

---

---

---

---

---

---

---

---

---

---



## Range Minimum Query Etc. good

fenwick  $O(\log n)$  → Quicksun

segment → min max

lazy seq → update վեհաց մինմաք

$O(\log n)$  գույք  
Դմաց update աղջկա

RMQ Նույնական գույքը  
process →  $O(n \log n)$   
բարձրակարգ  $O(1)$

	32	36	21	84	42	14	7
0	0	1	2	3	4	5	6
1	32	36	21	84	42	14	7
2	(21)	91	14	7	7	7	7

$2^i = 2^j \rightarrow$

$i = j$

$\min_{i=0}^{j-1} \text{min } 0-3$

$\min_{i=j}^{j+1} \text{min } 1-4$

$\min_{i=j+2}^{j+3} \text{min } 2-3$

$$\text{սահման } k = \lfloor \log(y-x+1) \rfloor \rightarrow \text{օօօ } \log$$

↑ վեհաց

Որտե՞ղ  $k$  է և լայ 2<sup>k</sup> ան

Etc օօօ  $x-y$   
 $0-b$

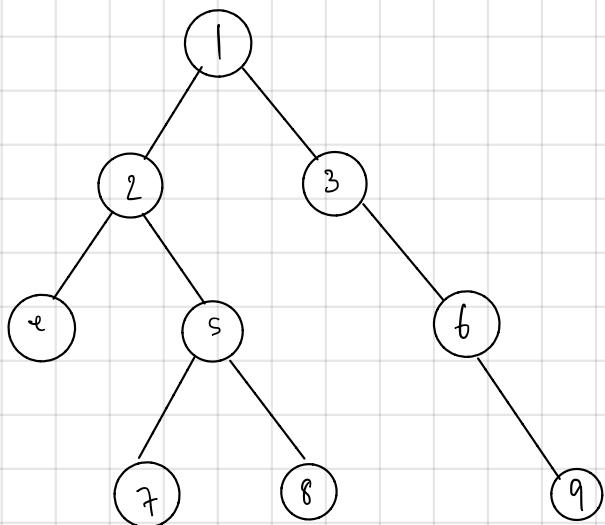
$$k=2$$

$$\text{ինչ } \min(pcc[2][0], pcc[k][y-2^k+1])$$

↑ մին մին  
↑ որդիում

## Lowest Common Ancestor

2. mic Binary lifting \* (recommended)



$$dp[i][j] = \text{พ่อที่ } 2^i \text{ ของ } j$$

$$lv[i] = \text{level ของ } i$$

for ( $i = 1 ; i \leq \log n ; i++$ )

    for ( $j = 1 ; j \leq h ; j++$ )

$$dp[i][j] = dp[i-1][dp[i-1][j]]$$

② For ( $i = \log n ; i \geq 0 ; i++$ )

    if ( $dp[i][a] \neq dp[i][b]$ )

$$a = dp[i][a], b = dp[i][b]$$

    ให้ shift level ขึ้นพื้นที่ลง

Conclusion : ① ห้าม level ห่าง

② เลื่อน level ขึ้นพื้นที่ลง ก็จะหาได้

LCA หาโหนดที่มีร่วมกัน โหนด根 โผล่根  
ในระดับชั้น (level)  $\downarrow$

1. จุดเด่นที่ Euler Tour / RMQ Etc. FoodWeb

E = เก็บเวลาเข้าออก (เวลา ไป นี่คือ โผล่ตัวเอง)

L = เก็บ level ของโหนดใน array C

h = เก็บ ว่า ปริมาณ Euler Tour ครั้งแรกของโหนด  
เวลาเข้าก็จะเป็น ไปตาม เวลาที่ไป

$$\text{LCA}(5,8) = E[\text{RMQ}(4,9)] = E(5) = 3$$

base case  $dp[0][j]$  dfs มาก แล้วน้ำหนักต่ำ

① ทำ LCA บันทึก (จับเวลาหนักยังไงก็ได้แล้วท้า)

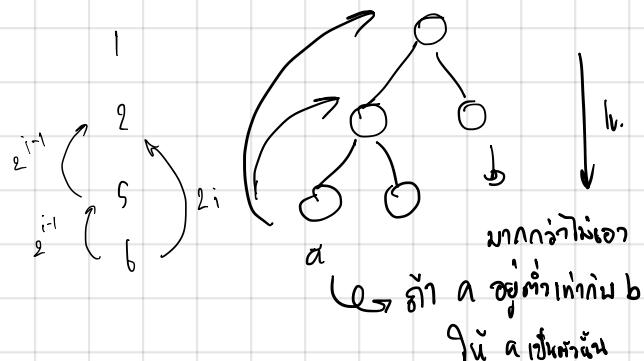
$a, b$ ,  $|lv[a]| > |lv[b]|$  มากไปด้วย  
ในระดับชั้นที่ไม่ถูก

for ( $i = \log n ; i \geq 0 ; i--$ )

    if ( $|lv[dp[i][a]]| \geq |lv[dp[i][b]]|$ )

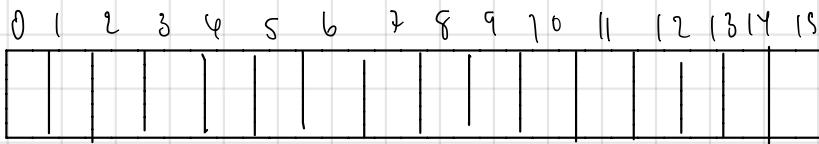
$$Etc. \quad a = dp[i][a];$$

ที่ต้องทราบ  
ให้ล = เลขเชิง  
ฐาน 2 แบบ  
เดียว ไม่ต้อง  
มีเลข

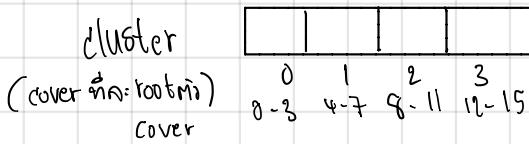


## Square Root Decomposition $O(n \sqrt{n})$

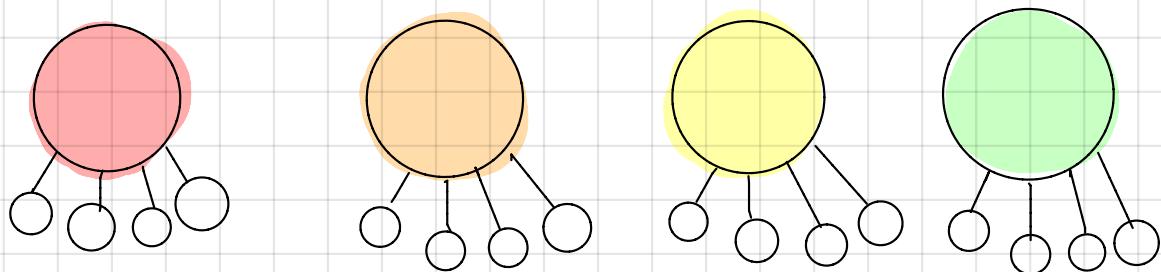
แบบ segment tree



$$\sqrt{16} = 4$$



UPDATE



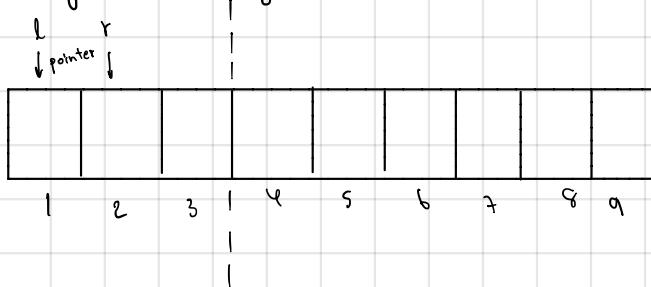
ช่วง ก็เป็นช่วงที่อยู่ใน cluster นั้นๆ cluster ด้วย

ช่วง ก็จะหดลงช่วงนั้นๆ ถ้าจด cluster

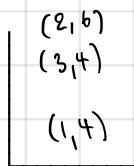
ก็จะ sticker ไว้แล้วปักไว้ต่อไป  
(ปักไว้ต่อ cluster ต่อๆ กัน)

หากซึ่ดใน cluster ใหม่  
เพื่อ แบบค่าในช่วงนั้นๆ  
เราปักต่อๆ กันจนหมด

MQ Algorithm Query Square Root Decomposition Etc. Antique competitor



นี่ query ของเรื่องปั้นกลุ่มๆ ตามๆ กัน



sort ตาม  
ห้องไปทาง  
พอกำลังสี่  
2 ไปกล่อง 1  
(ห้องแรก, ห้อง  
สอง 1 ห้อง 2  
(ห้องแรกต้องเล็กกว่า))



$1 - \text{sqrt}$   
 $\text{sqrt} + 1 - 2\sqrt$   
 $2\sqrt + 1 - 3\sqrt$

Algo แบบ offline

Counting sort  
ฟาร์ม

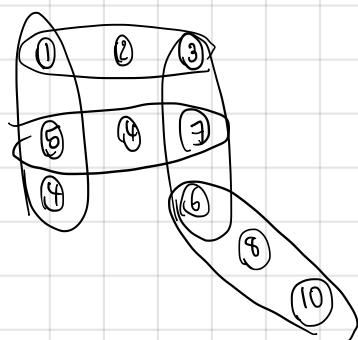
1. เลื่อนหัวหัวตาม  
ห้องสี่

2. นำไป l r  
อยู่จุดเดิม

3. เลื่อนหัวหัวตาม

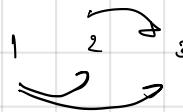
พอหัวหัวไปแล้ว ก็เลื่อน 1 ไปต่อ เรื่อย  
พิมพ์ลง array counting sort

Hypergraph  $\rightarrow$  1 เรื่องเชื่อมคลายในแต่ Etc. NC\_Teleport



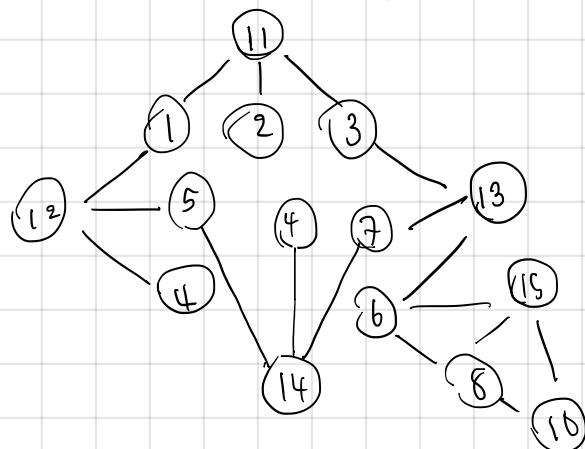
ถ้าสร้างเป็น grahp

Etc.  $H=1000$  คือ  $H$  ต้อง  $\geq 2$  เพื่อให้  $H$



ก็ต้องมีความกว้าง และ BFS จะทำ

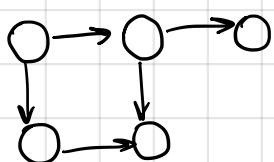
มา 2 รอบ



SCC strongly connected component

$\hookrightarrow$  ไปหาคันได้บัด

Connected graph



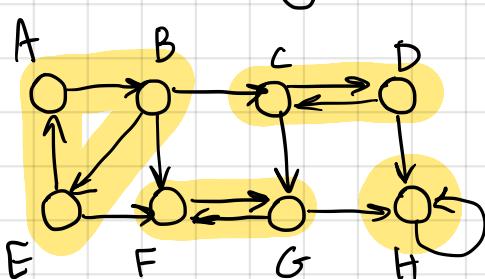
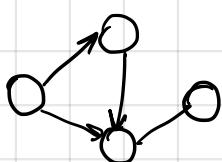
2 connected component

มี 2 Algo

1. Kosaraju

DFS 2 รอบ

เข้าจ่อๆ เช่นนี้



2. Tarjan

DFS ร่องเดียว

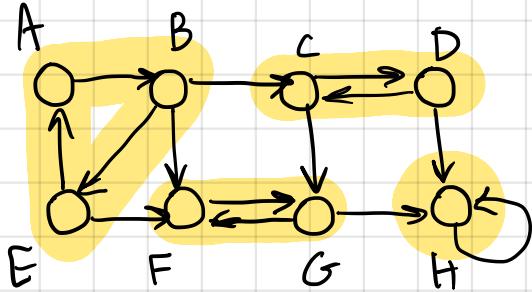
เดินทาง ที่ยังลื้อกล่า เข้าใจยาก

Etc. Natty Land

FC - Feeding Frenzy

The Great ATM Robbery

## Kosaraju



มี 2 graph

1. ปั๊ม G

2. กราฟลิปส์  $G^T$

## Algorithm

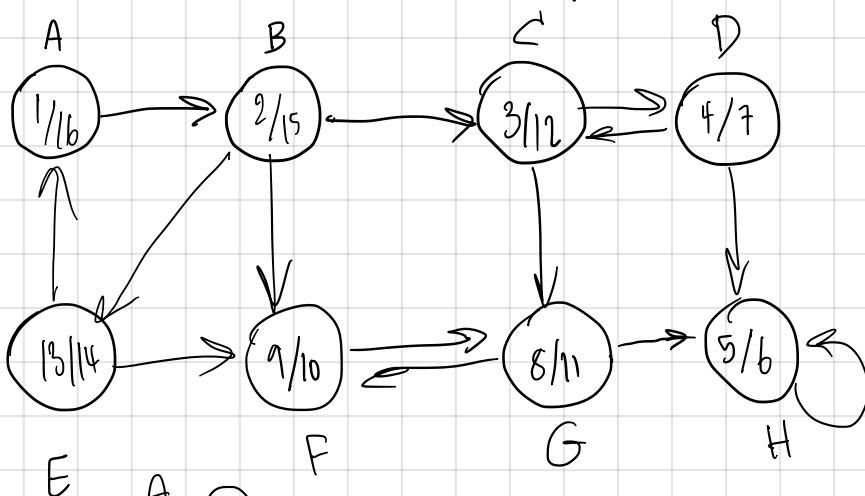
1. DFS ของ G เก็บเว็บบนลง stack

2. pop stack ของ DFS ของ  $G^T$

DF ครั้งหนึ่ง  $O(v+e)$

ทุกโนดที่ reachable จะบันทึก เติบโต

100000 node 200000 รั้งดู



รอบปรีวาร์

ที่อยู่ในตู้

push ตัวผู้ชนะ

เดินทางมา

backtrack

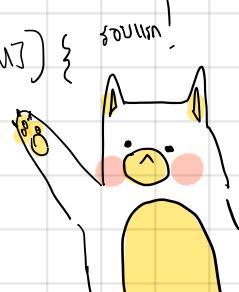
GET!

A
B
C
G
F
D
H

ใช้ DFS ชี้กลับใน  $G^T$   
หานode ที่มากที่สุด  
SCC เติบโต

```
dfs(u)
{
    if(mark[u])
        return;
    mark[u] = 1;
    for (auto x:g[u]) {
        dfs(x);
    }
    S.push(u);
}
```

กระบวนการ



# Tarjan's algorithm

(SCC) DFS 1 รอบ

หน้า

- index = ชั้น node นั้น
- lowlink = root ของ node นั้น

vertex / non stack = DFS เสร็จแล้ว / ยังไม่ DFS  
 \ in stack = กำลัง DFS อยู่

① คำนวณ DFS จาก ไป กลับ

- คำนวณ ไม่ทึบไว้

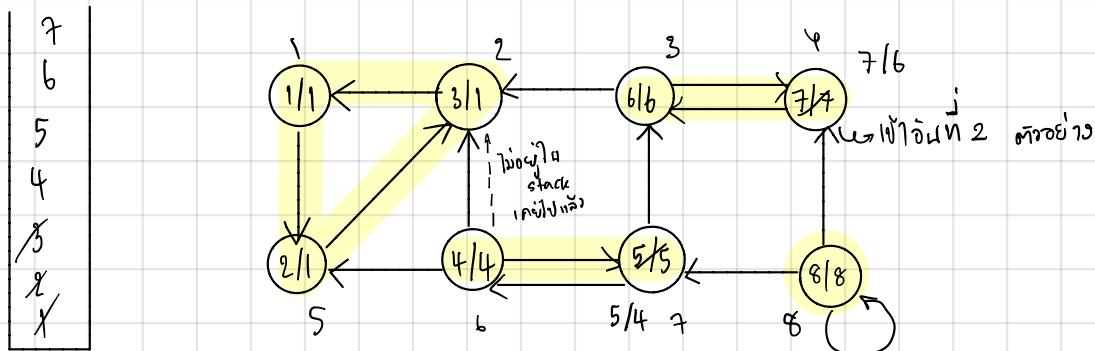
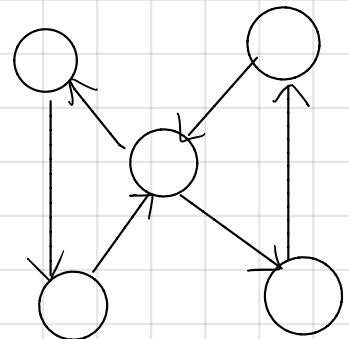
DFS(v)

เรียกคืน lowlink ลง

$\text{lowlink}[u] = \min(\text{lowlink}[u], \text{lowlink}[v]);$

- คำนวณ อยู่ใน stack จึงต้อง index ขึ้น

$\text{lowlink}[u] = \min(\text{lowlink}[u], \text{index}[v]);$



② คำนวณ  $\text{lowlink}[u] = \text{index}[u] \rightarrow \text{root SCC}$

pop ทุกโนดใน stack ที่อยู่ใน SCC โดยรักษา

จำนวน n

