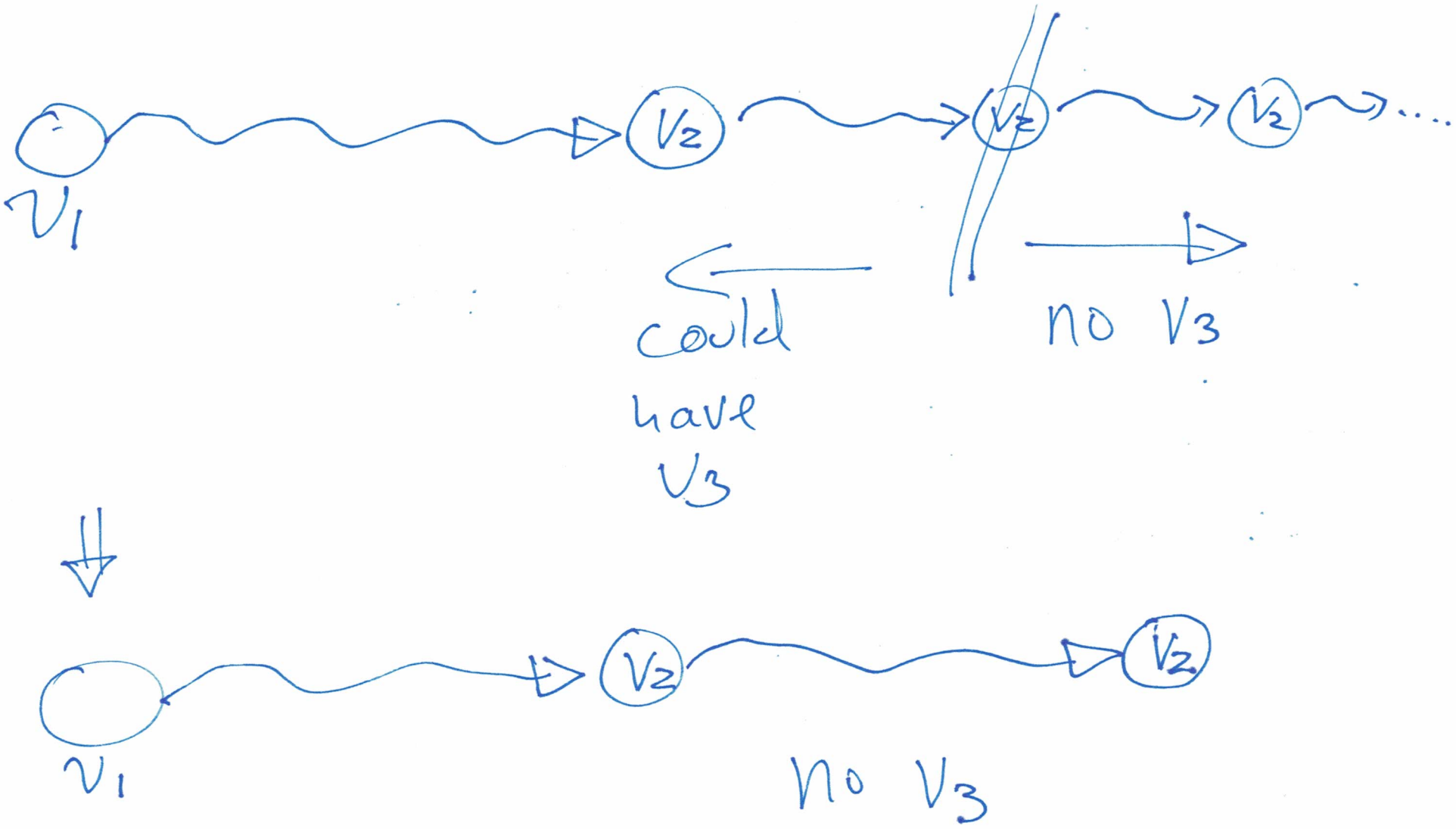
infinite walk.

Given: a directed graph G
three nodes (distinct)
v_1, v_2, v_3

Question: $\exists$ an inf. walk α where
α starts from v_1 and
passes v_2 for inf. # of times
but passes v_3 for only finite
# of times ?

---

v_1
v_2
v_3
h0


v_1
v_2
v_2
v_2
could have v_3
no v_3
v_1
v_2
v_2
no v_3

ALG:

DFS

Check $V_1 \rightsquigarrow V_2$

Check $V_2 \rightsquigarrow V_2$ but in between, no V_3.

either DFS or SCC.

drop V_3 from G

ALG: use DFS to check $V_1 \rightsquigarrow V_2$.

Drop V_3 from G, and get G'.

Run SCC on G'.

check the SCC containing V_2 does contain a loop.

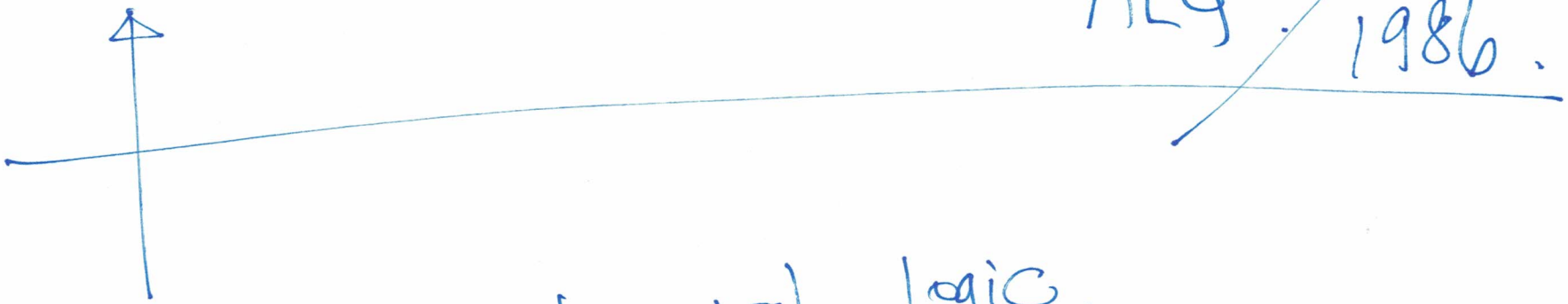
v_2 has a self-loop or

the |scc| > 1

///

SCC —— importance.

LTL model-checking, Vardi-Wolper ALG / 1986.



linear-time temporal logic.