

I P S T

Pongsaphol Pongsawakul

ID : win11905

► Topic

- Number Theory ✓
- Game Theory ✓
- Amortized Analysis ✓
- Persistent Data Structure ✓
- Coding Theory ✓
- NP - Completeness ✓
- Advance Data Structure ✓
- Graph Theory ✓
- Fixed Parameter Algorithm ✓
- Linear Algebra ✓
- Computational Geometry ✗
- String Algorithm ✗
- Maxflow ✗
- Evolutionary Computing ✗

Number Theory

การหารส่วนตัว

$a \in \mathbb{Z}$ หารด้วย $y \in \mathbb{Z}$ ลงตัว $\Leftrightarrow \exists k \in \mathbb{Z}, a = ky$ ผ่าน商เท่ากับ y/a

เลขคู่ $2 | a \Leftrightarrow \exists k, a = 2k$

เลขคี่ $2 \nmid a \Leftrightarrow \exists k, a = 2k+1$

$\forall n, x \in \mathbb{Z}$: ลักษณะทั่วไป $n = a_0 + a_1x + a_2x^2 + \dots + a_kx^k$ โดยมี a_i ($i < k$)

และ ลักษณะทั่วไป a_i เป็นแบบเดียวกัน ถ้า $0 \leq a_i < n$

Prime factorization

$\forall n \in \mathbb{Z}$ ลักษณะของตัวประกอบของ n เป็นจำนวนเฉพาะไม่ซ้ำเดิม

$$n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k} \quad \forall p_i \text{ prime}, p_1 < p_2 < \dots < p_k$$

Greatest Common Divisor

$d = \gcd(a, b) \Leftrightarrow d$ ผ่านจำนวนเต็มบวกที่มากที่สุดที่

หาร a และ b

$$\begin{aligned} a &= p_1^{a_1} p_2^{a_2} \dots p_k^{a_k} \\ b &= p_1^{b_1} p_2^{b_2} \dots p_n^{b_n} \end{aligned}$$

$$(a, b) = \prod_i p_i^{\min(a_i, b_i)}$$

$$\Leftrightarrow d | a \wedge d | b$$

Lowest Common Multiple

$d = LCM(a, b) \Leftrightarrow d$ เป็นจำนวนเต็มบวกที่น้อยที่สุดที่ $a|d \wedge b|d$

$$d = [a, b]$$

ด้วย

$$a = p_1^{a_1} p_2^{a_2} \cdots p_k^{a_k}$$

$$b = p_1^{b_1} p_2^{b_2} \cdots p_k^{b_k}$$

$$[a, b] = \prod_i p_i^{\max(a_i, b_i)}$$

$$\text{และ } y = \gcd \cdot lcm$$

$$(a, b) = (a \% b, b)$$

สมการ Diophantine / Linear Combination

$$ax + by = c$$

ผลลัพธ์ $p = x^2 + y^2 \Leftrightarrow p \equiv 1 \pmod{4}$; p prime, $x, y \in \mathbb{Z}$

จาก $d|a \wedge d|b \Leftrightarrow d|ax+by$

\therefore ถ้า $ax+by=c$ $d|c$ ถ้า $d|a \wedge d|b$

\therefore สมการมีค่าตอบเมื่อมี d ที่ $d|a \wedge d|b \wedge d|c$

วิธี $b = aq + r$

$$a = r_1 q_2 + r_2$$

\vdots

$$r_{n-2} = r_{n-1} q_n + r_n$$

$$r_{n-1} = r_n q_{n+1} + 0$$



โดย r_n หารด้วย a, b (r_n เป็น 0 หรือ a, b)

Extended GCD

ຈະເປີຍນ Linear Combination ວອ 12, 34

$$34 = 12(2) + 10$$

$$12 = 10(1) + 2$$

$$10 = 2(5) + 0$$

$$z = 12 - 10(1)$$

$$= 12 - (34 - 12(2))(1)$$

$$z = 12(3) - 34(1)$$

↪ ແຕ່ໄປແຜ່ນສູງກວ່າ Diophantine ໂດຍ

$$p \mid ab \wedge p \nmid a \rightarrow p \mid b$$

ພິຈາລະນີ ຈາກ $p \nmid a \therefore (p, a) = 1 \therefore p \nmid ax + by$

$$\cdot pbx + aby = b$$

$$pbx + (pk)y = b$$

$$p(bx + ky) = b$$

$$\therefore p \mid b$$

Congruence

$$a \equiv b \pmod{m}$$

ເນື້ອ

- $a \equiv b \pmod{m}$ ເນື້ອເຫັນວ່າ a ແລະ b ດີເລີ່ມຕົ້ນຈາກກາງທາງຄ້ວຍ m
- $m \mid a - b$
- $\exists k \in \mathbb{Z} \quad b = a + mk$

* $a \equiv b \pmod{m}$

$$\rightarrow ax \equiv bx \pmod{m}, x \in \mathbb{Z}$$

$$\rightarrow a+k \equiv b+k \pmod{m}, k \in \mathbb{Z}$$

$$\rightarrow a^n \equiv b^n \pmod{m}, n \in \mathbb{Z}^+$$

* $ak \equiv bk \pmod{mk}$

$$\rightarrow a \equiv b \pmod{m}$$

* $ak \equiv bk \pmod{m} \quad \text{ถ้า } (m, k) = 1$

$$\rightarrow a \equiv b \pmod{m}$$

"逆" หมายความว่า Inverse การคูณสำหรับ congruence modulo m คือ

$$a \cdot a^{-1} \equiv 1 \pmod{m}$$

เช่น $3 \times 3^{-1} \equiv 1 \pmod{5}$

ท.g. $(3, 5) = 1 \therefore 3x + 5y = 1$

$$3x \equiv 1 \pmod{5}$$

$$x \equiv 3^{-1} \pmod{5} \quad x \text{ เป็นอินเวอร์สการคูณ}$$

ตั้งงี้จะวัด Inverse การคูณสำหรับ congruence modulo m ก็ต่อเมื่อ $(a, m) = 1$

โจทย์ Congruence

$$5n+12 \equiv 0 \pmod{9}$$

$$5n \equiv -12 \pmod{9}$$

$$5n \equiv 6 \pmod{9}$$

$$n \equiv 6 \cdot 5^{-1} \pmod{9}$$

$$n \equiv 6 \cdot 2 \pmod{9}$$

$$n \equiv 3 \pmod{9}$$

$$\therefore n = 9k+3, k \in \mathbb{Z}$$

ຮະບັນດາການ congruence

$$2x+1 \equiv 5 \pmod{7}$$

$$3x+7 \equiv 6 \pmod{5}$$

ເວົ້າ

$$x \equiv 1 \pmod{2}$$

$$x \equiv 2 \pmod{3}$$

$$x \equiv 4 \pmod{5}$$

trick :
$$\begin{cases} x \equiv -1 \pmod{2} \\ x \equiv -1 \pmod{3} \\ x \equiv -1 \pmod{5} \end{cases}$$
 } ລວມ $x \equiv -1 \pmod{30}$

ເຄີຍໄວ້ Trick \rightarrow Chinese remainder theorem

Chinese Remainder Theorem

idea : $X = A + B + C$

$$X \equiv A + B + C \pmod{2} \equiv A \pmod{2}$$

$$X \equiv A + B + C \pmod{3} \equiv B \pmod{3}$$

$$X \equiv A + B + C \pmod{5} \equiv C \pmod{5}$$

$\therefore A$ ມີ 3, 5 ວິທີ ; B ມີ 2, 5 ວິທີ ; C ມີ 3, 5 ວິທີ

$$X = 3 \cdot 5 \cdot a + 2 \cdot 5 \cdot b + 2 \cdot 3 \cdot c$$

ຕ້ອງຢືນ $a \quad \tilde{n} \quad 3 \cdot 5 \cdot a \equiv 1 \pmod{2} \rightarrow a \equiv 1 \pmod{2}$

$b \quad \tilde{n} \quad 2 \cdot 5 \cdot b \equiv 2 \pmod{3} \rightarrow b \equiv 2 \pmod{3}$

$c \quad \tilde{n} \quad 2 \cdot 3 \cdot c \equiv 4 \pmod{5} \rightarrow c \equiv 4 \pmod{5}$

$$\therefore X = 3 \cdot 5 \cdot 1 + 2 \cdot 5 \cdot 2 + 2 \cdot 3 \cdot 4$$

$$= 15 + 20 + 24$$

$$= 59 \pmod{30}$$

$$x \equiv 29 \pmod{30}$$

$$2u+1 \equiv 5 \pmod{7}$$

$$3u+7 \equiv 6 \pmod{5}$$

$$\textcircled{1} \quad 2u \equiv 4 \pmod{7}$$

$$77 \cap (3,7) = 1 \quad \therefore u \equiv 2 \pmod{7}$$

$$\textcircled{2} \quad 3u \equiv -1 \pmod{5}$$

$$3u \equiv 4 \pmod{5}$$

$$77 \cap (3,5) = 1 \quad \therefore u \equiv 3 \pmod{5}$$

$$\begin{aligned} X &= A + B \\ &= 5a + 7b \end{aligned}$$

$$5a \equiv 2 \pmod{7} \rightarrow a \equiv 6 \pmod{7}$$

$$7b \equiv 3 \pmod{5} \rightarrow b \equiv 4 \pmod{5}$$

$$\therefore X = 5 \cdot 6 + 7 \cdot 4$$

$$= 30 + 28$$

$$= 58$$

$$X \equiv 23 \pmod{35}$$

$$\textcolor{red}{*} \text{ 逆元 } \quad \frac{1}{2} + \frac{1}{3} \equiv \frac{5}{6} \pmod{7}$$

$$1 \cdot 2^{-1} + 1 \cdot 3^{-1} \equiv 5 \cdot 6^{-1} \pmod{7}$$

(7)

Fermat's Little Theorem:

$$a^{p-1} \equiv 1 \pmod{p}; p \text{ prime}, 0 < a < p$$

Proof: พิจารณาตัวทั้งหมดของ $p: 1, 2, 3, \dots, p-1$

เมื่อ a ใดๆ ใน $\{1, 2, 3, \dots, p-1\}$: $a, 2a, 3a, \dots, (p-1)a$

มีคู่ ยกเว้น ay ที่จะเป็น $0 \pmod{p}$

Proof by Contradiction

สมมุติว่า x, y ที่ $0 < x, y < p, x \neq y$ ที่ $ax \equiv ay \pmod{p}$

$\therefore (a, p) = 1 \therefore a \equiv y \pmod{p}$ ต่อไปนี้ $x \neq y$

เห็นได้ว่า $a = i$ ใดๆ ก็ได้กับ $i \in \{1, 2, 3, \dots, p-1\}$

ผลคูณของทั้งหมด \equiv ผลคูณของทั้งหมด \pmod{p}

$1 \times 2 \times 3 \times \dots \times (p-1) \equiv a \times 2a \times 3a \times \dots \times (p-1)a \pmod{p}$

$(p-1)! \equiv a^{p-1} (p-1)! \pmod{p}$

จาก p prime: $((p-1)!, p) = 1$

$\therefore a^{p-1} \equiv 1 \pmod{p}$

$$x^2 \equiv 1 \pmod{p} \rightarrow x \equiv \pm 1 \pmod{p}; (x+1)(x-1) \equiv 0 \pmod{p}$$

$$\begin{array}{l} (\pmod{7}) \\ \begin{array}{ccccccc} a & = & 1 & 2 & 3 & 4 & 5 & 6 \end{array} \end{array}$$

$$\begin{array}{ccccccc} a^{-1} & = & 1 & 4 & 5 & 2 & 3 & 6 \end{array}$$

- จำนวน x ที่ inverse ต้องเป็นคู่ ($\forall x, y \quad ax \neq ay \equiv 1 \pmod{p}$)

- จำนวน x ที่ inverse ไม่เป็นคู่: 1 จำนวน คู่ $1 \pmod{p}$

$(p-1)$ จำนวน คู่ $-1 \pmod{p}$

- จำนวน x ที่ inverse คู่ จำนวน $[2, p-2]$ จำนวน

$$\therefore (p-1)! \equiv 1 \times 2 \times 3 \times \dots \times (p-2) \times (p-1)$$

$$\equiv 1 \times 1 \times -1 \pmod{p}$$

$$\equiv -1 \pmod{p}$$

$$\text{ดังนั้น } a \times a^{-1} \equiv 1 \pmod{p}$$

Primality Testing

- ไอล์ ზาร์ ตั้งแต่ 2 ถึง $n-1$
- ไอล์ ზาร์ ตั้งแต่ 2 ถึง \sqrt{n}
- ไอล์ ზาร์ prime ตั้งแต่ 2 ถึง \sqrt{n}
 - ไม่ต้องการ prime : ไอล์ หาง $6n+1, 6n-1$ (คุณ prime ก็อยู่ทุกตัว)

- ทดสอบ Pseudo prime :

$$\text{ทดสอบ } 2^{n-1} \equiv 1 \pmod{n}$$

ถ้าหาก ไม่ใช่ prime จะให้ออก

ถ้าหาก ไม่ใช่ prime แต่เข้าจดบัญชี \rightarrow นี่ set ของ pseudo prime

$$2^{340} \equiv 1 \pmod{341}$$

- Carmichael Number

$$a^{n-1} \equiv 1 \pmod{n}$$

สำหรับทุกค่า a ที่ $(a, n) = 1$ และ n ไม่ใช่ prime เช่น 561

- Miller-Rabin primality test

ถ้า n เป็น prime $n-1 = 2^r \cdot d$ (d ไม่ลงตัว)

n นำลงบัญชี prime ถ้า

- $a^d \equiv 1 \pmod{n}$ จะถือ

- $a^{2kd} \equiv -1 \pmod{n}$ สำหรับ $0 \leq k \leq r-1$ $(a, n) = 1$

เช่น 221 เป็น prime ไหม?

$$221-1 = 220 = 2^2 \cdot 55 \quad (d=55, r=2)$$

random ค่า a ($1 < a < n-1$) ล้วงค่า $a=174$

$$a^{2^0 d} \equiv 47 \pmod{221}$$

$$a^{2^1 d} \equiv -1 \pmod{221}$$

$\therefore 221$ อาจเป็น prime (มีตัวชี้ว่า ไม่เป็น)

ถ้าเปลี่ยนค่า a อาจเห็นว่า $221 \neq \text{prime}$

ถ้า a : strong pseudo prime (สามารถเชื่อ prime ได้มาก)

การเข้ารหัส public key / private key

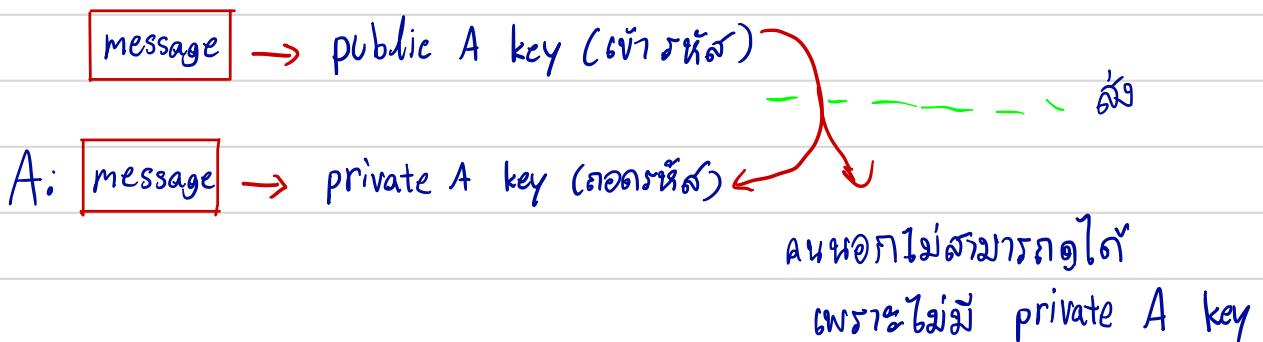
- Caesar encoding : ASCII + "3" เช่น "Hello" → "Khoor"

* คุณจะต้อง / ปลายทางที่มีคนรับ : Symmetric key เช่น → ตัวอักษรจะเป็น เช่น

$$A \leftrightarrow B \quad B \leftrightarrow C \quad C \leftrightarrow A$$

* Asymmetric key: คุณจะไม่รู้ว่าคนส่งมาอย่างไร

- Private key : A (user) จะไม่รู้
- Public key : A (user) จะรู้ทุกคน



หลักการทำงาน



$$(M^e)^d \equiv M \pmod{n}$$

private public

คณิตศาสตร์ : ถ้า d, n และ a, b 使得 $a \equiv b \pmod{n}$

RSA Algorithm

$$\text{ສາມ } M^{p-1} \equiv 1 \pmod{p}; p \text{ prime}$$

$$M^{q-1} \equiv 1 \pmod{q}; q \text{ prime}$$

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

ស្ថិតលេខា prime p, q ទាំងបីនាក់ (128 បិត)

$$\text{ហើយ } n = pq$$

$$a^p \equiv a \pmod{p}$$

$$a^q \equiv a \pmod{q}$$

ផ្លូវការ e នឹង 1 តាតា ($1 < e < p$) $\text{ និង } (e, (p-1)(q-1)) = 1$

↳ ភពិជ្ជករ key

$$\text{និង inverse នៃ } e: d \equiv e^{-1} \pmod{(p-1)(q-1)}$$

$$ed \equiv 1 \pmod{(p-1)(q-1)}$$

$$\begin{aligned} \therefore M^{ed} &\equiv M^{(p-1)(q-1)k+1} \pmod{n} \\ &\equiv M^{(p-1)(q-1)k} \cdot M \pmod{n} \\ &\equiv 1^k \cdot M \pmod{n} \\ &\equiv M \pmod{n} \end{aligned}$$

Trick ការសម្រួលចិត្តរបៀបរាយលុយសម្រាប់ $n = pq$ ដែលមានការណា $(O(\sqrt{n}))$

បន្ទាត់ថា p, q នឹងជា prime នូវវិធី ដើរការ $(O(\log n))$

Euler Totient Function

$\phi(n)$ = จำนวนของทั้งหมดที่ไม่หารด้วย n ที่ ระหว่าง 1 กับ n เมื่อ $n \neq 1$

$$\phi(2) = 1 : 1$$

$$\phi(3) = 2 : 1, 2$$

$$\phi(4) = 2 : 1, 3$$

$$\phi(5) = 4 : 1, 2, 3, 4$$

$$\phi(6) = 2 : 1, 5$$

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

$\phi(p) = p-1$; p prime หมายความว่า p คือตัว p ที่หารลงตัว

$\phi(p^k) \rightarrow$ จำนวนที่หารลงตัว p ลงตัว

\therefore หาก $p \cdot x < p^k \rightarrow (p \cdot x, p^k) \neq 1 \rightarrow$ มีทุกๆ p เท่า

$$\phi(p^k) = p^k - p^{k-1} = p^k \left(1 - \frac{1}{p}\right) \quad \phi(p^k \cdot q^\ell) = \phi(p^k) \phi(q^\ell)$$

$$\phi(pq) = \phi(p)\phi(q) = (p-1)(q-1)$$

$$n = p_1^{a_1} p_2^{a_2} \dots p_k^{a_k}$$

$$\phi(n) = \phi(p_1^{a_1}) \phi(p_2^{a_2}) \dots \phi(p_k^{a_k})$$

$$= n \left(1 - \frac{1}{p_1}\right) \left(1 - \frac{1}{p_2}\right) \dots \left(1 - \frac{1}{p_k}\right)$$

$$= n \prod_{i=1}^k \left(1 - \frac{1}{p_i}\right)$$

$$\forall (a, n) = 1, 1 < a < n$$

$$a^{\phi(n)} \equiv 1 \pmod{n}$$

proof: ពិការសារតាមនៅទំនួរ. ដើម្បី 1 និង $\phi(n)$ ត្រូវបាន $\phi(n)$ ចំណាំ

$$\begin{matrix} 6\text{ជំនួយ} & 1, 5, 7, 11 \\ & \downarrow \downarrow \downarrow \downarrow \end{matrix} \quad (n=12)$$

$$1 \cdot a \ 5 \cdot a \ 7 \cdot a \ 11 \cdot a$$

ឱ្យមើល $a \cdot i = a_j$ ដូច $i=j$ (ឯងចំណាំ) \rightarrow គត់ទូ Fermat

$$\rightarrow \text{ទូរសារអនុវត្ត: } a^{\phi(n)} \equiv 1 \pmod{n}$$

$$\sum_{i|n} \phi(i) = n$$

$$\text{សម្រាប់ } n=12$$

1	2	3	4	5	6	7	8	9	10	11	12
$\frac{1}{12}$	$\frac{2}{12}$	$\frac{3}{12}$	$\frac{4}{12}$	$\frac{5}{12}$	$\frac{6}{12}$	$\frac{7}{12}$	$\frac{8}{12}$	$\frac{9}{12}$	$\frac{10}{12}$	$\frac{11}{12}$	$\frac{12}{12}$
$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{4}$	$\frac{1}{3}$	$\frac{5}{12}$	$\frac{1}{2}$	$\frac{7}{12}$	$\frac{2}{3}$	$\frac{3}{4}$	$\frac{5}{6}$	$\frac{11}{12}$	$\frac{1}{1}$

នឹងចាកចេញថា $\phi(n)$ ត្រូវបានចំណាំ

$$(a, 12) = 1: \frac{1}{12}, \frac{5}{12}, \frac{7}{12}, \frac{11}{12}$$

$$(a, 6) = 1: \frac{1}{6}, \frac{5}{6}$$

$$(a, 4) = 1: \frac{1}{4}, \frac{3}{4}$$

$$(a, 3) = 1: \frac{1}{3}, \frac{2}{3}$$

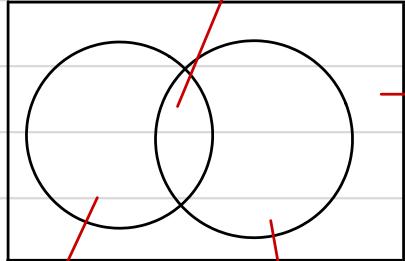
$$(a, 2) = 1: \frac{1}{2}$$

$$(a, 1) = 1: \frac{1}{1}$$

$$\therefore \sum_{i|n} \phi(i) = n$$

$$\phi(pq) = \phi(p)\phi(q)$$

ឧបរាយ PQ :



ឧបរាយ p និង q មិនមែន $= pq - p - q + 1$ ទេ

$$= (p-1)(q-1)$$

$$= \phi(p)\phi(q) \star$$

8/1/19

Game Theory

* Nim: เกมเด่น 2 คน: พลังกันช่วย

การเคลื่อน: หัวใจใน ภาคของ 1-3 ไป

ชนะ เมื่อคนนั้น ใช้อินส์ติทีบ

ผู้เล่น 10 อัน

|| ||| ||||

(1-3 ไป)

A: กำลังปั่นก่อน = ชนะ

(นับ 2 \rightarrow B ห้าม i A ห้าม 4-i)

10 อัน

(2 หลัง 3 ไป)

||||| |||||

> ร่วง 4 อัน

A: ห้าม 3 อัน

B: ห้าม 2 อัน - ชนะ

* สถานะ ไม่มีขึ้นกับผู้เล่น: (Impartial Games)

ตรวจสอบ: เกมที่ขึ้นกับผู้เล่น

การแข่งขัน: ไม่ทราบ

	Impartial Games	Partisan Games
Perfect Information	Nim	Checkers, Chess, Go
Imperfect Information		Card Games, poker

วิธีคิด: จำแนกอยู่

0 1 2 3 4 5 6 7 8 9 10

ผลตอบ / ร่องรอย ผลพ ผลพ ชนะ ชนะ ชนะ ผลพ ผลพ ผลพ ผลพ ผลพ ผลพ

วิธีสร้าง: Dynamic Programming



Nims : มีมีช่วยกัน หรือบวกกันได้ใน 1 ก้อน หรือบดลสุ่มห้าบวนะ

(3) |||

(5) |||||

(7) ||||||

นั่น (1) |

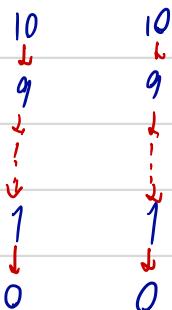
(2) || \rightarrow ให้บ 2 อันจากลง 2 \rightarrow ชนะ

(1) |

• มี 2 ก้อน

- ถ้าก้อนหนาต่อกัน : ชนะ (ชนะแล้ว)

วิธี: เล่นตามๆ



- ถ้าไม่ต่อกัน ถ้าหัวทั้งสองต่อกัน

• มี 3 ก้อน

- เก็บ state 3 มิติ :

(0,0,0)

:

(a, b, c)

► (3, 5, 7)

= (11, 101, 111) จะนับ 2

11

101

111

(XOR) 011

$$\blacktriangleright (3, 5, 6) = (11, 101, 110) \text{ ฐาน } 2$$

$$\begin{array}{r} 11 \\ 101 \\ 110 \\ \hline (\text{XOR}) \quad \underline{000} \end{array}$$

ถ้าผล XOR = 0 นั้นก็แสดง 朋รำ ราชะ ลั่นตามเท่าไปแล้วเสมอ ตัวเลขลั่นจะไม่

ราชะส่องกรุ ท่ำงหัน 90 องศา 90 องศา (朋รำ ราชะ ลั่น 1 90 องศา หันเป็น逆)

ตัวอย่าง $\begin{array}{c} 11 \\ \rightarrow \text{เทา หิบอันนี้} \end{array}$

$\begin{array}{c} 101 \\ \leftarrow \text{บากชันอันนี้} \end{array}$ $\rightarrow \text{บากชันอันนี้}$

$\begin{array}{c} 110 \\ \leftarrow \text{ราชะ หิบอันนี้} \end{array}$

โดยตัว การหิบปแบบที่ไม่ใช่ 1 ใน representation ฐาน 2

$$\begin{array}{r} 11 \\ 101 \quad \rightarrow 011 \quad (\text{เทาหิบก้องลงไป } 2) \\ 110 \quad 110 \end{array}$$

* เราก็ maintain XOR ให้ได้ 0: หิบ 2 จากกลุ่ม 3 \rightarrow ได้เหมือนกัน

แต่ถ้าตอนนี้มีตัว XOR ไม่เท่ากับ 0 ล่ะ?

$$\text{เช่น } (3, 5, 7) = \begin{array}{r} 11 \\ 101 \\ 111 \\ \hline \underline{001} \end{array}$$

ถ้าเป็นตามเราก่อน: ทำยังไงก็ได้ให้ XOR เป็น 0:

$$\begin{array}{r} 11 \\ 101 \\ 110 \quad \rightarrow \text{หิบปอก } 1 \\ 000 \end{array}$$

ถ้าเป็นตามเราต่อ: ก็ทำยังไงก็ได้ให้ XOR เป็น 0:

กรณียากหน่อย: เหลือ $(3, 5, 1)$

$$\begin{array}{r} 11 \\ 101 \\ 001 \\ \hline 111 \end{array}$$

\rightarrow เราหายิบ 7 ให้ได้แต่ งานสามารถเปลี่ยน XOR sum
เป็น 0 ได้โดย

$$\begin{array}{r} 101 \\ 111 \\ \hline 010 \\ 001 \\ \hline 000 \end{array}$$

\rightarrow หายิบ 3 ผลกماจาก กองที่ส่วน (ที่หักเหลือ 2) เพื่อจะได้

เกณฑ์จะชนะไปแล้ว (นอกจგกว่า XOR SUM เป็น 0 อย่างแล้ว เราจะไม่มีข้อจำกัดอีกต่อไป)

เงื่อนไข: $x_1 \oplus x_2 \oplus x_3 \oplus \dots \oplus x_n = S$ (XOR SUM)

ผลลัพธ์: $y_1 \oplus y_2 \oplus y_3 \oplus \dots \oplus y_n = T$ (XOR SUM)

โดย $\forall i \neq k, x_i = y_i$ และ $k \in [1, n]$ (เปลี่ยนค่าได้แต่ครั้งเดียว)

พิจารณา $T = T \xrightarrow{^0}$
 $T = (S \oplus S) \oplus T$
 $= S \oplus (x_1 \oplus x_2 \oplus \dots \oplus x_n) \oplus (y_1 \oplus y_2 \oplus \dots \oplus y_n)$
 $= S \oplus (x_k \oplus y_k)$

- \rightarrow กรณี $S > T$: ทำได้แล้ว
 \rightarrow กรณี $S < T$: ก็ทำได้เช่นกัน

$$S \overline{111}, T \overline{101} \therefore x_k \oplus y_k = 010$$

ตัวอย่าง x_k ที่มากกว่า y_k ที่มีมูลค่าน้อย

★ Polar Nim

- คณิตศาสตร์ Nim แต่ตัวเลขที่บัญชีมีตัวเลข ฐานสอง และ เก้าอี้ภาษาดำเนินมาจากการ
คลังมาเติมกลับในกระดาษเคมีได้

Solution: เก้าอี้น้ำหนัก 7 ตัว ก็ ให้ข้อมูลห้ามหัน ที่เหลือ ก็ แล้ว เชื่อว่าเดิม

★ Northcott กรณี 8×8 มีเบื้องหลัง(A) ดังนี้

- คณิตศาสตร์: สัญลักษณ์และ แนวโน้มทางของตัวเลข ที่จะตรวจสอบหันนั้น ถ้าเดินไปได้ นั่นเอง

		●				●	
	●			●			
			●	●			
		●				●	
						●	●
	●				●		
●			●				
●					●		

≡ Nim แบบ ซ้อนรากฐานสอง \rightarrow จำนวนไม้ที่นกจะ

(Polar Nim)

$\rightarrow (3, 2, 0, 3, 0, 3, 2, 4)$

★ Nimble: มีช่วง n ช่อง, มีหนึ่งรากฐาน k แห่งรากฐาน

ผู้คนค่อนข้าง ชี้ว่าเป็นไปตามที่อยู่ ตำแหน่งที่ 2 ได้ หัวใจรักษาไว้ หัวใจรักษาไว้ หัวใจรักษาไว้

หัวใจรักษาไว้ หัวใจรักษาไว้

≡ Nim แบบ: ตัวเลขทั้งหมดนับ 2 \rightarrow จำนวนไม้ที่นกจะ

★ Nim แบบที่บัญชีคณิตศาสตร์ห้ามหันนั้น

- Fine tune แต่ห้ามหันนั้น

★ Nim Advanced

(2,3) Nim |||

(1,3) Nim ||||

(1,2,3) Nim ||||| ||

Grundy Number: ចំណាំលេខការដែលត្រូវបានដឹងដើម្បីដឹងថាការងារនេះមានតម្លៃដែរ

$$\text{ដែល} \left\{ \begin{array}{l} g(u) = \min \{ n \geq 0 : n \neq g(y) \text{ for } y \in f(u) \} \\ g(u) = \max \{ g(y) : y \in f(u) \} \end{array} \right.$$

ចំណាំលេខការ (2, 3)	Nim =	0	1	2	3	4	5	6	7	8	9	10
		L	L	W	W	W	L	L	W	W	W	L
	$g(u) =$	0	0	1	1	2	0	0	1	1	2	0

(1, 2, 3)	Nim =	0	1	2	3	4	5	6	7	8	9	10
		0	1	2	3	0	1	2	3	0	1	2

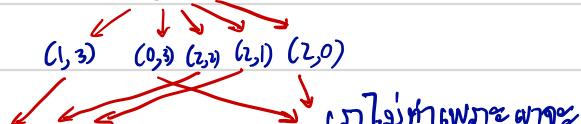
∴ បាន 10 < (2, 3) Nim >, 10 < (1, 2, 3) Nim >

តាត់ status នៀន $(10, 10) \xrightarrow{(2, 3) \text{ Nim}, (1, 2, 3) \text{ Nim}} g(u) = (0, 2)$

ហើយ ឯណានៅក្បែង (0, 0) : នៅនៅ (0, 8)

នៅលើ ពិភោះការឱ្យធ្វើ ធម៌សង្គមនៅ នៅ (0, 0) → ស្ថាកម្មនៃរាយការសង្គមនៅ

តាត់ status នៀន (9, 7) $\rightarrow g(u) = (2, 3)$



នៅនៅ ឯណានៅក្បែង

បាន នៅ (0, 0) ។

* លេខកុំពួក

ដឹងទីលូក និង លូក

Goal: ដលើកដឹងទីលូក និង លូក ទៅការងារអាមេរិក

គឺជាសម្រាប់ការងារ

ការរំលែក

1. ត្រូវការងារ 1 ដឹងទីលូក និង លូកដឹងពី T

2. (optimal) និងការងារ 1 ដឹងទីលូក និង លូក (H \rightarrow T និង T \rightarrow H)

ទូទាត់នូវការងារ

Sprague-Grundy Number

លេងបានដែលចិត្តអាមេរិកថា មាត្រាការណ៍នៃការលេងស្ថិតិសាស្ត្រភាពនៅពេល កំណត់ស្ថិតិសាស្ត្រភាព 0 (សារុទ្ធសាស្ត្រភាព 0 = ឯកសារ)

$$G(u) = g(u_1) \oplus g(u_2) \oplus \dots \oplus g(u_n)$$

ឱ្យ 4, 4, 4 $(1, 2, 3)$ Nim $\rightarrow g(0) = (0, 0, 0)$ $G(u) = 0$

ឱ្យ 3, 5, 7 $(1, 2, 3)$ Nim $\rightarrow g(0) = (3, 1, 3)$ $G(u) = 1$

គឺជា Nims Advanced នូវ g(u) ដែលបាន រួម Nims

កំណត់លើកដោយ \rightarrow Nims

$$\begin{matrix} \text{HTHTHTHT} \\ | \quad 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \end{matrix} \rightarrow (1, 3, 4, 6, 7)$$

1.) HTHTHTHTT $\rightarrow (1, 3, 4, 6, 0)$

2.) HTTHHTHTTT $\rightarrow (1, 0, 4, 6, 0) \equiv (1, 3, 4, 6, 3)$ ពេលវេលា $3 \oplus 3 = 0$

3.) HHHTHTHTTT $\rightarrow (1, 3, 4, 6, 2)$

Game + State-space Search

• Branch-and-Bound

• Extended-Lists តាមលទ្ធផល state នៃលក្ខណៈ cost ដីលក្ខណៈការងារការណ៍លើ
តាមលទ្ធផល state តិច នៃ cost នាក់ការងារ

• Heuristic / Goal Estimation

* រួមចិត្តថា ឯកសារស្ថិតិសាស្ត្រភាព និងការងារ

• A*

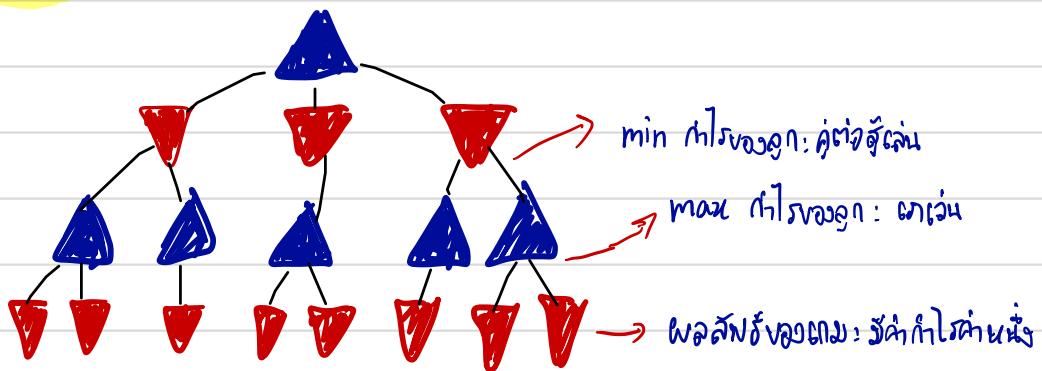
• heuristic function:

1.) $h(u, G) \leq d(u, G)$

2.) $|h(u, G) - h(y, G)| \leq d(u, y)$

= Branch-and-bound + extended-list + heuristic \rightarrow ប៉ូតាតិកិច្ច

Minimax Tree

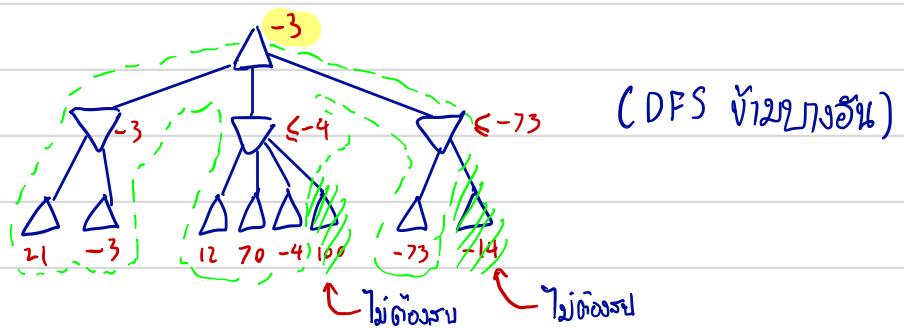


Time complexity $O(CL^n)$

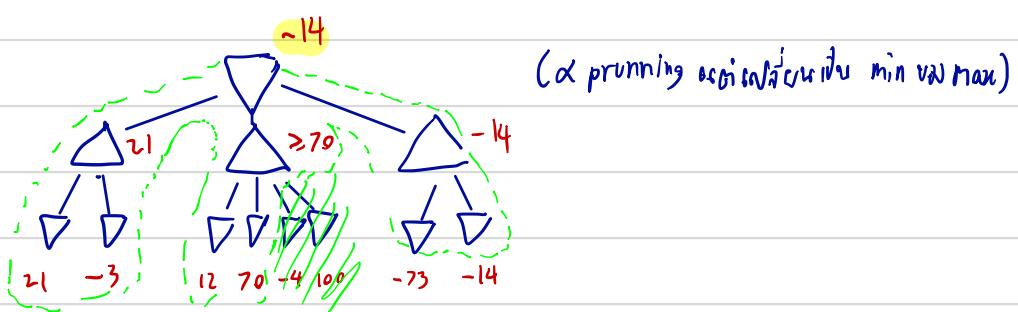
Space complexity $O(CL^m)$

Pruning Minimax Tree

α Prunning



β Prunning



Silver Coin Game

มังกรรับสุ่มอยู่ k อัน (ไม่ต่อ กัน)

- ผู้เล่น 1 สร้างรากเลื่อน เนื้อร่อง ให้เป็นร่อง หนึ่งช่องทางซึ่งก่อสร้างก็ได้ (หัวมีด/หัวเขียว หรือ สีอื่น)
- ผู้เล่น 2 ต่อไม่ได้ = แพ้

Amortized Analysis

▶ Worse Case Analysis

▶ Average Case Analysis

ตัวอย่าง: stack

- push(): O(1)

- top(): O(1)

- pop(): O(1)

ถ้าจะเพิ่ง Operation multipop(k) \rightarrow pop ออก k ตัว $\rightarrow O(n)$

★ เรียกว่าค่าสัมมติ 4 อันที่ดำเนินการต่อๆ กัน $\rightarrow O(n^2)$

- การวิเคราะห์แบบนี้ ไม่ผิด แต่เข้มงวด tight พอ

★ Aggregate Analysis: วิเคราะห์การทำงานรวมของหลาย operation ต่อๆ กัน

- ตัวรวม = $O(fcn)$

amortized cost ต่อ Operation = $O\left(\frac{fcn}{n}\right)$

ดูในเรื่อง pop ไปเมื่อกัน n ตัว $\rightarrow O(n)$

\therefore ต่อ 1 operation จะใช้เวลา $\approx O(1)$

ตัวอย่าง Binary counter

- แม่ array k bit (0/1)

- เริ่มต้นทุก 0 ทุก bit

- operation: increment()

★ worst case: $O(k)$ ต่อ operation

pseudo code:

$i=0$

while($i < k \wedge A_i = 1$) $A_i = 0;$

if($i < k$) $A_i = 1;$

★ ลบทิ้งกรณี amortized cost

000 \rightarrow 1

001 \rightarrow 2

010 \rightarrow 1

011 \rightarrow 3

100 \rightarrow 1

101 \rightarrow 2

110 \rightarrow 1

111 \rightarrow 4

$$\text{sum} = n + \binom{n}{2} \cdot 2 + \binom{n}{4} \cdot 3 + \dots + \binom{n}{2^i} \cdot (i+1)$$

$$= \frac{n}{2^{i+1}} (2^{i+1} + 2^i + \dots + 2^1)$$

$$= \frac{n}{2^{i+1}} (2^{i+2}) = 2n = O(n)$$

∴ สำหรับ increment ทุกครั้ง ต้องต่อ กัน รวม

$$= n + \frac{n}{2} + \frac{n}{4} + \dots = \sum_{i=1}^{\log n} \frac{n}{2^{i-1}}$$

$$= n \cdot 2$$

$$= n \cdot 2$$

$$= O(n) \rightarrow \text{amortized } O(1)$$

★ Accounting Method

ស្ថិតិ cost នីមួយា

push	\hat{C}_{push}
pop	\hat{C}_{pop}
multipop	$\hat{C}_{multipop}$

cost ទី២

C_{push}
C_{pop}
$C_{multipop}$

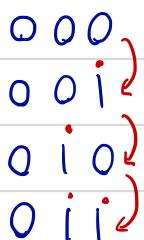
$$\text{ត្រូវពិនិត្យថា } \sum_{i=1}^n \hat{C}_i \geq \sum_{i=1}^n C_i \quad \hat{C}_i \text{ គឺជាប័ណ្ណុអាមេតិសែសាលា}$$

$$\left. \begin{array}{l} \text{push cost ជាន់ } c=1 \text{ សម្រាតិ } \hat{C}=2 \\ \text{pop cost ជាន់ } c=1 \text{ សម្រាតិ } \hat{C}=0 \\ \text{multipop cost ជាន់ } c=k \text{ សម្រាតិ } \hat{C}=0 \end{array} \right\} \text{ ពិនិត្យ } \rightarrow \text{amortized cost} = O(1)$$

លេខចូលកញ្ចប់ binary counter:

★ charge cost នៃ `increment()` $\hat{C}=2$

វិវាទិភាព



charge ដែលចូលកញ្ចប់ 1 កន្លែងពី 0

★ Potential Method

"ផលរងរាយគឺជាការបង្កើតនៃការងារ"

ឬ D ជាន់ Data Structure

$\Phi(D)$ ជាន់ Potential function

Amortized cost: $\hat{C} = c + \Delta \Phi(D)$

ត្រូវ D₀ ជាន់ data structure ទឹកចំណាំ

$$\Phi(D_i) \geq \Phi(D_0)$$

ឬ $\Phi(D_i) \geq 0$ ដើម្បីវិនិច្ឆ័យថា potential នៅពេលនៃការងារ ≥ 0

ສະແດງ stack:

ຖືກ $\Phi(D_i)$ ຂະໜານ ຈຳກັນຫອງລົງໃນ stack ອະນຸຍາວ operation ທີ່ i

$$\hookrightarrow \text{ໃຊ້ວ່າ } \Phi(D_0) = 0$$

$\Phi(D_i) \geq 0$ ແນ້ນຂອງ (stack ໄວ້າຫຍກຕ່າງໆ)

* push $\hat{C} = C + \Delta \Phi(D) = 1 + 1 = 2$ potential ຖໍ່ປັບປຸງໄປຢູ່ 1 operation

* pop $\hat{C} = C + \Delta \Phi(D) = 1 + (-1) = 0$

* multipop $\hat{C} = C + \Delta \Phi(D) = k + (-k) = 0$

\therefore ທີ່ສະນຸ operation ມີ amortized cost ເປົ້າ $O(1)$

ລະບຽບໂຕ Binary Counter

ອອກແບບ $\Phi(A)$ = ຈຳກັນຫອງໃຫຍ່ໃນ array ທີ່ເປົ້າ 1

$$\Phi(A_0) = 0$$

$\Phi(A) \geq 0$ ແນ້ນຂອງ

$$\hat{C} = C + \Phi(A)$$

$$= k + (1 - (k-1))$$

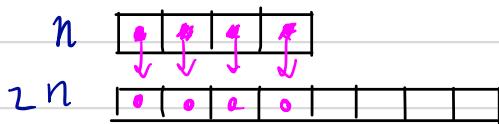
$$\begin{array}{c} \downarrow \quad \downarrow \quad \downarrow \\ \text{ລະບຽບໄດ້ປັບປຸງໃນ } k \text{ ຕົວ } 1 \text{ ຫຼື } 1 \text{ ດີນ } k-1 \text{ ດີນ } \\ = 2 \end{array}$$

* Dynamic Array (Vector)

- push_back():

ເພີ້ອຫາດ array ເຕີມ n ຊົ່ວ

ຈະຕ້ອງຫາຫ້ອງໄວ້ 2n ຊົ່ວ ແລະ duplicate ພົມຫລັກ n ຊົ່ວ



$$\hat{C} \text{ ສະເງິນ } \text{push} = 3$$

\hookrightarrow ອັກ 2 ຝຳກ່າວເພື່ອໄປຈອງ array ຫຼາຍ n ຊົ່ວ, copy n ຊົ່ວ

- pop_back(): ກີ່ເຜົາຄົດເຫັນກົດ

(ຍຸ້ອງຫາດ array ແລ້ວ ຖໍ່ມີຫຍຸ້ລູກ $\leq \frac{n}{2}$ ຊົ່ວ)

Potential method จะคิดยังไง?

$$\Phi(A) = 2 \times \underbrace{\text{จำนวนของรากที่อยู่ในตัว} - \text{size}(A)}_{n(A)}$$

ถ้า array ตัวนึง $\Phi(A) = 2 \cdot n - n = n$

ฟังฟังที่ expand มาก $\Phi(A) = 2n - 2n = 0$

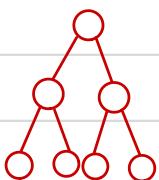
$$\hat{C} = C + \Delta \Phi(A) = 1 + 2 = 3$$

$$\hat{C} = C + \Delta \Phi(A) = (n+1) + ([2(n+1)-2n] - [2n-n]) \\ = 3$$

เพิ่มข้อมูลเดียว

↑

Amortized Analysis for BST:



- search
- insert
- delete

Lazy deletion :

- search ว่า มีใบบุ้ง? (ไม่มี: ช่างดี)
- ถ้า มี mark node ที่น่าถูก delete แล้ว

▶ หลังจากการ lazy deletion k node : จะมีการ reorganize tree

สร้าง tree ใหม่ ที่มี $n-k$ node และเป็น fully balanced

▶ Big O ของ delete ตอนหนึ่ง รวม กันเท่าไร?

▶ k คืออะไร เช่นไร?

• สร้าง perfectly balanced tree จาก tree cũ ($O(n)$)

- inorder traversal

- เดิน node ตามก่อนล่างสุดของ root

- recursive วนไป จน subtree หาย, subtree หาย

• k ควรจะเป็น ที่

• amortized cost = $O(\log n)$

$$\hat{C} = \log(n) + 1$$

Search: ใหญ่ไป $\log n$ ห่าง 1 เท่าครึ่ง

reorganize: ใช้หนึ่งชั่วโมง $\frac{n}{2}$ → สร้าง tree ใหม่ขนาด $\frac{n}{2}$

\therefore amortized cost = $O(\log n)$

lazy insertion:

- insert โภคป้อมะส่งให้ความ balance ของ tree

- rebalance ที่ต่อไปเมื่อ

$\rightarrow O(\text{size}(v))$

- ถ้ามี subtree (root เป็น v) มีความสูง $h(v) > \alpha \log(\text{size}(v))$
($\alpha = \text{constant}$ มากกว่า 1)

- re balance subtree ที่ v เป็น root

- ค่าความ $I(v) = \begin{cases} 0 & \\ |\text{size}(l_v) - \text{size}(r_v)| - 1 & \end{cases}$

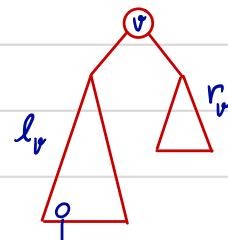
"ค่าความ imperfect"

- ถ้าหัวจะต้อง rebalance subtree ใน v ที่เป็น root:

$$\begin{aligned} \alpha \cdot \log(\text{size}(v)) &< h(v) \\ &= h(l_v) + 1 \end{aligned}$$

หากจะต้อง rebuild ที่ v

หมายความว่า l_v ถูก balanced อยู่



node ลักษณะนี้ต้อง rebalance

$$\alpha \log(\text{size}(v)) \leq \alpha \log(\text{size}(l_v)) + 1$$

$$\log(\text{size}(v)) \leq \log(\text{size}(l_v)) + \frac{1}{\alpha}$$

$$\text{size}(v) \leq \text{size}(l_v) \cdot 2^{\frac{1}{\alpha}}$$

$$\text{size}(l_v) \geq \frac{\text{size}(v)}{2^{\frac{1}{\alpha}}}$$

$$\therefore \text{size}(r_v) = \text{size}(v) - 1 - \text{size}(l_v)$$

$$\leq \text{size}(v) - 1 - \frac{\text{size}(v)}{2^{\frac{1}{\alpha}}} = \left(1 - \frac{1}{2^{\frac{1}{\alpha}}}\right) \text{size}(v) - 1$$

$$I(v) = |\text{size}(l_v) - \text{size}(r_v)| - 1$$

$$\geq \frac{\text{size}(v)}{2^{\frac{1}{\alpha}}} - \left(1 - \frac{1}{2^{\frac{1}{\alpha}}}\right) \text{size}(v)$$

insert ใหม่

ดังนั้นจะได้ว่า amortized cost $\hat{C} = \log n + 1 \rightarrow$ เท่ากับ $O(\log n)$

โดย $I(v) \in O(\text{size}(v))$ จะมีการ rebalance ที่ $O(\text{size}(v))$ (θ ของน้ำผลไม้)

$\therefore \text{amortized cost} = O(\log n)$

★ ด้วยที่เรียกว่า "Scapegoat Tree"

Splay Tree

ຈົດການທີ່ Amortized cost (ນອກຕາງ splay) $\xrightarrow{\text{"rank"}}$ ຈະພິສູນວ່າເປົ້າ $O(\log n)$

$$-\text{ຖື່ນ } r(v) = \log(\text{size}(v))$$

$$\Phi(T) = \sum_{v \in T} r(v)$$

Lemma: Amortized cost ພອມ single rotation (zig or zag) ທີ່ node x ຈະມີຄ່າໄຟເກັ້ນ

$$1 + r'(u) - r(u)$$

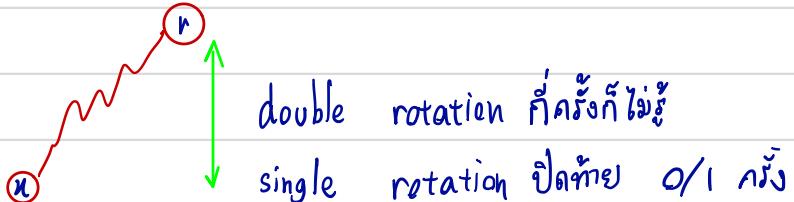
\downarrow
r(u) ນັ້ນ rotate

Double rotation:

$$3(r'(u) - r(u))$$

ຕໍ່າ lemma ເປົ້າໃຈ (ຈະພິສູນທີ່ໜັງ)

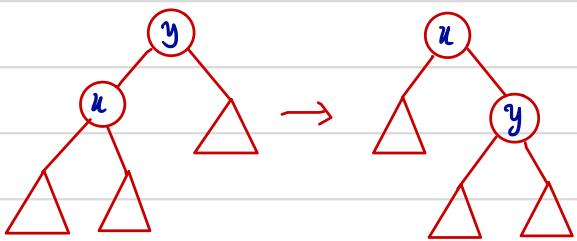
$$1 + r'(u) + r(u) \leq 1 + 3(r'(u) - r(u))$$



$$\begin{aligned}\hat{C}_{\text{splay}} &= \hat{C}_{\text{double}} + \hat{C}_{\text{single}} \xrightarrow{\log(n)} \\ &\leq 1 + 3(r'(u) - r(u)) \\ &\quad \uparrow \quad \uparrow \\ &\quad \text{ຖື່ນນັ້ນກັງ} \quad \text{ຕັ້ງແຕ່ວ່າວິວດັນ} \\ &= O(\log(n))\end{aligned}$$

► ພິຈຸນົດ lemma:

$$\begin{aligned}\hat{C}_{\text{single}} &= C_{\text{single}} + (\Phi'(T) - \Phi(T)) \\ &= 1 + (r'(u) + r'(y) - r(u) - r(y)) \quad (\text{rank node ທີ່ນັ້ນມີຄ່າ})\end{aligned}$$

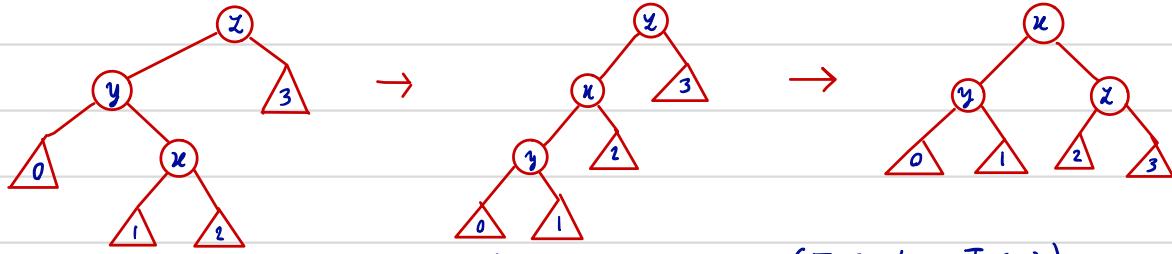


ຈາກ $\text{size}'(y) < \text{size}(y)$

$$\therefore r'(y) < r(y)$$

$$\begin{aligned}r'(y) - r(y) &< 0 \rightarrow \text{ຜົດອອກໄວ້} \\ &\leq 1 + r'(u) - r(u)\end{aligned}$$

► Zig-Zag:



$$\hat{C}_{\text{zigzag}} = C_{\text{zigzag}} + (\Phi(T') - \Phi(T)) \\ = 2 + (r'(u) + r'(y) + r'(z) - r(u) - r(y) - r(z))$$

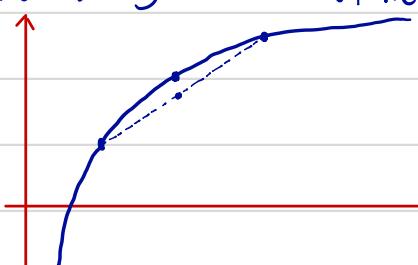
1.) $r'(u) = r(u)$

2.) $r(y) \geq r(u)$

3.) $r'(u) > r(u)$

$$\leq 2 + (r'(y) + r'(z) - 2r(u)) \\ = 2 + (r'(y) - r(u) + (r'(z) - r(u))) + 2(r'(u) - r(u)) \\ = 2 + \boxed{\log\left(\frac{\text{size}'(y)}{\text{size}'(u)}\right) + \log\left(\frac{\text{size}'(z)}{\text{size}'(u)}\right)} + 2(r'(u) - r(u))$$

* \log is not monotone

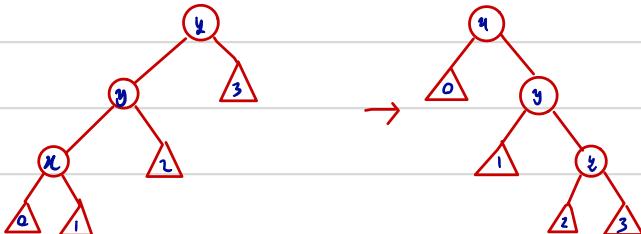


$$\frac{f(a) + f(b)}{2} \leq f\left(\frac{a+b}{2}\right) \\ f(a) + f(b) \leq 2f\left(\frac{a+b}{2}\right)$$

4.) $\text{size}'(y) + \text{size}'(z) \leq \text{size}'(u)$

$$\leq 2 + 2 \log\left(\frac{\text{size}'(u)}{2 \text{size}'(u)}\right) + 2(r'(u) - r(u)) \\ = 2(r'(u) - r(u)) \\ \leq 3(r'(u) - r(u))$$

► Zig-Zig:



1.) $r'(u) = r(u)$

2.) $r'(u) > r(u)$

3.) $r(y) \geq r(u)$

$$\hat{C}_{\text{zigzag}} = C_{\text{zigzag}} + (\Phi(T') - \Phi(T)) \\ = 2 + (r'(u) + r'(y) + r'(z) - r(u) - r(y) - r(z))$$

$$\leq 2 + (r'(y) - r'(z) - 2r(u)) \\ \dots$$

$$\leq 3(r'(u) - r(u))$$

Persistent Data Structure

► Idea. เก็บฐาน version ของ tree ให้แต่ละคนดูที่เก็บไว้ในมือ

• naive: copy ทุก tree ทุกครั้ง ($O(n)$ ครั้งครึ่งจะ $O(n)$) \Rightarrow space and time

• มีวิธีลดลงที่ลด space, time (ไม่ต้องวนcopy ทุกครั้งที่มีการเปลี่ยน)

* ตัวอย่างโดยย่อ:

- Input list of numbers

1 5 3 7 (version 0)

• operations:

- change head

- add to head

- erase head

- get max of list

1 \rightarrow 5 \rightarrow 3 \rightarrow 7 (v.0)

5 \rightarrow 5 \rightarrow 3 \rightarrow 7 (v. 1) change head to 5 from ver. 0

2 \rightarrow 1 \rightarrow 5 \rightarrow 3 \rightarrow 7 (v. 2) add 2 to head in ver. 0

5 \rightarrow 3 \rightarrow 7 (v. 3) delete head from ver. 0

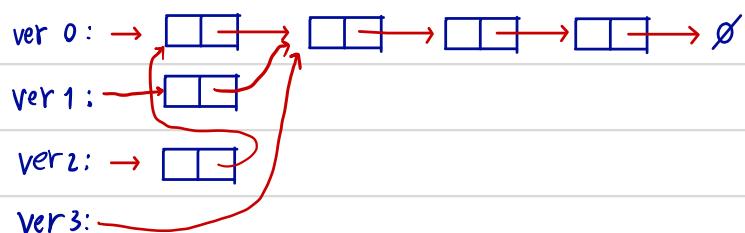
Max in list = ?

ตัวอย่างอีกแบบ

1.) Path Copying

2.) Fat Node

1.) Path Copying: ถ้าเกตเวย์ต้องบันทึก list ให้สามารถอ่านได้:



* กรณี list ว่าง: $list = null$

$= \boxed{\quad} \rightarrow list$ หมายความว่า list คือ null
 \therefore Complexity ของ Path Copying = $O(n+k)$ (Space & time)

• ห้าม maximum?

- ถ้าต้อง node เก็บ maximum ต้องเก็บอันดับด้วย

Ver 0: $5^5 \rightarrow 1^4 \rightarrow 4^4 \rightarrow 1^1$

Ver 1: 6^6

Ver 2: 9^9

Ver 3: 2^5

▶ implement

struct data {

 int v;

 int max;

 data* next; → malloc จัดที่

}

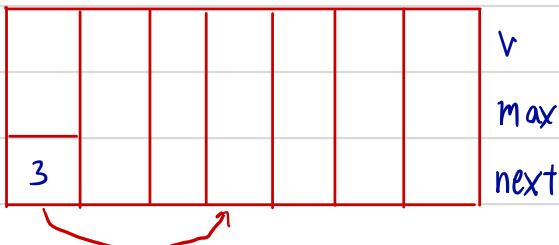
▶ free?

- shared_ptr: ถ้าไม่มีอะไรรักษาไว้แล้ว มีช่องฟรีเมมทิ้ง

ฟรีเมม: มีไฟฟ์รีบีน recursive → อาจทำให้ seg fault ได้

▶ แบบลับๆ อธิบาย implement array รองรับต่อไปนี้

0 1 2 3 4 5 6



* จงหา k th number input: Array A ($|A| \leq 10^5$), Query Q ($|Q| \leq 10^5$)

· $\text{Query}(L, K)$:

list of int: List

for value v in A :

if $v \geq L$:

$B \leftarrow B + v$

return $B[k]$

* ចំណាំ :

1. ដំឡូលបន្ទែន ក្នុង
Query $y \rightarrow$ តាមអតិថិជន
 $y \in$ អ៊ូតិភាពខ្លួនទៅកែតែ

- វិធីកំណើមបាយវិធី:

- Sort + Binary Search
- BST សំបុរាណ (AVL, Red-Black, Treap)



ដើម្បី interval n ឬ

Query : (x, y)

តាមរយៈការពិនិត្យ (x, y) ទីនេះ

- វិធីកំណើម:

- BST $\leq 2n$ ចំនួន តាមរឿងបង្ហាញ event point
- តាមចំនួន query x
- នៃ n interval បាន query y .

ផែនក្នុង persistent data structure នៃ size complexity ($O(n \log n)$)

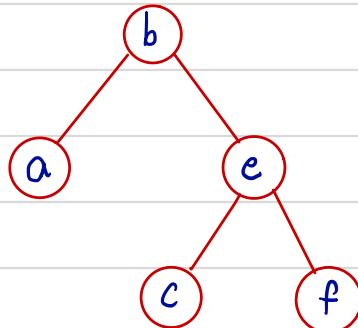
What is Treap?

- Binary Search Tree + Heap

- Example:

key	priority
'a'	3
'b'	9
'c'	2
'e'	6
'f'	5

(តម្លៃការ)



ម៉ោងមួយក្នុង: 1. priority (parent) $\geq \max\{\text{priority}(\text{left}), \text{priority}(\text{right})\}$

2. key (parent) \leq key (right)

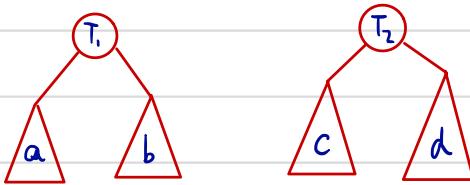
key (parent) \geq key (left)

► Claim : តាមសំណើមី priority \rightarrow ពីរ tree នឹង balanced (with high probability)

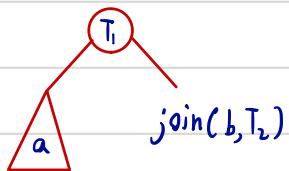
► insertion of a new node

* joining of Treaps:

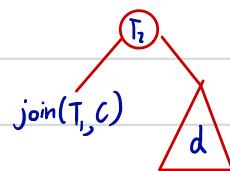
$\text{join}(T_1, T_2)$:



if $T_1 \cdot \text{priority} < T_2 \cdot \text{priority}$

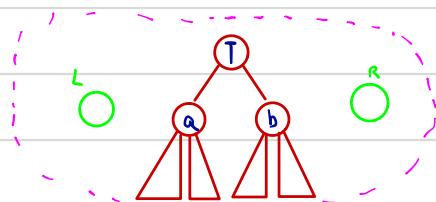


if $T_1 \cdot \text{priority} \geq T_2 \cdot \text{priority}$

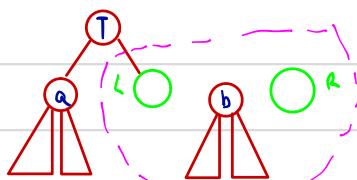


* splitting of a treap

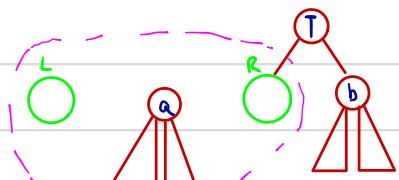
$\text{split}(T, p) \rightarrow (\text{tree } L \text{ key} < p, \text{ tree } R \text{ key} \geq p)$



if $T \cdot \text{key} < p$

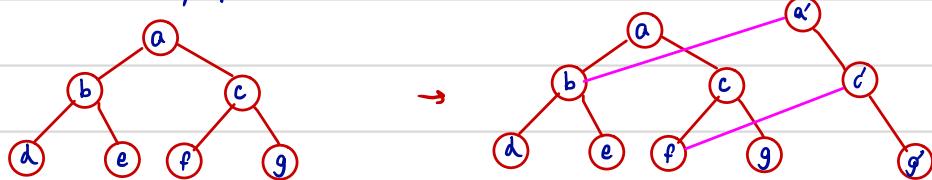


if $T \cdot \text{key} \geq p$



วิธีการ persistent ของ binary tree (path-copying)

1.) new node: copy path from root to node



2.) rotation



- Solution to k^{th} number

- นิยาม $f(i, L) = \text{จำนวนตัวเลขที่มีค่า} \geq L \text{ ที่อยู่ระหว่าง } A[i] \text{ และ } A[i+L], \text{ implement } f(i, L) \text{ ยังไง?}$

1.) segment tree

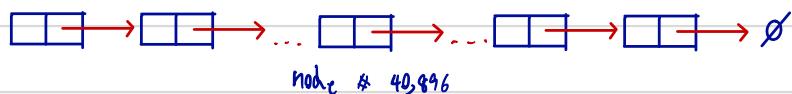
2.) สร้าง BST N ตัว วนลูป traverse L ให้ tree ต้นที่ i ($\log N$)

แต่ละ node ต้องรักษา node ให้ subtree ต้องดูแล

Persistent: $T_{i+1} = T_i . add(A[i+1])$

2. Fat node

- สร้างตัวอักษรไปลงใน array ทุกตัว ให้เวลา path-copying ต้องใช้เวลา O(1)



\therefore ผลลัพธ์: ตัวอักษร node รักษาไว้



ข้อสังเกต: เวลา traversal list ใช้เวลา $O(1)$ ต่อการเดิน 1 步

ต้องที่ ต้อง binary search ต้องหาค่าของ node ให้ + ตรวจสอบ $O(\log n)$

พื้นที่: space +1 ถ้า 1 update

Coding Theory

* Information needs to be encoded transmitted and decoded through some kind of channel(s)



- ตัวอย่าง:
 - ภาษาอังกฤษ มี ~ 10^6 คำ ~ 20 bit ต่อคำ state
 - ตารางการกับบินในรูปของตัวอักษร จะมี ~ $5 \times 8 = 40$ บิต (ASCII)

Fair Coin: โอกาสหัว/tails ทุกครั้ง 50% แต่ละครั้ง นี่คือการเก็บ

Biased Coin:

- 0 (0%) 1 (100%) \rightarrow ไม่มีช่องทางข้อมูล โฆษณาต้องมี 1
- 0 (10%) 1 (90%) \rightarrow สนใจ ที่ 90% ต้องใช้ 1 ช่องในการเก็บข้อมูล?
ไม่ใช่ที่ 10% ด้วย

นิยาม entropy:

$$H = \sum_i p(u_i) \log\left(\frac{1}{p(u_i)}\right)$$

$$\begin{aligned} H_{50/50} &= 0.5 \cdot \log\left(\frac{1}{0.5}\right) + 0.5 \cdot \log\left(\frac{1}{0.5}\right) \\ H_{90/10} &= 0.9 \cdot \log\left(\frac{1}{0.9}\right) + 0.1 \cdot \log\left(\frac{1}{0.1}\right) \end{aligned}$$

= 1 \rightarrow ผลลัพธ์ 1 bit ต่อครั้ง 1 ครั้ง
 = 0.47 \rightarrow ผลลัพธ์ 0.47 bit ต่อครั้ง 1 ครั้ง

เก็บยังไง? \rightarrow เก็บต่อๆ กันโดยที่ 0:

$$\begin{array}{ll} 0111111111 \rightarrow 0000 & | \text{ ผลลัพธ์ } \approx 0.47 \text{ บิตต่อการเก็บ} \\ 1011110111 \rightarrow 00010110 & | \end{array}$$

* บิตที่เก็บข้อมูล (encoding) ต้อง \geq ค่าที่ต้องการได้จากการ H

Lousy compression vs. Lossless compression

Huffman Coding

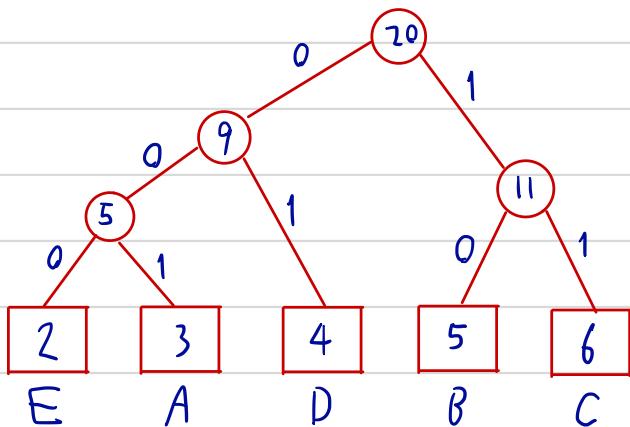
→ ຈະໄດ້ຜົນພວ່ມລົບບ່ອນຕາ ທີ່ $H \leq huffman < H+1$

Message → BCCA BBB DDA ECC BBA EDD CC

→ ຕັ້ງເກີນແນວໆ (ASCII) ດັວນໃຈ $20 \times 8 = 160$ bits

char	count	code
A	3	000
B	5	001
C	6	010
D	4	011
E	2	100

- ຕະຫຼອນຂໍ້ມູນຄົກໜ້າ $20 \times 3 = 60$ bits
- ຕັ້ງໃຊ້ສັນນິຕອງເກີນທາງນິ້ງ ASCII
 $5 \times 8 \rightarrow \text{character}, 5 \times 3 \rightarrow \text{code}$
 $40 + 15 = 55$ bits
- ເກັບພ້ອຍວ້າທັງໝົດ $60 + 55 = 115$ bits



char	count	code	size
A	3	001	$3 \times 3 = 9$
B	5	10	$5 \times 2 = 10$
C	6	11	$6 \times 2 = 12$
D	4	01	$4 \times 2 = 8$
E	2	000	$2 \times 3 = 6$

- ▶ ເຊື້ອກ 2 node ທີ່ໄດ້ນຳຫຼັກ ແລ້ວ merge
- ▶ node ຖະນຸກົນ sum ຂອງ node ດຳກ່າວ
- ▶ ເພີ້ມລົງໄວ່ຢັບ 0 ແລ້ວ 1

- 9 ຊົ່ວໂມງທີ່ເກີນພ້ອມລົງທັງໝົດ 45 bits
- 9 ຊົ່ວໂມງທີ່ເກີນທາງງາງ
 $5 \times 8 \rightarrow \text{character}, 12 \rightarrow \text{code}$
- ເກັບພ້ອຍວ້າທັງໝົດ $45 + 52 = 97$ bits

decode ?

ກົດວິດກາງ tree traversal ມານ root ລົງ leaf

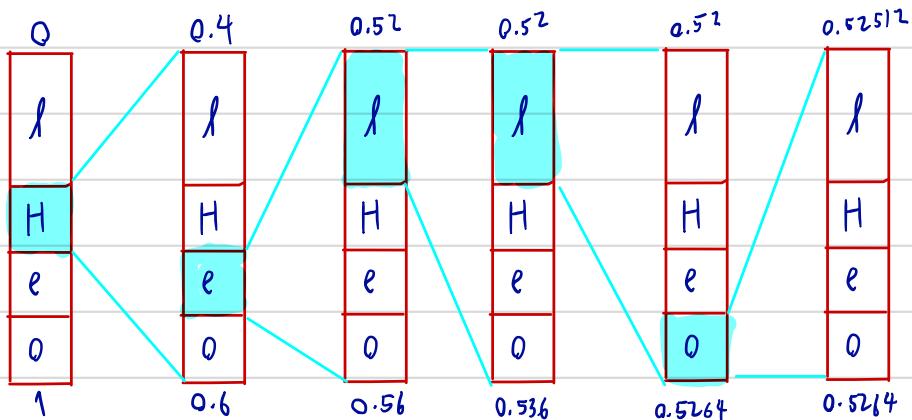
* Arithmetic coding

→ ប្រព័ន្ធផែនក្នុងបញ្ហាបច្ចេកទេស [0,1]

Ex. Hello

	0	0.4	0.6	0.8
char	l	H	e	o
freq	2	1	1	1

$\rightarrow [0,0.4) [0.4,0.6) [0.6,0.8) [0.8,1)$



ចាប់តាំងពី 0.52512?

→ មិនអាមេរោគថា Hello l l l l ... ក្នុង

→ តើជាដែល $[0.52512, 0.5264)$ កំរាលនៅ bit ដែលបាត់

Decode : $0.52512 \in [0.4, 0.6) \rightarrow H$

$$\hookrightarrow (0.52512 - 0.4) / 0.2 = 0.6256$$

$$0.6256 \in [0.6, 0.8) \rightarrow e$$

$$\hookrightarrow (0.6256 - 0.6) / 0.2 = 0.128$$

$$0.128 \in [0, 0.4) \rightarrow l$$

$$\hookrightarrow (0.128 - 0) / 0.4 = 0.32$$

$$0.32 \in [0, 0.4) \rightarrow l$$

$$\hookrightarrow (0.32 - 0) / 0.4 = 0.8$$

$$0.8 \in [0.8, 1) \rightarrow o$$

$$\hookrightarrow (0.8 - 0.8) / 0.2 = 0$$

វិធីកែ:

$$\rightarrow \text{ចាប់ពី } 0.52512 = 5$$

→ ដំណឹងចំណែកក្នុង \$ ដោយប្រើប្រាស់ (ការបញ្ជូនដើម l, H, e, o, \$)

$$H \rightarrow 0.4$$

$$He \rightarrow 0.4 + 0.2 \times 0.6 = 0.52$$

$$Hel \rightarrow 0.52 + 0.2 \times 0.2 \times 0 = 0.52$$

$$Hell \rightarrow 0.52 + 0.2 \times 0.2 \times 0.4 \times 0 = 0.52$$

$$Hello \rightarrow 0.52 + 0.2 \times 0.2 \times 0.4 \times 0.2 \times 0.8 = 0.52512$$

* upper bound នៃក្រោម Huffman Coding

និង Average case នៃក្រោម Huffman Coding

Running code

0000000 1 00

0x7 , +1, 0x2

Encryption

- noise + error detection



- parity bit: 101110 → ระบุจำนวนเลข 1 และจำนวนเลข 0

(ใช้ในการตรวจสอบความถูกต้อง)

- check sum: MD5, SHA, etc.

citizen card, credit card

* password ที่ database ต้องไม่เก็บ password จริงๆ (เก็บ checksum)

→ ห้ามมา เป็นวิธีซึ่งคู่กัน ข้อมูลหายหรือผิด

แต่ต้องการรู้ว่าข้อมูลผิดเท่านั้น : repetition

1011 → 111 000 111 111

detect 000 → 001, 010, 100

111 → 011, 101, 110

Hamming Distance. "ระยะห่าง" คือความแตกต่างเป็นจำนวนบิต

Ex 000, 001 → hamming distance = 1

000, 111 → hamming distance = 3

Hamming code:

-(3 parity bits for 4 data bits)

1 0 1 1

d₁ d₂ d₃ d₄

{P₁, P₂, P₃, d₁, d₂, d₃, d₄}

P₁ 1 0 × 1 ⇒ 0.

P₂ 1 × 1 1 ⇒ 1

∴ จะได้ hamming code บน 1011 010

P₃ × 0 1 1 ⇒ 0

▶ สื่อสารกันผ่านช่องทางเดียว พลัด 1010010

จะแนบ List ที่มีดัง:

$$\begin{array}{cccc}
 1 & 0 & 1 & 0 \\
 d_1 & d_2 & d_3 & d_4 \\
 P_1 & 1 & 0 & \times & 0 \Rightarrow 1 \\
 P_2 & 1 & \times & 1 & 0 \Rightarrow 0 \\
 P_3 & \times & 0 & 1 & 0 \Rightarrow 1
 \end{array}
 \left. \begin{array}{l}
 \text{ผิดกับ parity bit ที่งั้นจะถูก削除} \\
 \text{ผิดกับ parity bit ที่งั้นจะถูก削除}
 \end{array} \right\}$$

∴ แสดงว่า data list ที่ผิด (ทำให้ parity bit ผิดพลาด) $\Rightarrow d_4$

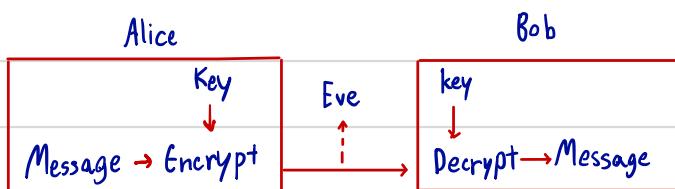
▶ สื่อสารกันผ่านช่องทางเดียว พลัด 1011011

จะแนบ List ที่มีดัง:

$$\begin{array}{cccc}
 1 & 0 & 1 & 1 \\
 d_1 & d_2 & d_3 & d_4 \\
 P_1 & 1 & 0 & \times & 1 \Rightarrow 0 \\
 P_2 & 1 & \times & 1 & 1 \Rightarrow 1 \\
 P_3 & \times & 0 & 1 & 1 \Rightarrow 0
 \end{array}
 \left. \begin{array}{l}
 \text{ผิดกับ parity bit 1 ตัว} \\
 \text{ผิดกับ parity bit 1 ตัว}
 \end{array} \right\}$$

ไม่ว่า data list ค่าไหนที่ผิดไปแล้วทำให้ต้องตัด parity bit ผิดแต่ 1

∴ parity bit P_3 ผิด



Key:

- Symmetric Key

- Public Key

- Caesar Cipher: $ABCD \xrightarrow{(3)} DEFG$

- Substitution Cipher: $A-Z \rightarrow A-Z$ (ล่าสุดปัจจุบัน)

- decode:

- Brute-Force: $26!$

- Frequency analysis:

• Enigma machine (Poly alphabet cipher)

Text: "Hello world"

- គ្រប់ចុងក្រោមនៃការ encrypt: ស៊ុណី sunday (+19 +21 +14 +4 +1 +25)

- នឹង key តាមរយៈបញ្ជីការដែលត្រួតពិនិត្យ:

Hello - World
 ↗
 +19 +21 +14 +4 +1 +25 +19 +21 +14 +4
 Az Z Pp - V h m z h

នៅពេល: នឹង key នៅលើ 4 នូវការ (secret key)

→ នូវការទូទាត់ថា ការបង្កើតចំណាំអាប់រំលឹងការពិនិត្យសំរាប់ secret key នឹង

រូបរាង: $R_1 + R_2 + \dots + R_n \equiv \text{key} \pmod{M}$

→ នូវការបង្កើតចំណាំ 5 នូវលេខ 3 គឺត្រូវបានរាយការណ៍ដើម្បី secret key

រូបរាង: - polynomial degree 2 (តូចធី 3 គួរតិចជាអក្សរ $a_2x^2 + a_1x + a_0$)

Finite Field: $\{0, 1, 2, \dots, n-1\}$ និង operation $\oplus, \ominus, \otimes, \div$ ដែលត្រូវបានរាយការណ៍ដោយ

$n=7$: $\{0, 1, \dots, 6\}$

$$3 \oplus 5 \equiv 3+5 \pmod{7}$$

$$\equiv 1 \pmod{7}$$

$$3 \otimes 6 \equiv 3 \cdot 6 \pmod{7}$$

$$\equiv 4 \pmod{7}$$

$$5 \ominus 3 \equiv 5-3 \pmod{7}$$

$$\equiv 2 \pmod{7}$$

$$n \otimes m \equiv n \times m^{-1} \pmod{7}$$

$$(m, 7) = 1 \text{ កំណត់ថា } m^{-1} \text{ នឹង}$$

Polynomials in Finite Field

$$f(x) = x^2 - 2x + 1$$



នូវការរាយការណ៍ 3 ពីរីករាយការណ៍ $f(0)$ ។

$$\Delta(x) = \frac{\prod_{i>j} (x-x_i)}{\prod_{i>j} (x_i-x_j)} \quad ; \quad p(x) = \sum_{i=0}^{d+1} y_i \Delta_i(x)$$

Advance Data Structure

Van Ende Boas Trees

features:

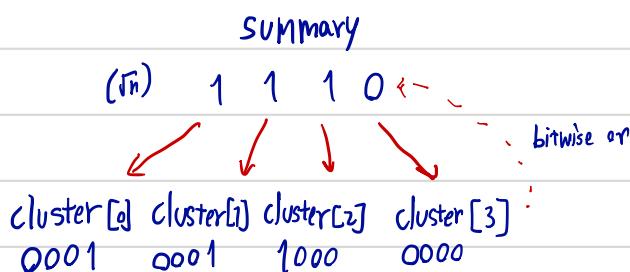
- insert
 - delete
 - check if x is a member
 - min
 - max
 - successor (x)
 - predecessor (x)

↓ คือจะที่มากกว่าเท่ากับ x
 ↓ คือจะที่น้อยกว่าเท่ากับ x

► លេខ: implement តាម hash

- insert, delete, check = $O(1)$
 - min, max, predecessor, successor = $O(n)$

► Summary array (\sqrt{n} cluster arrays of \sqrt{n} bits)



- check, insert = $O(1)$
 - delete - ลบจาก cluster $\approx O(\sqrt{n})$
- ลบจาก summary ตัวบ
• check ว่า min/max cluster == null?
• min, max - หาจาก summary ก่อน $\approx 2\sqrt{n} = O(\sqrt{n})$
- check ว่า min/max ของ cluster ต้องอะไร \rightarrow return concat (ชื่อ summary, ชื่อ cluster)
 - successor, predecessor $\approx 3\sqrt{n} = O(\sqrt{n})$
- หา cluster ใหม่ ตัวถัดไป
- หา cluster ใหม่ ตัวก่อนหน้า
- หา cluster ใหม่ ที่มีค่าต่ำสุด

Recursion: constructs a summary/cluster for each level

- check: $T_c(n) = T_c(\sqrt{n}) + 1 \rightarrow O(\log \log n)$
- insert: $T_i(n) = 2T_i(\sqrt{n}) + 1 \rightarrow O(\log n)$
- delete: $T_d(n) = 2T_d(\sqrt{n}) + T_{\min}(\sqrt{n}) + 1 \rightarrow O(\log n \log \log n)$
- min: $T_{\min}(n) = 2T_{\min}(\sqrt{n}) + 1 \rightarrow O(\log n)$ (same for max)
- successor: $T_{succ}(n) = 2T_{succ}(\sqrt{n}) + T_{\min}(\sqrt{n}) + 1 \rightarrow O(\log n \log \log n)$ (same for predecessor)

$$* T_c(u) = T_c(\sqrt{u}) + 1$$

$$\sqrt{u} = l = \log u$$

$$\therefore X(l) \in O(\log l)$$

$$\therefore T_c(z^l) = T_c(2^{\frac{l}{2}}) + 1$$

$$T_c(z^l) \in O(\log l)$$

$$\text{but } X(l) = T_c(z^l)$$

$$T_c(u) \in O(\log \log u)$$

$$X(l) = X\left(\frac{l}{2}\right) + 1$$

► van Ende Boas Trees

- find max or min element \rightarrow binary search tree bit voldoende voor 1

- interface:
 - min, max = constant time $O(1)$

- min == null: data structure is empty

- min == max \neq null: data structure contain one element

- insert into empty data structure \rightarrow set min == max == u $(O(1))$

- delete from a data structure that contains one element \rightarrow set min == max == null $(O(1))$

- check:
 - if $x = \min \rightarrow$ return true

- return cluster [high(x)].check(low(x));

$$T_c(u) = T_c(\sqrt{u}) + 1 \quad O(\log \log n)$$

- successor:
 - if $\min \neq \text{null}$ $\wedge u < \min$ return min

- $\neq \max$ $\neq \text{null}$ cluster

Linear Algebra

Vector និគតិស៊រ

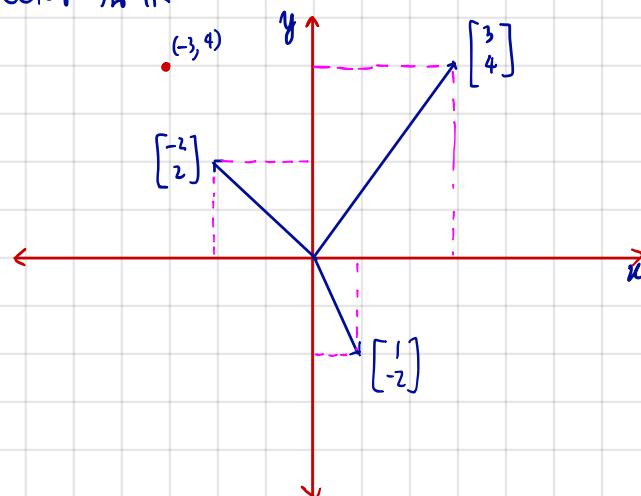
ជាសម្រាប់ការគិតវិទ្យានុវត្តន៍ និងការគិតវិទ្យានុវត្តន៍

In CS vector សំណងការគិតវិទ្យានុវត្តន៍

In math vector និគតិស៊រ

vector និគតិស៊រ និងគិតវិទ្យានុវត្តន៍ matrix

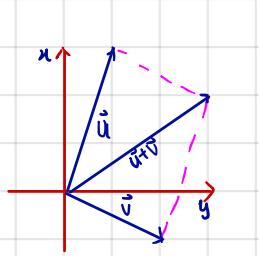
Vector នៃ \mathbb{R}^2



u នៅ u axis, y នៅ y axis
និង 1 ដីបុ

ការរួចរាល់ vector នៃ \mathbb{R}^2 Point ទាំង ១

សរុបនៃ vector នៃ vector

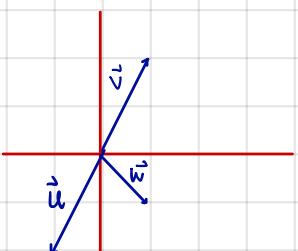


$$\vec{U} = \begin{bmatrix} a \\ b \end{bmatrix}, \vec{V} = \begin{bmatrix} c \\ d \end{bmatrix}, \vec{U} + \vec{V} = \begin{bmatrix} a+c \\ b+d \end{bmatrix}$$

$$\vec{U} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}, \vec{V} = \begin{bmatrix} 2 \\ -1 \end{bmatrix}$$

$$\vec{U} + \vec{V} = \begin{bmatrix} 1 \\ 3 \end{bmatrix} + \begin{bmatrix} 2 \\ -1 \end{bmatrix} = \begin{bmatrix} 1+2 \\ 3-1 \end{bmatrix} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

Span



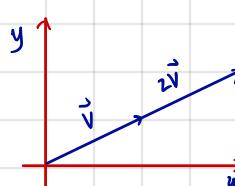
សំណងការការពារឱ្យ \vec{v}, \vec{w} និង \vec{u} , \vec{v}, \vec{w} និង \vec{u}

$a\vec{v} + b\vec{w}$ និង $a\vec{v} + b\vec{w}$ reachable

អ្នកកំណត់ នូវ $a, b \in \mathbb{R}$

អ្នកកំណត់ នូវ $a\vec{v} + b\vec{w}$ ដែលនិគតិស៊រ reachable
អ្នកកំណត់

ការសរុបនៃ vector នៃ \mathbb{R}^3

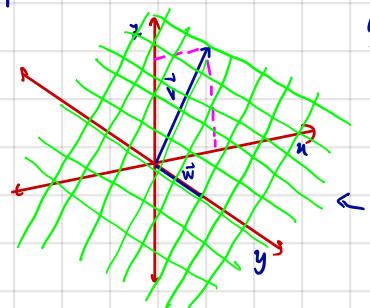


$$\vec{V} = \begin{bmatrix} a \\ b \end{bmatrix}, k\vec{V} = \begin{bmatrix} ak \\ bk \end{bmatrix}$$

$$\vec{V} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$$

$$2\vec{V} = 2 \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2(2) \\ 2(1) \end{bmatrix} = \begin{bmatrix} 4 \\ 2 \end{bmatrix}$$

Span នៃ \mathbb{R}^3



$\vec{v}, \vec{w}, \vec{u}$ linearly independent
នៅក្នុង span នៃ surface

ការសរុបនៃ \vec{v}, \vec{w} linearly independent,
 \vec{u}, \vec{v} linear dependent

"span" នៃ \vec{v}, \vec{w} នឹង set នៃ linear combination នូវវឌិត $a\vec{v} + b\vec{w}$

សំណងការការពារឱ្យ scalar និង vector និង scalar

សំណងការការពារឱ្យ scalar និង vector និង scalar

$$\hat{i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \hat{j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} a \\ b \end{bmatrix} = a\hat{i} + b\hat{j} \quad \begin{bmatrix} 3 \\ 2 \end{bmatrix} = 3\hat{i} + 2\hat{j}$$



Matrices and linear transformation

Transformation i.e. function $\vec{v} \rightarrow \vec{w}$ vector $\text{if } A = \text{return vector}$

maps to origin $\vec{0}$ if $\vec{v} = \vec{0}$ $\text{no scale or rotation}$

$T: \mathbb{R}^n \rightarrow \mathbb{R}^n$ is a linear transformation if

$$1) T(u+v) = T(u) + T(v)$$

$$2) T(ku) = kT(u)$$

$$\vec{i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \vec{j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \text{ then } \vec{v} \text{ has form } a\vec{i} + b\vec{j} = a\begin{bmatrix} 1 \\ 0 \end{bmatrix} + b\begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\text{If } T(\vec{i}) \text{ is linear transformation vec } \vec{i} \text{ then } T(\vec{i}) = x_i \vec{i} + y_i \vec{j} = \begin{bmatrix} x_i \\ y_i \end{bmatrix}$$

$$\text{If } T(\vec{j}) \text{ is linear transformation vec } \vec{j} \text{ then } T(\vec{j}) = x_j \vec{i} + y_j \vec{j} = \begin{bmatrix} x_j \\ y_j \end{bmatrix}$$

$$\text{then } \vec{v} = a\vec{i} + b\vec{j} = \begin{bmatrix} a \\ b \end{bmatrix}, \text{ If } T(\vec{v}) \text{ is linear transformation vec } \vec{v}$$

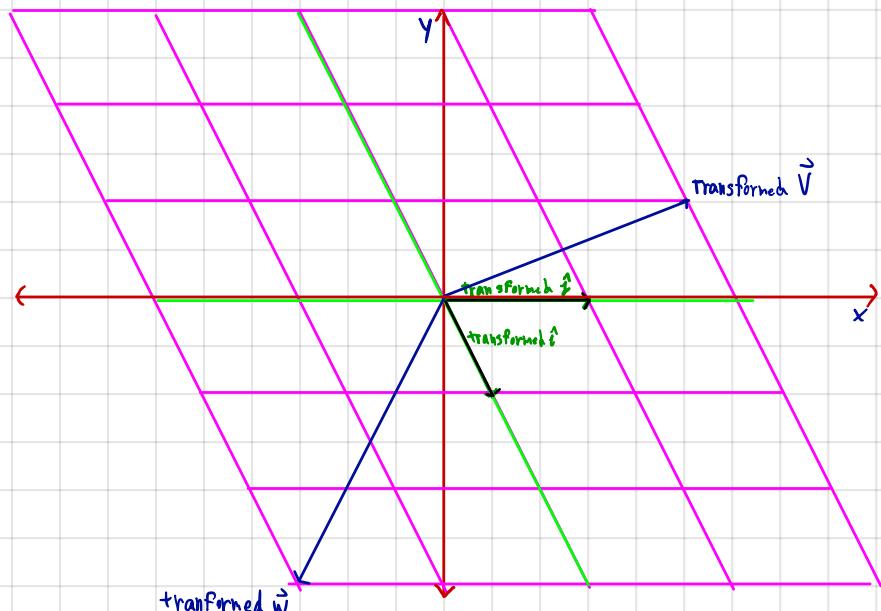
$$\text{then } \vec{v}' = a\vec{i}' + b\vec{j}' = a\begin{bmatrix} x_i \\ y_i \end{bmatrix} + b\begin{bmatrix} x_j \\ y_j \end{bmatrix} = \begin{bmatrix} ax_i \\ ay_i \end{bmatrix} + \begin{bmatrix} bx_j \\ by_j \end{bmatrix} = \begin{bmatrix} ax_i + bx_j \\ ay_i + by_j \end{bmatrix}$$

$$\text{so } T(\vec{v}') = a\begin{bmatrix} x_i \\ y_i \end{bmatrix} + b\begin{bmatrix} x_j \\ y_j \end{bmatrix} = \begin{bmatrix} x_i & x_j \\ y_i & y_j \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}$$

$$\vec{v} = -1\vec{i} + 2\vec{j} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}$$

$$\vec{v}' = -1\vec{i}' + 2\vec{j}' = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \vec{j}' = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

$$\vec{w}' = 3\vec{i}' - 2\vec{j}' = \begin{bmatrix} 3 \\ -2 \end{bmatrix}$$



$$\text{so } T(\vec{v}) = \vec{v}' \text{ where } \vec{i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \vec{j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow \vec{i}' = \begin{bmatrix} 1 \\ -2 \end{bmatrix}, \vec{j}' = \begin{bmatrix} 3 \\ 0 \end{bmatrix}$$

$$\text{so } \vec{v}' = -1\vec{i}' + 2\vec{j}' = -1\begin{bmatrix} 1 \\ -2 \end{bmatrix} + 2\begin{bmatrix} 3 \\ 0 \end{bmatrix} = \begin{bmatrix} -1 + 6 \\ 2 + 0 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

$$\text{so } \vec{w}' = 3\vec{i}' - 2\vec{j}' = 3\begin{bmatrix} 1 \\ -2 \end{bmatrix} - 2\begin{bmatrix} 3 \\ 0 \end{bmatrix} = \begin{bmatrix} 3 - 6 \\ -6 + 0 \end{bmatrix} = \begin{bmatrix} -3 \\ -6 \end{bmatrix}$$

using linear transformation \Rightarrow Matrix multiplication \Rightarrow

$$\vec{v} = \begin{bmatrix} -1 \\ 2 \end{bmatrix}, L(\vec{v}) = \begin{bmatrix} 1 & 3 \\ -2 & 0 \end{bmatrix} \begin{bmatrix} -1 \\ 2 \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$$

$$\vec{w} = \begin{bmatrix} 3 \\ -2 \end{bmatrix}, L(\vec{w}) = \begin{bmatrix} 1 & 3 \\ -2 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ -2 \end{bmatrix} = \begin{bmatrix} -3 \\ -6 \end{bmatrix}$$

Graph \rightarrow Undirected
 \rightarrow directed

$$G = (V, E), E \subset V^2$$

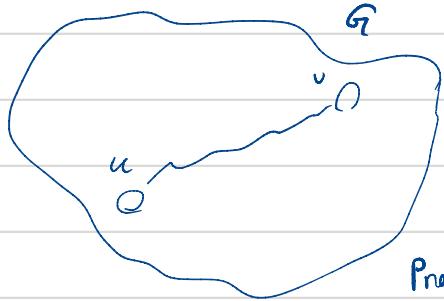
path - $\{u_1, u_2, \dots, u_k\} = P$ if $(u_i, u_{i+1}) \in E$ u_1 is source and u_k is end

simple path - path not vertex $\in P$

simple cycle - cycle not having vertex $\in P$ (i.e. no $u_i \in P$ such that u_i, u_k)

Connectivity

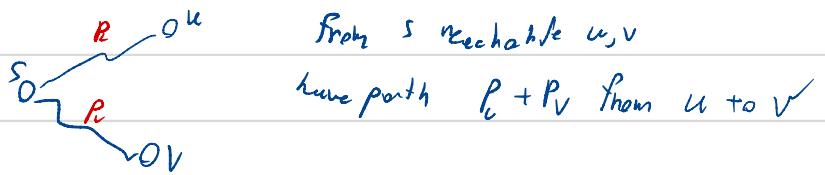
Def vertex $u \rightarrow$ reachable from v if have path from u to v



Proof

Graph is connected if for each edge $u, v \in V$
 u reachable v

Claim if have $u, v \in S \rightarrow u$ reachable from S , G connected



$C \subseteq V$ is connected component if

\Rightarrow for all vertices $u, v \in C$, u reachable from v

\Leftarrow C is set that maximal \leftarrow biggest inclusion

haven't have set C' that $C \subsetneq C'$

$$n=|V|, m=|E|$$

① Connected component not increasing

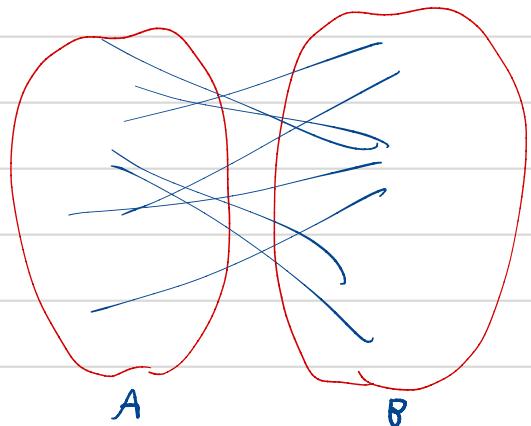
② Connected component decreasing 1 or not decreasing

③ if Graph connected, have 1 connected component
 \Rightarrow use $n-1$ edges at least

Def: Graph acyclic if no cycle in a graph

Bipartiteness $G = (V, E)$ is bipartite (2-colorable)

if have $A, B \subseteq V$, $A \cup B = V$, $A \cap B = \emptyset$
for all edge $(u, v) \in E$, $(u \in A \wedge v \in B) \vee (v \in B \wedge u \in A)$



certificate if $\text{Lanc}(A, B)$ refer bipartite

(Directed graph) Acyclicity

1) if graph have cycle \Rightarrow certificate no cycle \Rightarrow cycle

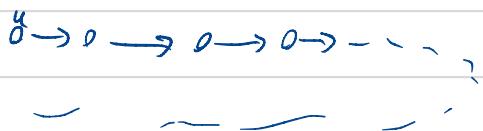
2) if graph acyclic

(certificate) topological order of graph

Sequence (u_1, u_2, \dots, u_n) is topological order if for all arc $(u_i, u_j) \in E$, $i < j$
if graph have topological ordering \Rightarrow acyclic graph

\Rightarrow graph acyclic \Leftrightarrow vertex has no edge

(proof sketch)



claim If G acyclic have vertex \nexists edge from

Proof well-order process

1. Choose vertex u

2. While have edge $(u, v) \in E$ $v \neq u$

claim: process terminate; if not terminate after m loop

Lemma: if G acyclic, have topological ordering

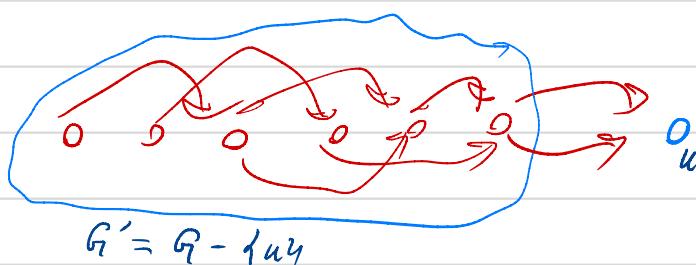
Proof by induction

Show claim G will have u that don't have ancestor

Given $G' = G - \{u\}$

by induction

G' have topo ordering



$$G' = G - \{u\}$$

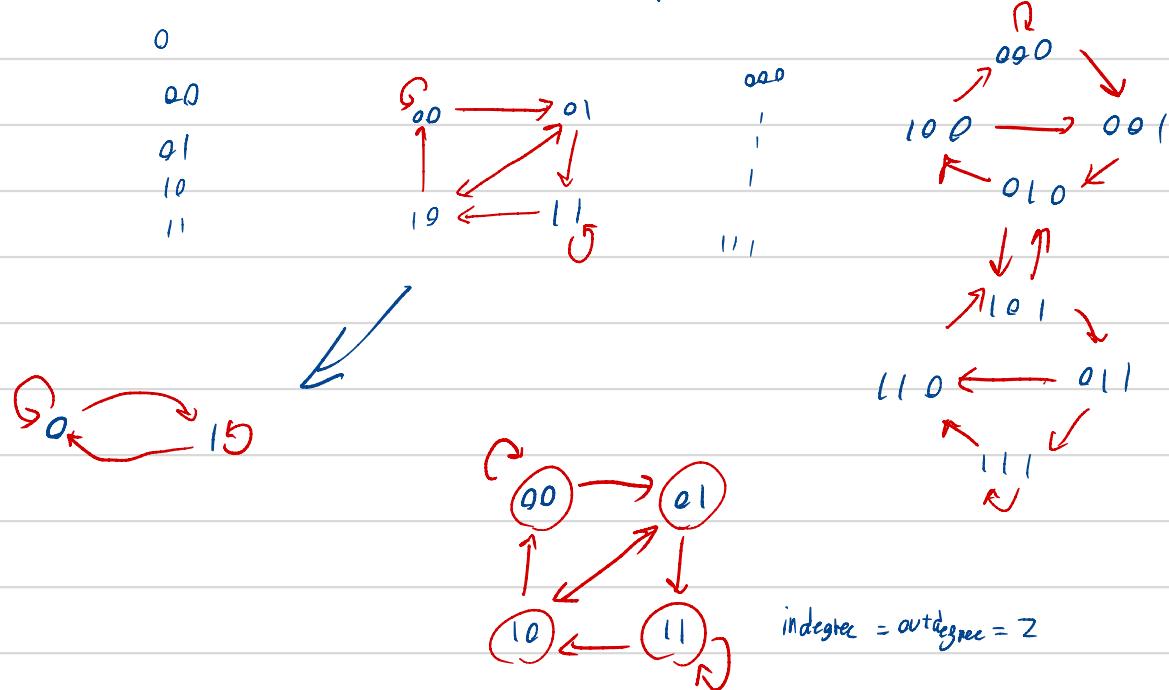
can build topological ordering of G from topo ordering G' plus u

Implement $\mathcal{O}(n^2)$, or $\mathcal{O}(m+n)$

undirected graph $G = (V, E)$ cycle C is Euler tour if C traversed all edge, & for each $e \in C$,

then if G connected & deg of all vertex is even \rightarrow have Euler tour

Def. Given $G(V, E)$, a simple cycle H is hamiltonian cycle if H pass all vertex



$G = (V, E)$, \deg varia $v \in V$, $\deg(v) \geq 0$

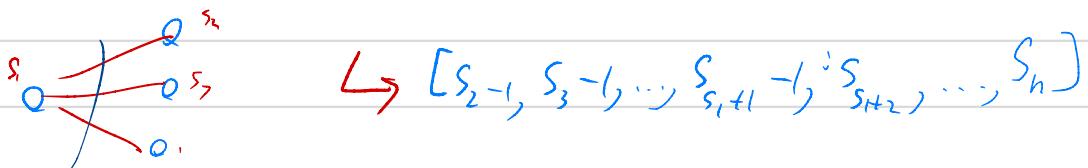
Fact (Hand-shaking lemma) $\sum \deg(v) = 2m$



Def. sequence S is graphic sequence if graph G set of $\deg v$ of $G = S$

Greedy strategy

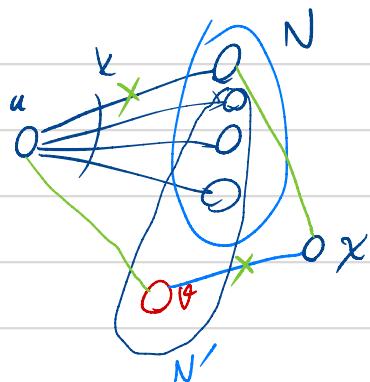
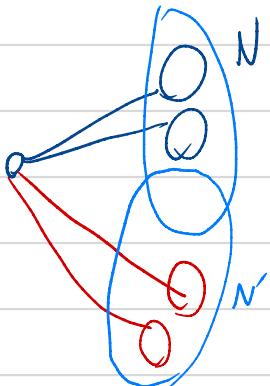
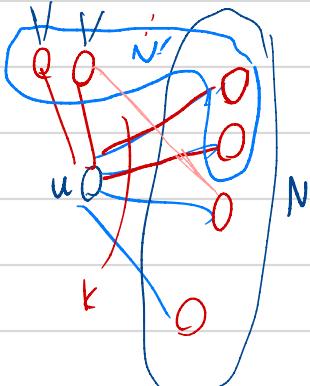
Input: $s_1, s_2, \dots, s_n \Rightarrow s_i \geq s_{i+1}, 1 \leq i < n$



für N Menge von Vertices existiert G für G

für N Menge von Vertices $\{d_i\}_{i \in N}$ existiert G für G

Proof

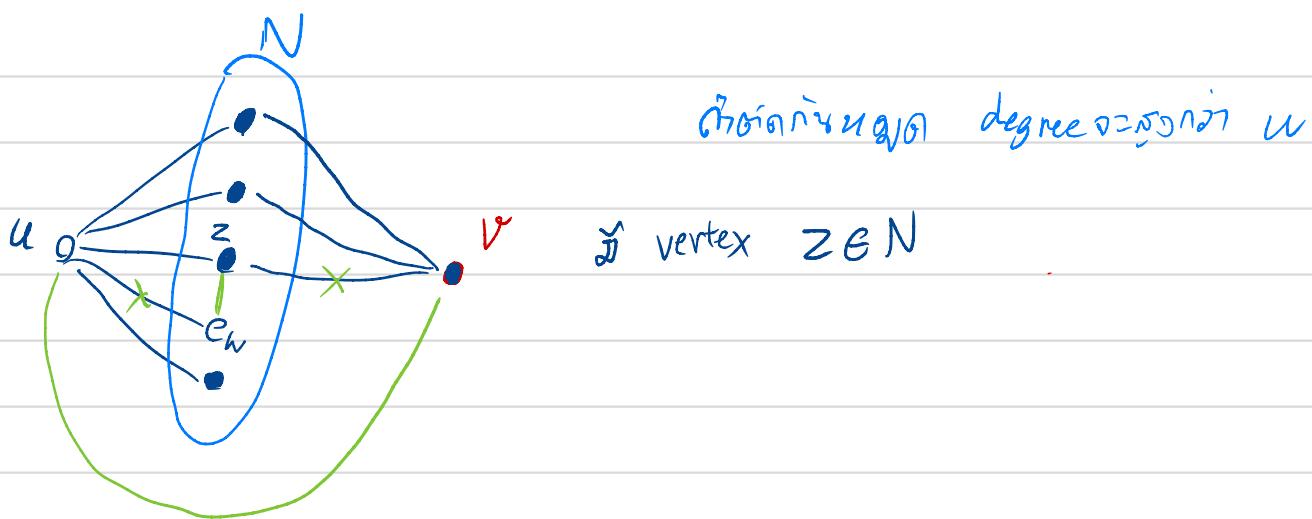


\Rightarrow G' an G ändert $w \in N$

für x nur vertice können v

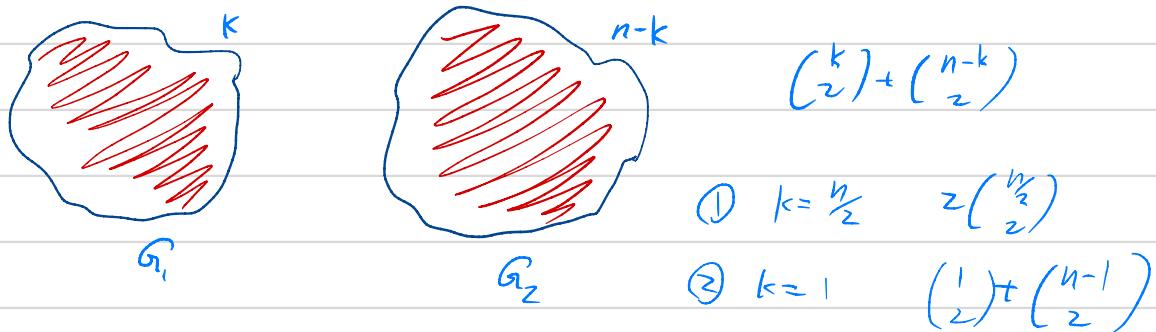
$$G' = G - \{(x, v)\} \cup \{(u, v), (w, x)\}$$

Case 1. $x \notin N$ $\Rightarrow w \in N$ mit \exists edge $(x, w) \in E$, \exists l^*

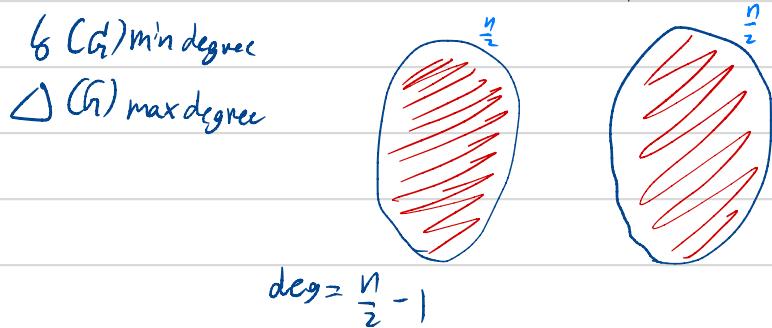


External Problems

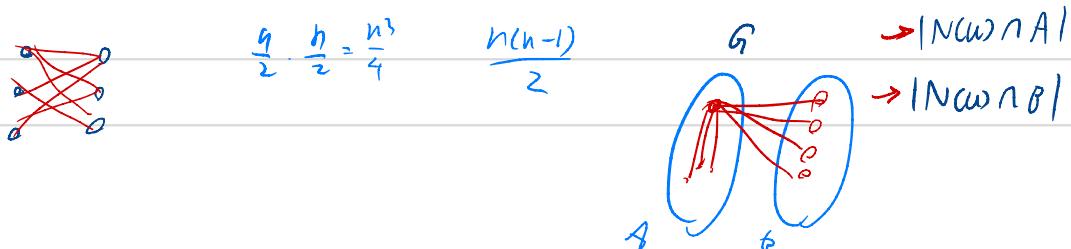
Question 1 If n vertices at edge arrangement edge minimally connected



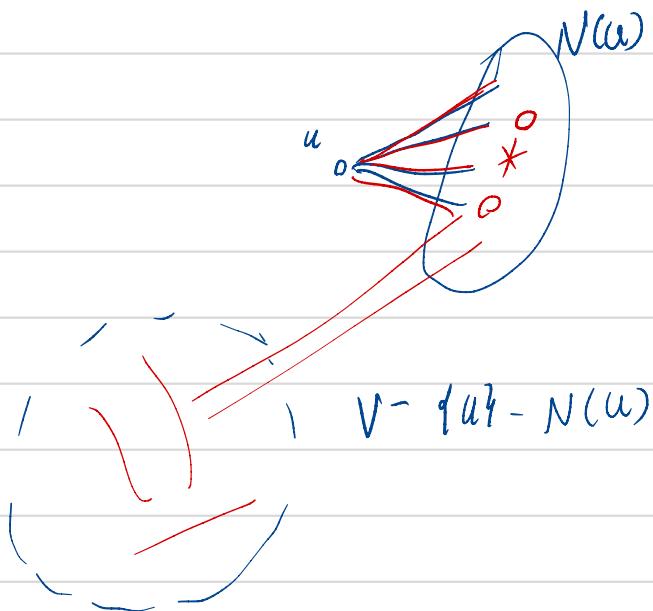
Question 2 If deg: minimum $\left[\frac{n-1}{2}\right]$ minimum node Graph Connected
 $\delta(G) \geq \left[\frac{n-1}{2}\right]$, G connected



Question 3 If $G = (V, E)$ \Rightarrow SubGraph of a bipartite Graph with exactly k



$\forall u \in V$ vertex u degree $\deg(u)$ $\exists k = \deg(u)$



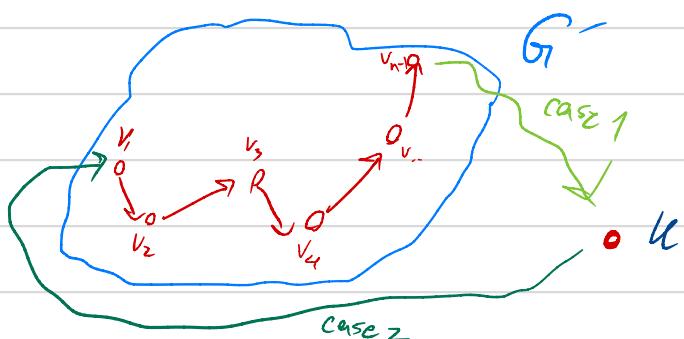
$k + (n-k-1)k$ upper bound of size

Tournament A directed graph $G = (V, E)$ is a tournament $\forall v \in V$ $\forall u \in V$ $(u, v) \in E$ or $(v, u) \in E$ exactly one



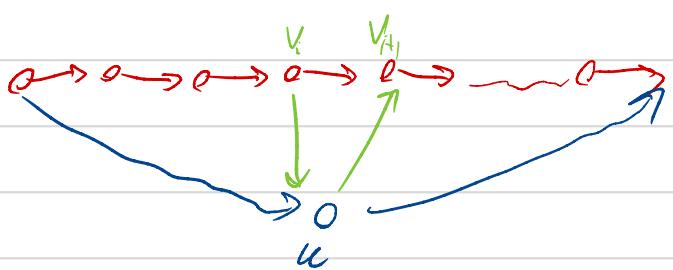
Thm: $\forall n \in \mathbb{N}$ Tournament G on n v has Hamiltonian path $\exists_{G'} G$

Proof: Base case $n=2$ $\rightarrow \checkmark$

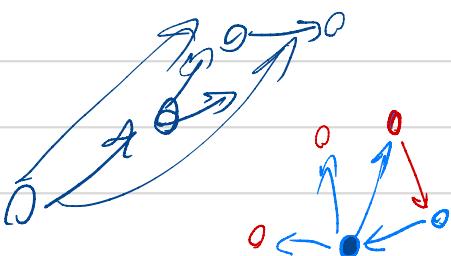


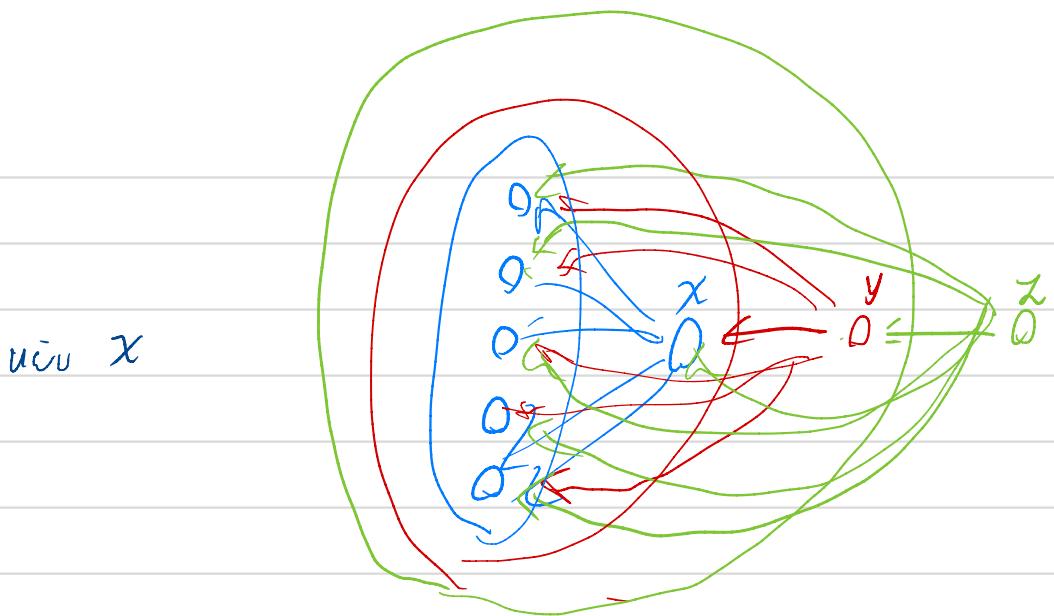
by induction on the path of G'

Def: In directed graph, vertex u is king if every other vertex is reachable from u by paths of length 1 or 2.

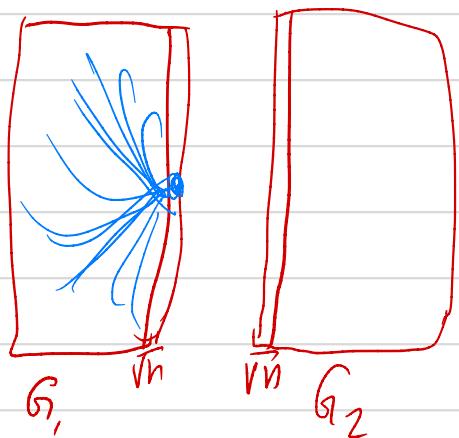
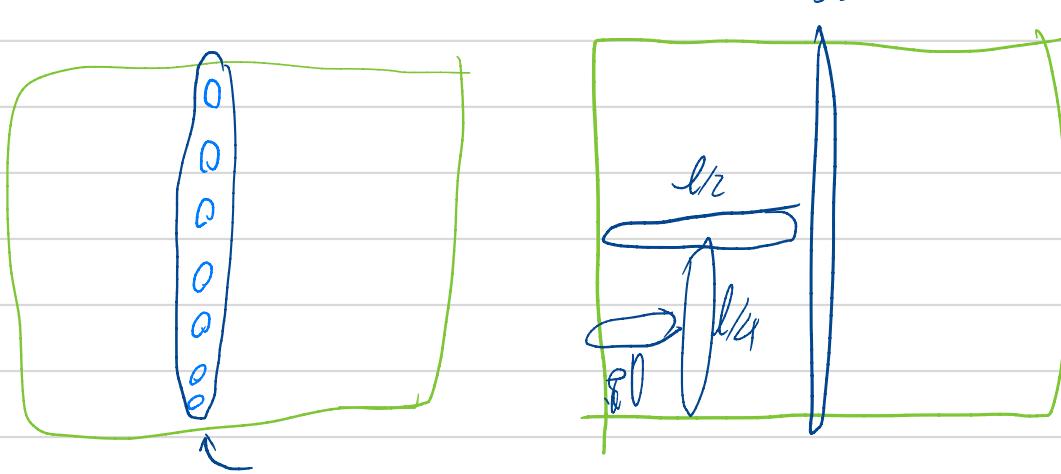
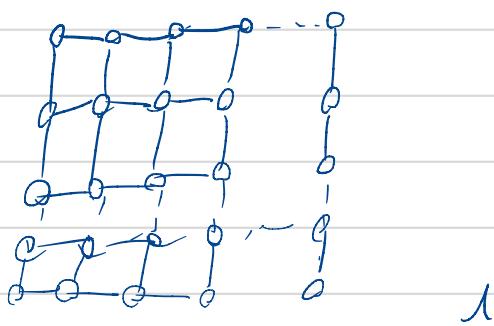


Lemma: \forall tournament \exists king.

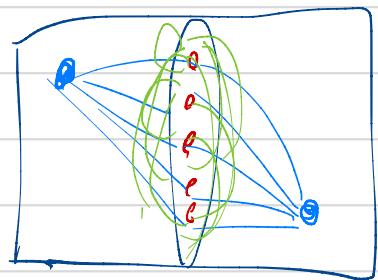




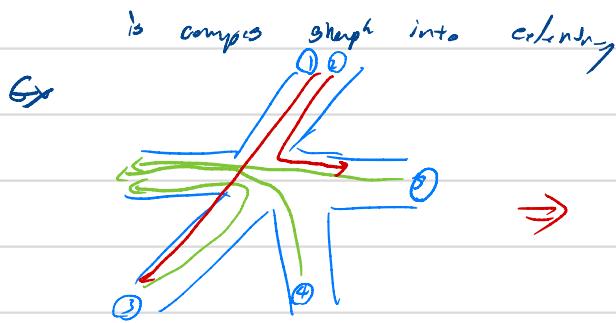
Grid graphs



running BFS after single source shortest path
 $O(D \cdot n \log n)$
 $= O(\sqrt{n} \cdot n \log n)$



Graph Coloring



For each graph G such neighbour \tilde{G} is given

Vertex coloring $\exists c: V \rightarrow \{1, \dots, k\}$ s.t. $\forall (u, v) \in E, c(u) \neq c(v)$.

$\chi(G)$ chromatic number \rightarrow min number of colors to vertex color G

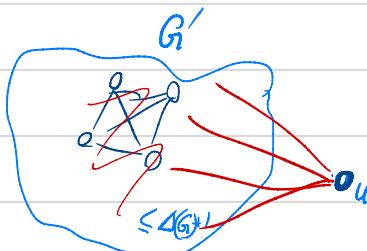
If G is k -colorable \Rightarrow min k

For every $v \in V, \Delta(G) = \deg(v)$

Lemma: $\chi(G) \leq \Delta(G) + 1$

Proof: Induction on number of vertex Base case $\varnothing \checkmark$

Inductive step



W.L.O.G. $G(V, E)$ has vertices $w, w \in V$ s.t. $G' = G - \{w\}$

so $\Delta(G') \leq \Delta(G)$

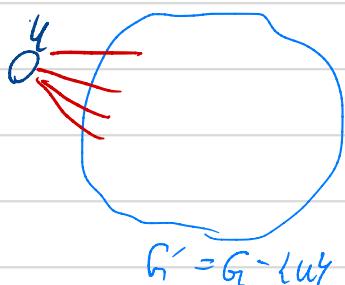
$\deg(w) \leq \Delta$, $\Rightarrow \Delta + 1$ colors $\Rightarrow w$ is colored

Brooks' theorem: If G isn't complete graph V cycle of size 3

$$\chi(G) \leq \Delta$$

Claim consider a $\delta(G) < \Delta(G)$, $\chi(G) \leq \Delta(G)$

Hint: when w if $\deg(w) < \Delta(G)$ \Rightarrow $\chi(G) \leq \delta(G) + 1$



since $\delta(G') < \Delta(G')$

abs from χ and coloring

observation $\exists G$ s.t. $\Delta(G) = 100$ but $\chi(G) = 2$



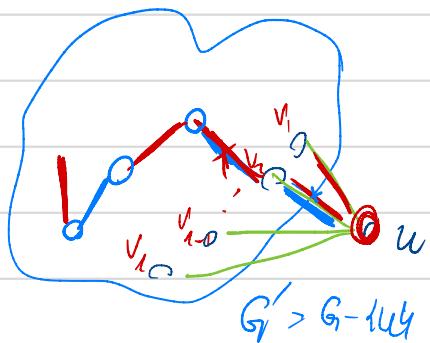
Edge Coloring

Def: Find $c: E \rightarrow \{1, \dots, k\}$, $\forall e \in V$, e_1, e_2 adjacent $\Rightarrow c(e_1) \neq c(e_2)$

$\chi'(G)$ = edge chromatic number

$$\Delta(G) \leq \chi'(G) \leq \Delta(G) + 1$$

Lemma If G is bipartite, $\chi'(G) = \Delta(G)$



assume $\deg(v_1) < \Delta$, $\deg(v_2) < \Delta, \dots$

v_1 has k_1 , v_2 has k_2 , ...

$\Rightarrow \Delta = \Delta$ \checkmark

$k_1 > k_2$

assume cycle arrangement \Rightarrow \exists v_1, v_2

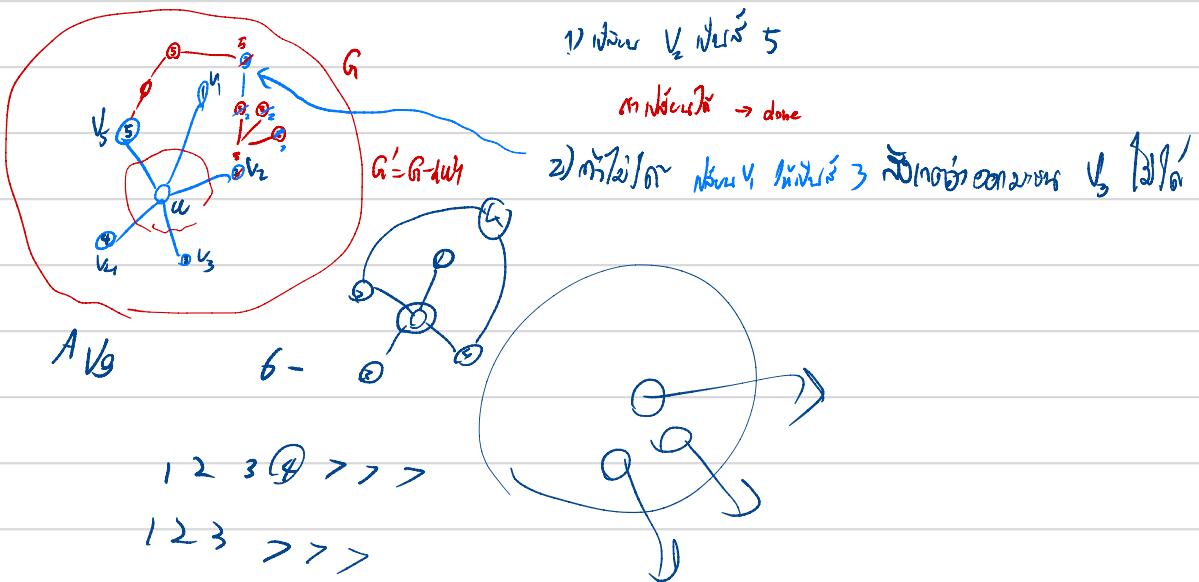
4-color theorem

Statement: every planar graph G has $\chi(G) \leq 4$.

5-color theorem

claim for planar graph $n=25$ vertex u in $\deg(u) \leq 5$

Inductive step

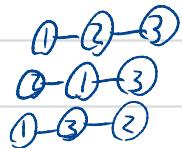


Question: Labeled tree numbering

$$n=1 \quad 0$$

$$n=2 \quad ①-②$$

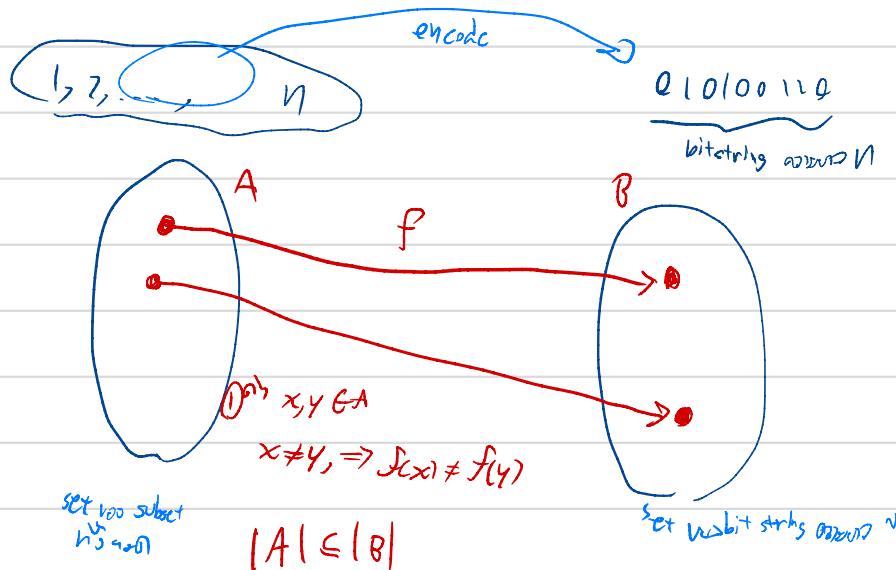
$$n=3$$



Cayley's formula

1. Labeled tree numbers

$$n^{n-2}$$



$$\text{② if any } a \in P \text{ such } x \in A \text{ in } f(a) = Q \Rightarrow |B| \leq |A|$$

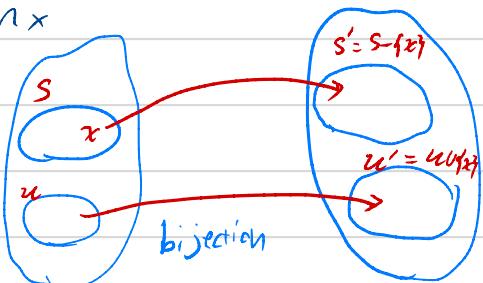
$$\text{① ②} \Rightarrow |A| = |B|$$

$${n \choose 0} + {n \choose 1} + {n \choose 2} + \dots = {n \choose 1} + {n \choose 3} + {n \choose 5}$$

$$\text{Pruefer 1} \quad (1-1)^n = {n \choose 0} - {n \choose 1} + {n \choose 2} - {n \choose 3} + \dots = 0$$

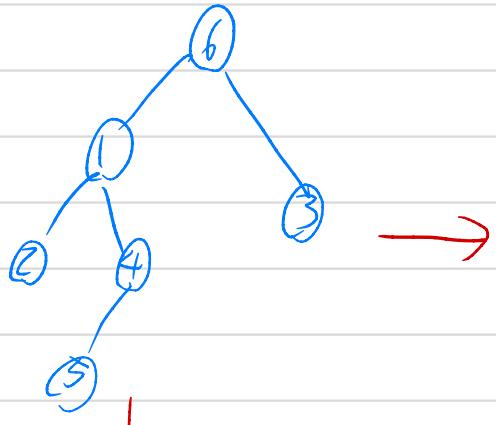
Pruefer 2)

నొన x



$$|A| = |B|$$

Labeled Tree

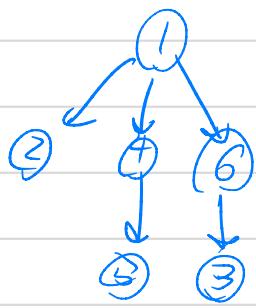


1 6
1 2
1 4
3 6
4 5

Rüper's code

$n-2$ 0s

1 0 0 2 0 2 0 1 1 1 ~ n



2 3 4 5 6
1 6 1 4 1

labeled tree $\leq n^{n-1}$

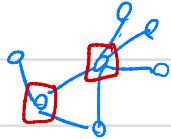
incorrect cycle

Ex 3, 4, 5, 6, 2

Catalan Number

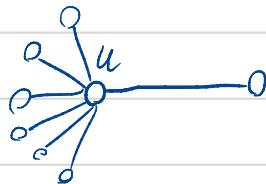
Fixed Parameter algorithms.

vertex cover given $G = (V, E)$ s.t. in $C \subseteq V$ isdr vertex cover in $\forall (u, v) \in E, u \in C \text{ or } v \in C$



Additional parameter : k

Subgraphs have vertex cover of size k . Then C is $|C| \leq k$ isdr vertex cover val G



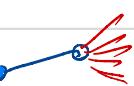
neighbor u neighbour \rightarrow neighborhood
node degree 1 \rightarrow isolation

Data reduction

① if $\deg(u) > k$ then u

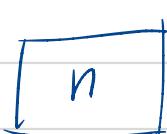


② if $\deg(u) = 1$, then $v \in (u, v) \in E$

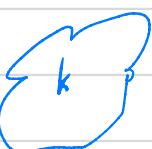


↳ Assume that graph have $\Delta \leq k$

→ two edge $\leq k^2$



data reduction



$$T(n) = T(n-1) + T(n-1)$$

Max Sat for CNF formula ϕ , integer k

ex $(x_1 \vee \neg x_2 \vee x_3) \wedge (x_4 \vee \neg x_1 \vee x_5) \wedge (\neg x_2 \vee x_3 \vee x_5) \wedge \dots$

min assignment n satisfy $\geq k$ clauses

if $n = t$ clauses

Observation minimum assignment n satisfy $\lceil \frac{n}{2} \rceil$ clauses

F_L : long clauses $\Rightarrow \geq k$ literals

$|F_L| \geq k$ clauses $\Rightarrow \geq k$ literals

F_S : short clause $\Rightarrow \leq k$ literals

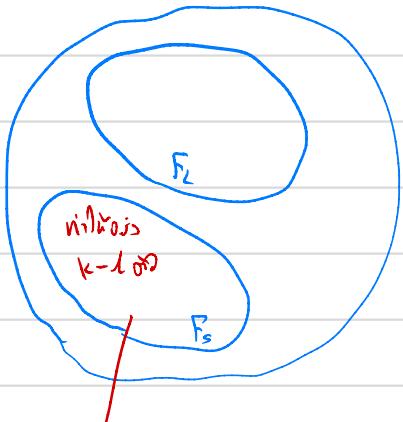


$\Rightarrow |F_L| \geq k \Rightarrow \text{done}$

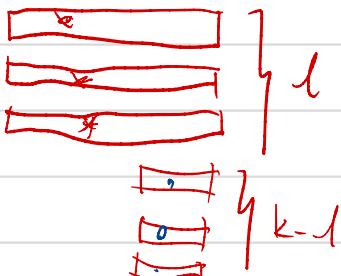
miss F_L , $|F_L| = 0$, $m < k$, min clause $\leq k$ literals

bruteforce, input $\leq m$ $\leq k \cdot k = O(k^2)$

$$\text{für } l = |F_s|$$



mindestens k clauses



l

$k-1$

$|F_S| \geq 2(k-d)$ if observation 1
 $|F_S| \leq 2(k-d) \leq 2k$ brute force

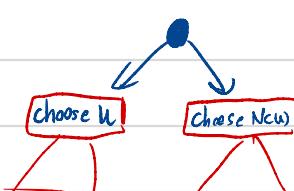
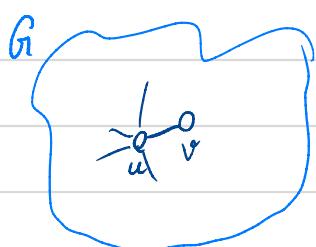
many clauses $\geq 2k$ \Rightarrow NP-hard

1) Data reduction narrowing \rightarrow k-clauses $f(k)$ + brute force,

2) Depth-bounded search tree

Vertex cover für $G = (V, E)$ & k mit vertex cover mit k :

Naive $\binom{n}{k}$ Ansatz



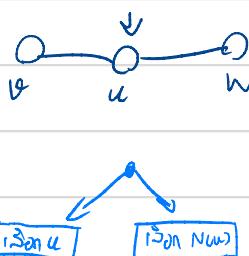
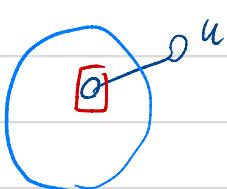
$O(2^k)$ - Anzahl der zu überprüften Kombinationen

- nur eine Kombination ist k-vertex cover

Vertex cover

falls $u \in V$ $\deg(u) = 1$

assume that mindegree = 2

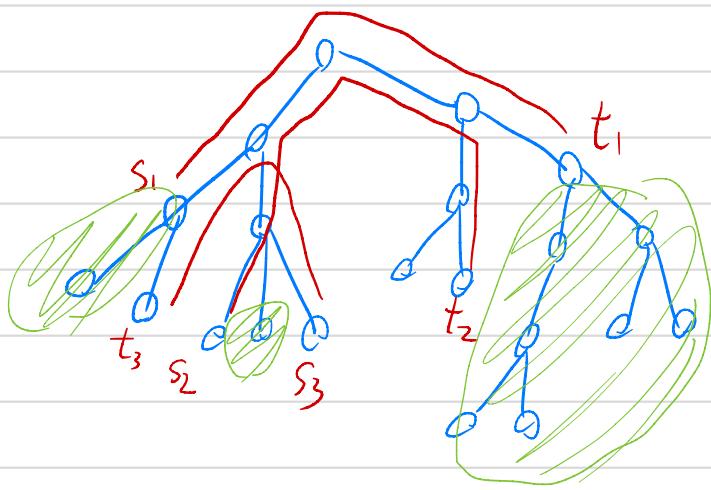


$$T(k) = T(k-1) + T(k-2)$$

$\downarrow \text{ falls } u$ $\downarrow \text{ falls } N(u)$

$$= \varphi^k = O(1.618^k)$$

Multicut in trees



ສົງລະເກີດ ຈະມີການ ລາຍ ວິວທີການ
brute force ອັນ ດັບ ຂົງກົງ / ດັບ ນັກ

Closest string

Input : Strings s_1, s_2, \dots, s_k of length L

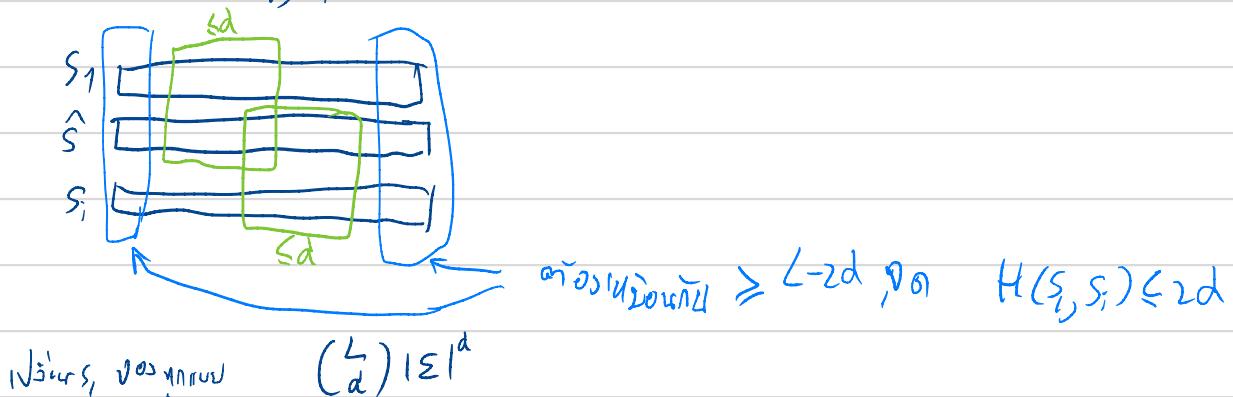
Parameter : d (distance)

Hamming distance : $H(x, y) = |\{i | x[i] \neq y[i]\}|$

$$H(A \text{ (PPE)} \atop A \text{ (TEE)}) = 3$$

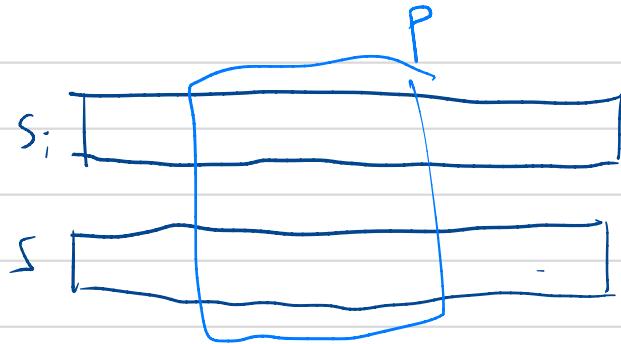
Output : \hat{s} ຕໍ່ $H(\hat{s}, s_i) \leq d, \forall i$

Observation ຂອງ $H(s_i, \hat{s}) > 2d \Rightarrow$ no solution



algo : BRUTE FORCE (S, Δ)
 strings in S in Δ \rightarrow quota in Δ

Terminate: (1) if $\forall i, H(S, S_i) \leq \Delta \Rightarrow \text{sol } S$
 (2) if $\exists i, H(S, S_i) > \frac{2\Delta}{\Delta+1} \Rightarrow \text{NOT Found}$



$\forall j \in P, |S_i[j] \neq S[j]|$

then S_i in $H(S_i, S) > \Delta$

for $j \in P$:

$|S_i[j] = S[j]|$

BRUTEFORCE($S, \Delta - 1$).

$$\begin{aligned} & O((2\Delta)^d) \\ & \Downarrow \\ & O((\Delta + 1)^d) \end{aligned}$$

VB Tree

operation

S.insert(x)

assume that $\{0, \dots, u-1\}$ where u doesn't matter of n

S.delete(x)

* support class writer interface

S.hin()

\rightarrow u bit array \rightarrow array of size u

S.mst()

insert, update, search constant time

S.succ(x)

assume there $> 0 \in [0, u)$

S.pred(x)

find max, min, pred, succ

using the DCL mechanism

\Rightarrow summary where



IT clusters

the summarization of the bits
This structure is called a bit cluster

bit cluster

where a bit is true will

$\boxed{1} \boxed{0} \boxed{1} \boxed{0} \boxed{1} \boxed{0}$

$$T(u) = T(\sqrt{u}) + 1$$

$$\text{let } l = \log u \quad X(l) = T(2^l)$$

$$X(l) = T(2^l) = T(u) = T(\sqrt{u}) + 1$$

$$= T(2^{\frac{l}{2}}) + 1 = X\left(\frac{l}{2}\right) + 1$$

Using master theorem gives $X(l) = O(\log l)$ then

$$T(u) = O(\log \log u)$$

vEB Tree

Insert (V, x):

if $V.\min = \text{None}$: $V.\min = V.\max = x$
return

if $x < V.\min$: swap $x \leftrightarrow V.\min$

if $x > V.\max$: $V.\max = x$

if $V.\text{cluster}[\text{high}(x)].\min = \text{None}$:

 Insert ($V.\text{summary}, \text{high}(x)$)

 Insert ($V.\text{cluster}[\text{high}(x)], \text{low}(x)$)

Successor (V, x):

if $x < V.\min$: return $V.\min$

if $V.\text{cluster}[\text{high}(x)].\max > \text{low}(x)$

$j = \text{successor}(V.\text{cluster}[\text{high}(x)], \text{low}(x))$

 return $\text{index}(\text{high}(x), j)$

else

$i = \text{successor}(V.\text{summary}, \text{high}(x))$

 return $\text{index}(i, V.\text{cluster}[i].\min)$

Delete (V, x):

if $V.\min = \text{None}$: return

if $x = V.\min$:

$i = V.\text{summary}.\min$

 if $i = \text{None}$:

$V.\min = \text{None}$

 return

 else

$j = V.\text{cluster}[i].\min$

$x = V.\min = \text{index}(i, j)$

Predecessor (V, x)

if $x > V.\max$: return $V.\max$

if $V.\text{cluster}[\text{high}(x)].\min < \text{low}(x)$

$j = \text{predecessor}(V.\text{cluster}[\text{high}(x)], \text{low}(x))$

 return $\text{index}(\text{high}(x), j)$

else

 if $V.\text{summary}.\min \geq \text{high}(x)$:

 return $V.\min$

$i = \text{predecessor}(V.\text{summary}, \text{high}(x))$

 return $\text{index}(i, V.\text{cluster}[i].\max)$

Delete ($V.\text{cluster}[\text{high}(x)], \text{low}(x)$)

if $V.\text{cluster}[\text{high}(x)].\min = \text{None}$:

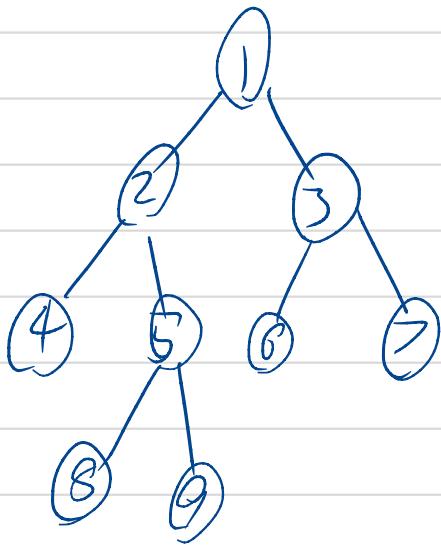
 Delete ($V.\text{summary}, \text{high}(x)$)

if $x = V.\max$:

 if $V.\text{summary}.\max = \text{None}$: $V.\max = V.\min$

 else $V.\max = \text{index}(V.\text{summary}.\max, V.\text{cluster}[V.\text{summary}.\max].\min)$

Least common Ancestor (LCA)



$$E = \{1, 2, 4, 3, 5, 8, 5, 9, 5, 2, 1, 3, 6, 3, 7, 3, 1\}$$

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

$$R = \{1, 2, 12, 3, 5, 13, 15, 1, 8\}$$

$$LCA(i, j) = E[RMQ_L(R[i], R[j])]$$

$RMQ \pm 1$

$\langle O(n), O(1) \rangle$ vs $RMQ \pm 1$

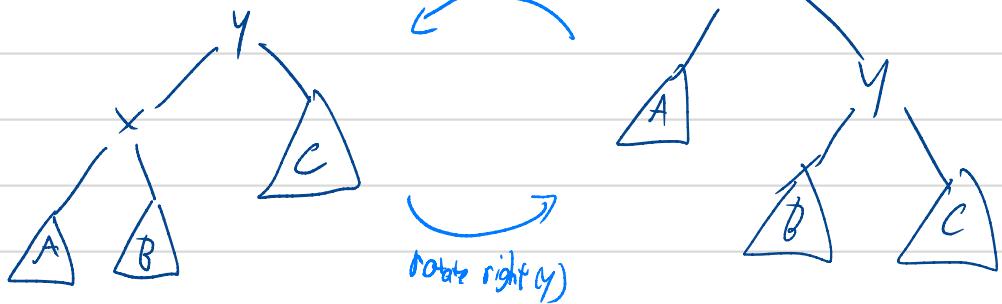
"U" A tree \rightarrow $RMQ \pm 1$ $\log n$ time precompute answer query $O(\log n)$ time

array A' , B' $\in \frac{n}{\log n}$

Splay Tree

Average $O(\log n)$

Tree Rotation



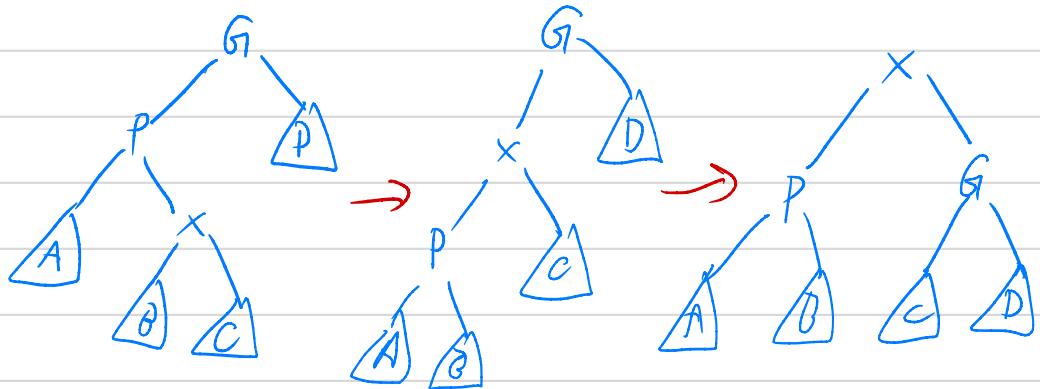
Splay Tree operation

1. find(x)

- insert x in BST
- search for node x
- splay (x)

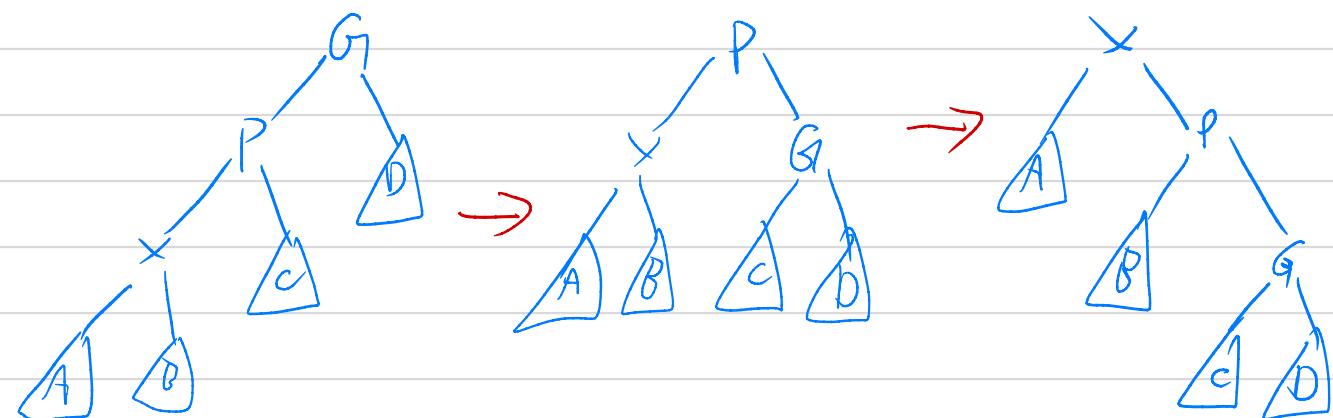
Splay (x) - x է այս տեղում գտնվում է

case 1. x է այս տեղում գտնվում է (այս տեղում գտնվում է այլ տեղում կամ առաջին հաջորդ տեղում է)

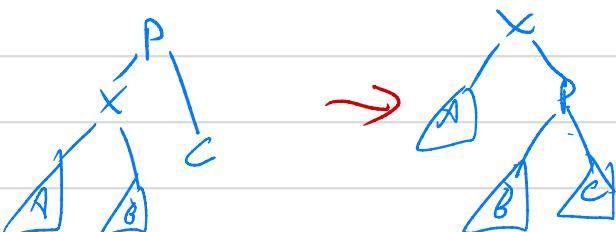


Case 2: x է այս տեղում գտնվում է (այս տեղում գտնվում է այլ տեղում կամ առաջին հաջորդ տեղում է)

zig-zig



case 3. x է այս տեղում գտնվում է առաջին հաջորդ տեղում



2. Min max

- minimized BST into splay min/max

3. remove(k)

- zigzag on splay tree

- if k is child of parent(k) splay (parent(k))

- if k is grandchild of parent(v), splay parent v of k

- if k is root of tree, move k to child of parent(k) and splay parent

splay parent root of tree

Dynamic Tree

Forest of trees

Operations

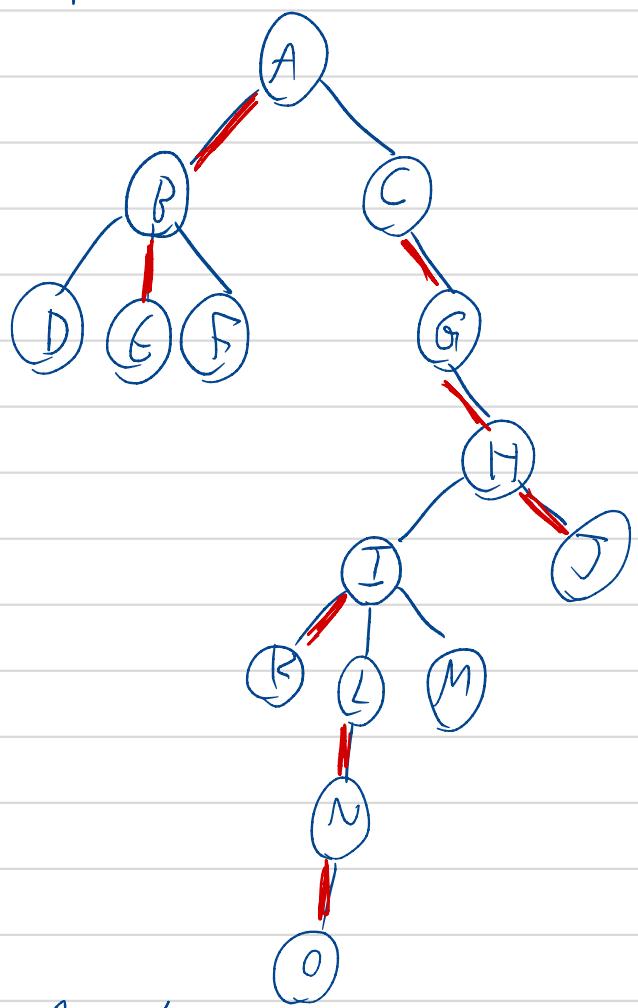
- Make-Tree() - return a new vertex in a single tree
- Link(v, w) - Make vertex v a new child of vertex w i.e. add edge (v, w) , assuming v is the root of its tree, v and w are node the tree
- CUT(v) - Delete edge between vertex v and its parent, parent(v) where v are not root
- Find-Root(v) - return the root of the tree that vertex v is in

Def of Link cut tree

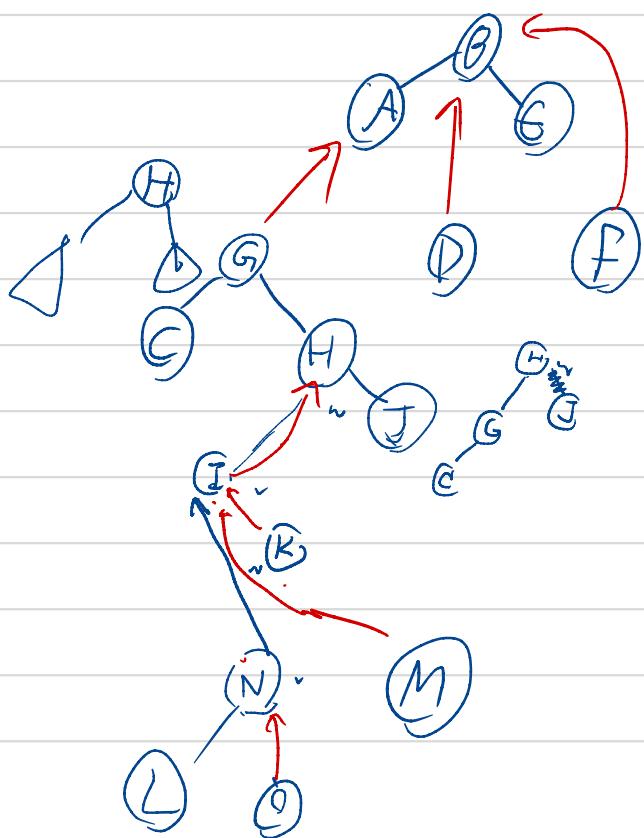
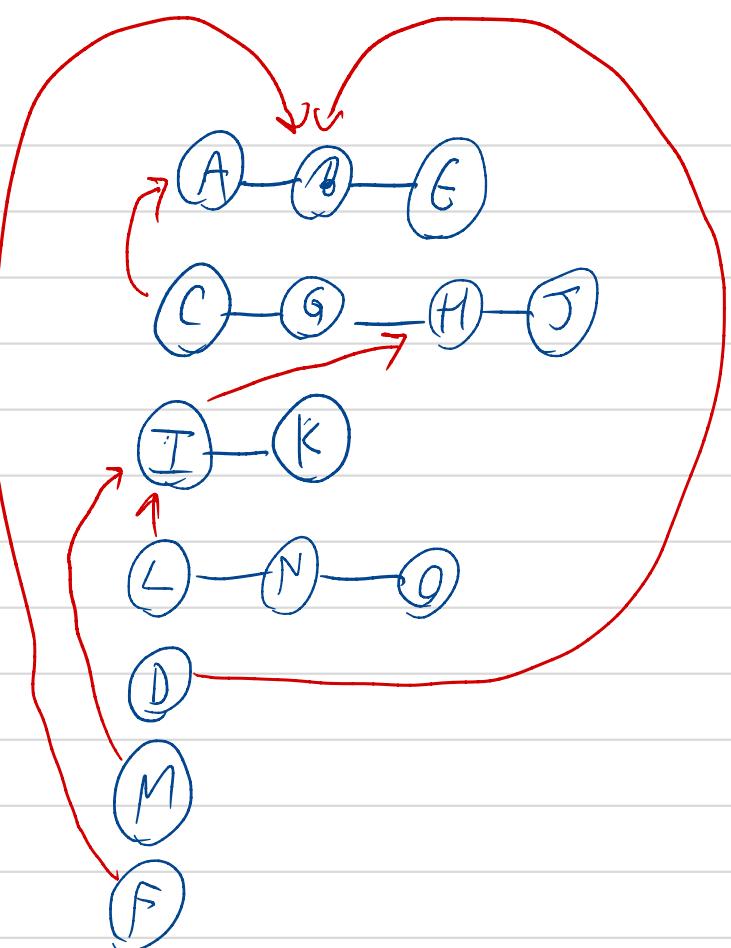
We say a vertex have been accessed if it was passed to any of the operations from above as an argument.

We call represented tree the abstract tree that the data structures represent.

Represented Tree

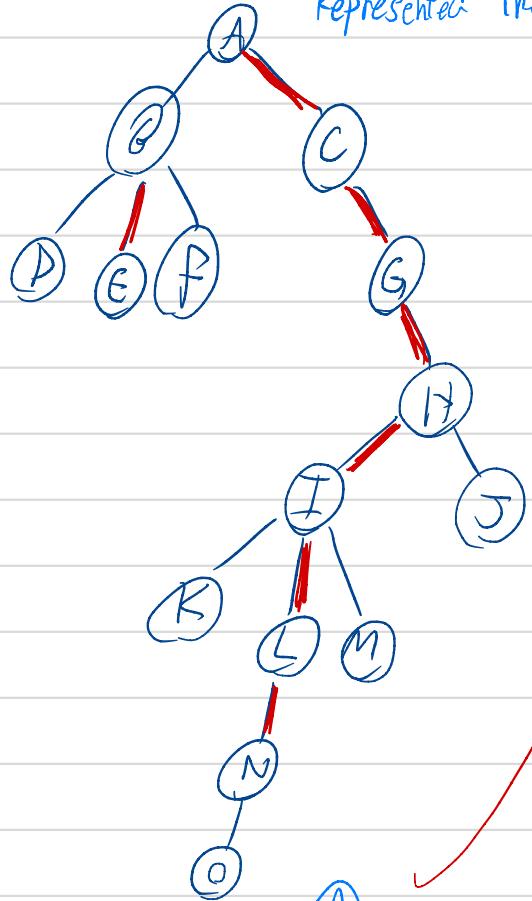


Auxiliary Tree

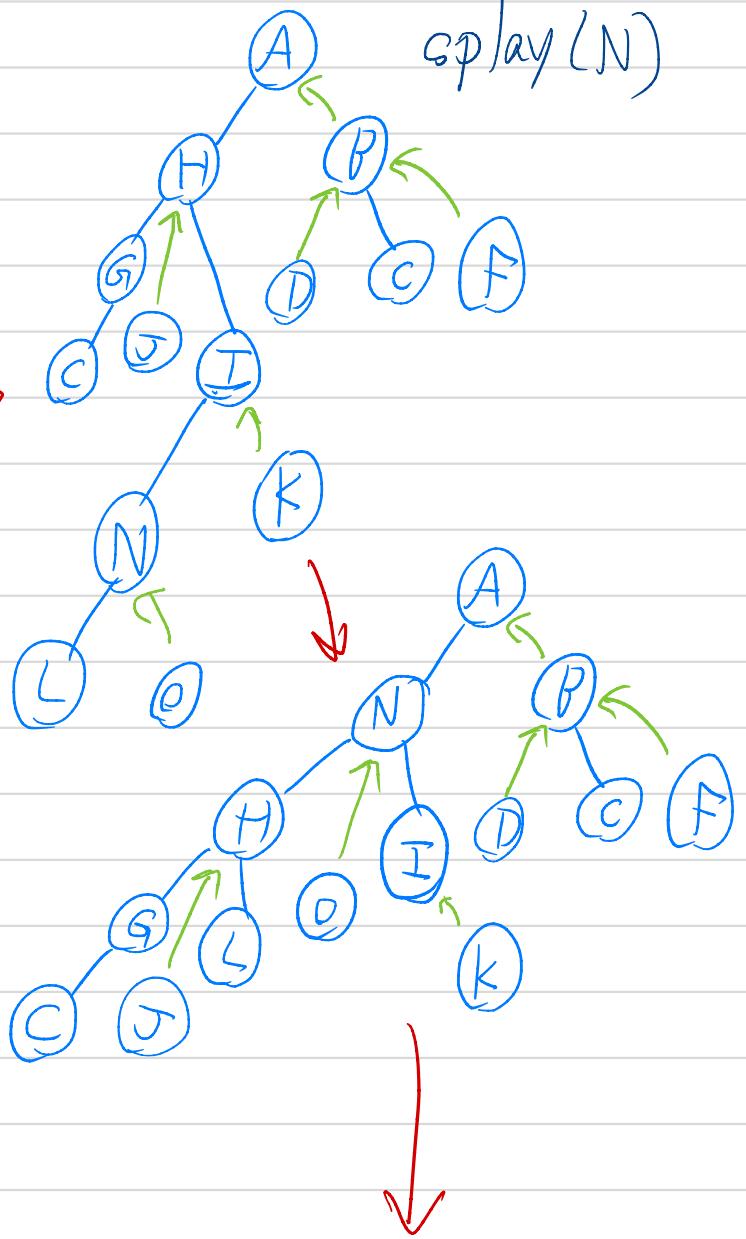


Access (N):

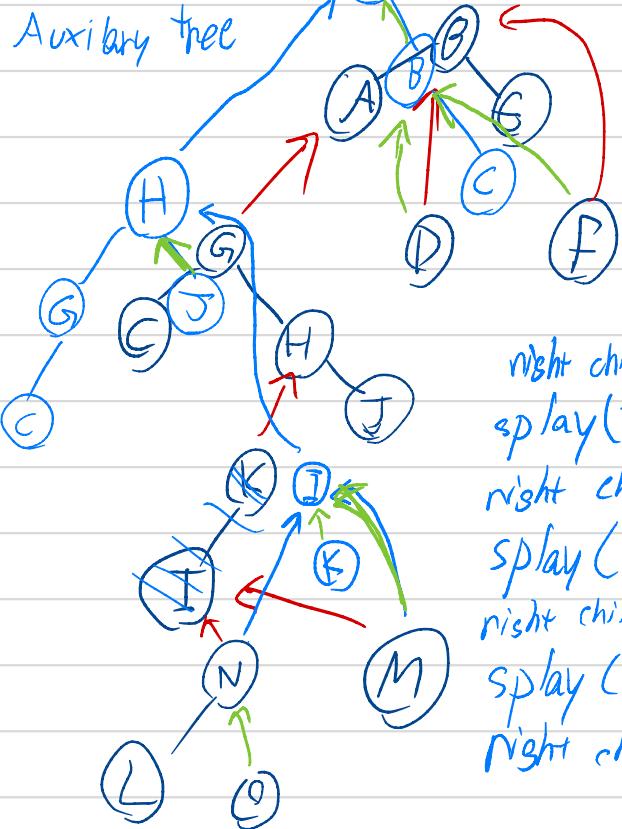
Represented Tree



splay (N)



Auxiliary tree

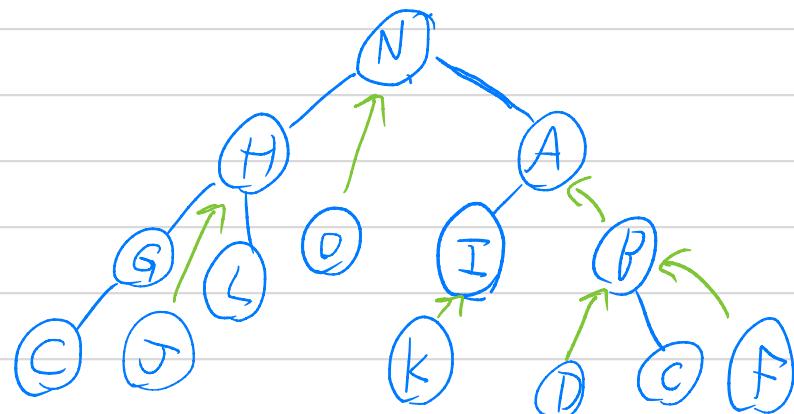


right child unlink
splay (I)

right child unlink
splay (H)

right child unlink
splay (A)

right child unlink
splay (N)



A amortized analysis

- worst-case Analysis
- Average-case Analysis
- Randomized Quick-Sort

Start with minimum cost operation in amortized analysis

Aggregate Method

Binary counter

operation-increment

init 0000...0

for increment m assignments the bit mask is given by powers of 2

$\begin{matrix} 2^3 & 2^2 & 2^1 & 2^0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \end{matrix}$

$$\text{cost of } i \text{ bit} = m + \frac{m}{2} + \frac{m}{4} + \dots$$

$$= m + \left[1 + \frac{1}{2} + \frac{1}{4} + \dots \right]$$

$= 2m$ ← upperbound since bit mask

Amortized cost of increment m assignments $\geq 2m = O(1)$ (amortized)

Accounting Method

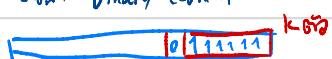
Allocate a fixed cost C_i to each assignment i so that cost of i assignment $\leq C_i$

Let c_i be the cost of i assignment \hat{C}_i is the amortized cost

$\hat{C}_i = c_i + \text{cost of previous assignments} - \text{cost of previous assignments}$

* If $c_i > \hat{C}_i$ then \hat{C}_i is negative

Worst case binary counter



Initial = 2nd last bit is 1

new value = 2nd last bit is 0

$$\text{Cost of } i = k + 1$$

Set 1 in bit 0 and 1 in bit 1 \rightarrow cost of 2 assignments $\leftarrow 1+1=2$ units

Set 1 in bit 1 and 0 in bit 2 \rightarrow cost of 2 assignments $\leftarrow 1+1=2$ units

$$\text{Amortized cost increment} = 0+0+0+\dots+2 = 2 \text{ units}$$

$k=5$

Potential Method

- ມະນາຄີ່ງສົດສະພາບກູ້ອັນໄວ້ Potential ວິວ D.S.
- ຜົນຂຶ້ນ $\Phi(D_t)$ ດີຈິນໄໝ Potential ວິວ Data Structure ຂອງ t
- amortized cost ວິວ operation ສ້າງ ຕະຫຼາມ $\text{cost}_{\text{real}} + \Delta\Phi$; $\Delta\Phi = (\Phi(D') - \Phi(D))$
- * $\Phi(D_t) \geq \Phi(D_0)$ ໂຕກໍ່ໄປ set $\Phi(D_0) = 0$ ຢືນຊະກິນວ່າ $\Phi(D_t) \geq 0$

ວິທີຈົນ Binary Counter

ກູ້ $\Phi(D_t)$ ສະແດງຈຳນວດ bit ຖ້າຕົວລີ່ມີ 1

$$\Phi(D_0) = 0, \quad \Phi(D_t) \geq 0 \quad \forall t$$

$$\begin{aligned} \text{amortized cost } \text{ວິວ increment} &= \text{cost}_{\text{real}} + \Delta\Phi \\ &= k+1 + (1-k) \\ &= 2 \end{aligned}$$

ວິທີຈົນ Dynamic arrays

$$S = \text{size}, \quad n = \# \text{element}$$

Ref array n ນີ້ມີຄວາມອະນຸຍາຍທີ່ຈຳເປັນ

$C = 3 \leftarrow \text{push_back}(k)$: ໃຫວ້າໃນ k ອີ່ມີ array

$C = 0 \leftarrow \text{Table doubling}$



• init $S=0, n=0 \rightarrow \boxed{1}$ ໃນ $n=0$

• push-back(k): ໃນ $n \Rightarrow$ table doubling
 $O(1) \rightarrow$ ດີວ້າ k ລູກອີ່ມ $n+1, n \leftarrow n+1$

- set n ມານວ່າສະບັບກົດຕົວຢ່າງ array ເຖິງຕະຫຼາມກົດຕົວ 2 ນີ້ \rightarrow cost ອະ 1 ນີ້ \rightarrow amortized $1+2=3$ ນີ້

- set n ມານວ່າ table doubling ມານ size $n \Rightarrow 2n$ ດັ່ງນີ້ n ນີ້ \rightarrow cost ອະ n ນີ້ \rightarrow amortized $n-n=0$ ນີ້

\therefore amortized cost $\text{ວິວ push_back} = 3+0=3$ ນີ້ $= O(1)$

$$\Phi(D_0) = 0, \quad \Phi(D_t) \geq 0, \quad \Phi(D) = 2n-s$$

$$n \geq \frac{1}{2}s \equiv 2n \geq s \equiv n-s \geq 0$$

amortized cost ວິວການກົດຕົວອີ່ມ

$$= \text{cost}_{\text{real}} + \Delta\Phi$$

$$= 1 + 2 = 3 \text{ ນີ້ }$$

$$\Phi(D) = 2n-s$$

$$\Phi(D') = 2(n+1)-s$$

$$= (2n-s)+2$$

$$\Delta\Phi = \Phi(D') - \Phi(D)$$

$$= [(2n-s)+2] - (2n-s) = 2$$

amortized cost $\text{ວິວການກົດຕົວ Table doubling}$

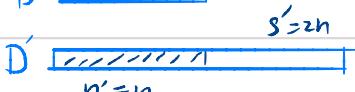
$$= \text{cost}_{\text{real}} + \Delta\Phi$$

$$= n-n = 0$$

$$\Phi(D) = 2n-s = 2n-n=n$$

$$\Phi(D') = 2n'-s'$$

$$= 2n-2n=0$$



- $\text{pop_back}() \Rightarrow O(1)$ amortized

$$\Phi(D) = \begin{cases} 2n - s & \text{if } n \geq \frac{1}{2}s \\ \frac{1}{2}s - n & \text{if } n < \frac{1}{2}s \end{cases}$$

Binary heap

$\forall u, \text{key}(u) \geq \text{key}(\text{parent}(u))$ | amortized

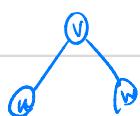
Cost vs insert = $O(\log n)$ | $O(\log n)$

Cost vs extractMin = $O(\log n)$ | $O(1)$

Decrease Key | $O(\log n)$ | $O(\log n)$

Delete | $O(\log n)$ | $O(1)$

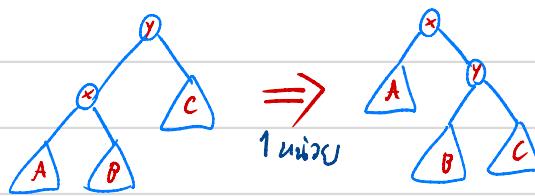
Splay Tree



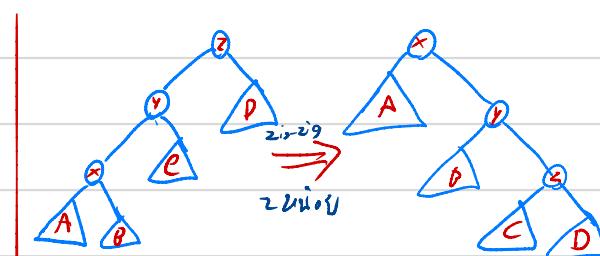
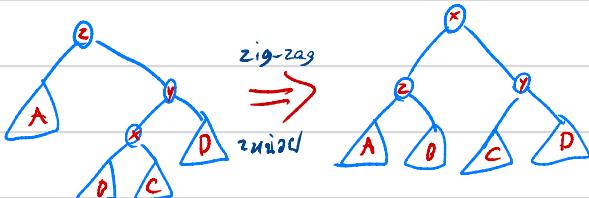
$u \leq v \leq w$

- operation
- Insert (v)
 - find (v)
 - delete (v)

single Rotation

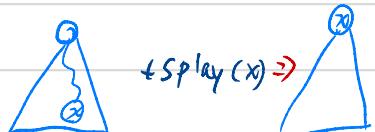


Double Rotation

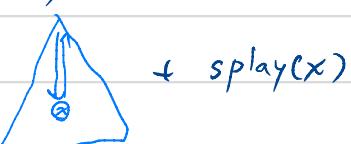


splay(x): performs double rotation in x (+ single rotation in x) until x is the root

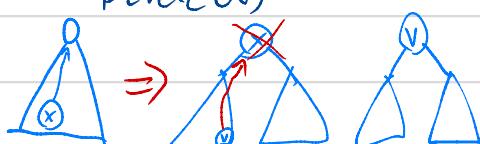
- Insert (x):



- find(x)



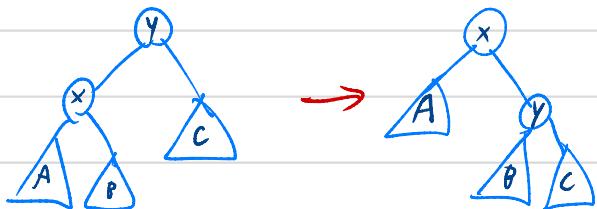
- Delete (x)



分析 amortized cost 与 splay

$$r(u) = \log(\text{size}(u)) \quad \Phi(T) = \sum_{u \in T} r(u) \quad \text{size}(u) = \# \text{node in subtree rooted at } u$$

① amortized cost vs single rotation



$$C_{\text{single}} = 1, \Delta \Phi = r'(x) + r(y) - r(x) - r(y)$$

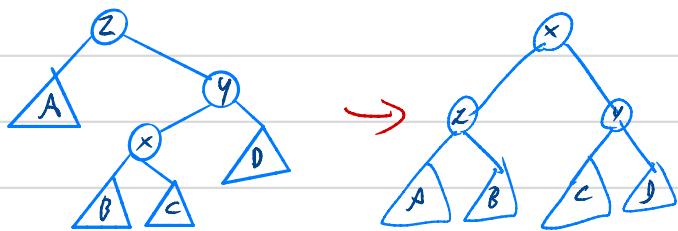
then $\text{size}(y) = \text{size}'(x)$

$$r(y) = r'(x)$$

$$\Delta \Phi = r'(y) - r(x) \leq r'(x) - r(x)$$

$$\hat{C}_{\text{single}} \leq 1 + r'(x) - r(x)$$

② amortized cost vs zig-zag



$$C_{\text{zig-zag}} = 2, \Delta \Phi = r'(x) + r(y) + r(z) - r(x) - r(y) - r(z)$$

1) $\text{size}(z) = \text{size}'(x)$, $r(z) = r'(x)$

2) $\text{size}(x) < \text{size}(y)$, $r(x) < r(y)$

$$\Delta \Phi \leq r(y) + r(x) - r(x) - r(y)$$

$$\leq \log(\text{size}(y)) + \log(\text{size}(z)) - 2r(x)$$

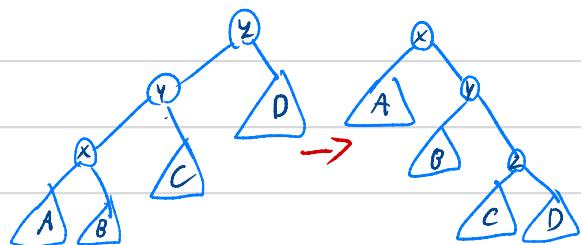
$$\leq 2\log\left(\frac{\text{size}'(y) + \text{size}'(z)}{2}\right) - 2r(x)$$

$$\leq 2\log\left(\frac{\text{size}'(x)}{2}\right) - 2r(x)$$

$$= 2r'(x) - 2r(x) - 2$$

$$\hat{C}_{\text{zig-zag}} = C_{\text{zig-zag}} + \Delta \Phi = 2 + 2r'(x) - 2r(x) - 2 = 2r'(x) - 2r(x)$$

③ amortized-cost vs zig-zig



$$C_{\text{zig-zig}} = 2, \Delta \Phi = r'(x) + r'(y) + r'(z) - r(x) - r(y) - r(z)$$

$$= r'(y) + r'(z) - r(x) - r(y)$$

$$\leq r'(y) + r'(z) - 2r(x)$$

$$\leq r'(y) + r'(z) + r(x) - 3r(x)$$

$$\leq r'(y) + 2\log\left(\frac{\text{size}'(z) + \text{size}'(x)}{2}\right) - 3r(x)$$

$$\leq r'(y) + 2r'(x) - 3r(x) - 2$$

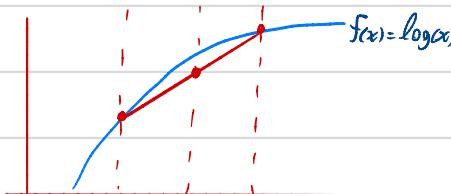
$$\leq 3r'(x) - 3r(x) - 2$$

$$\hat{C}_{\text{zig-zig}} = C_{\text{zig-zig}} + \Delta \Phi$$

$$\leq 2 + 3r'(x) - 3r(x) - 2$$

$$\leq 3(r'(x) - r(x))$$

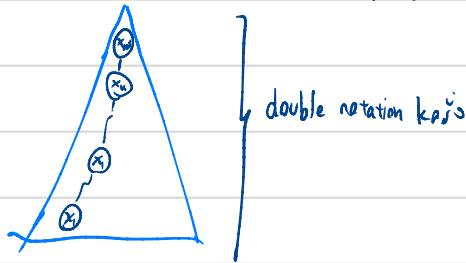
\log is concave function



$$\frac{\log a + \log b}{2} \leq \log\left(\frac{a+b}{2}\right)$$

$$\begin{aligned} \hat{C}_{\text{single}} &\leq 1 + r'(x) - r(x) \leq 1 + 3(r'(x) - r(x)) \\ \hat{C}_{\text{zigzag}} &\leq 2(r'(x) - r(x)) \leq 3(r'(x) - r(x)) \\ \hat{C}_{\text{bigzag}} &\leq 3(r'(x) - r(x)) \end{aligned}$$

\therefore amortized cost ของ splay(x) = ผลรวม amortized cost ของ notation ที่ใช้ในการ splay



$$\begin{aligned} \hat{C} &\leq 3(r(x_1) - r(x_0)) + 3(r(x_2) - r(x_1)) + \dots + 3(r(x_k) - r(x_{k-1})) + 1 + 3(r(x_{\text{root}}) - r(x_0)) \\ &\leq 1 + 3(r(x_{\text{root}}) - r(x_0)) \\ \hat{C}_{\text{splay}} &\leq 1 + 3(r(x_{\text{root}}) - r(x_0)) \\ &\leq 1 + \log(n) \\ &= O(\log n) \end{aligned}$$

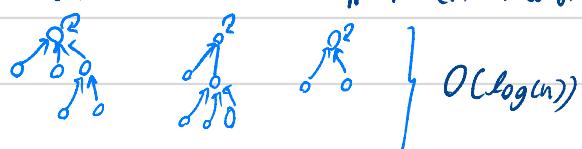
Union-Find (Disjoint-Set)

- initial(n)



- Union(i, j): รวมตัวกัน i และ j ให้เป็น 1 ตัวเดียว \Rightarrow 1 ตัวเดียวที่มีค่ามากกว่าเดิม

- Find(i): return key ของตัวที่ i อยู่ในชั้นเดียวกับ j , i, j อยู่ในชั้นเดียวกัน



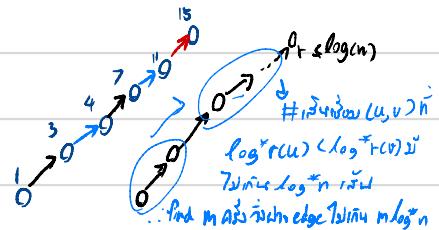
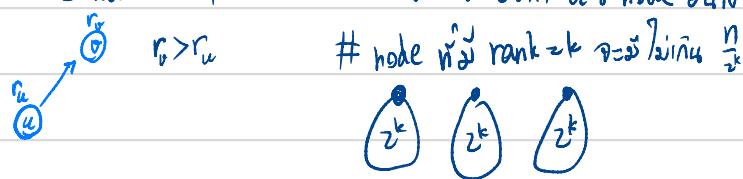
Union by size
Union by rank

$\log^* n$	n
0	1
1	2
2	3, 4
3	5, 6, 7, ..., 16
4	17, 18, ..., 65536
5	65537, ..., 2 ⁶⁵⁵³⁶

amortized cost ของ $\text{Find}(i)$ $\leq O(\log^* n)$

เมื่อ $\log^* n$ ต้องมากกว่า k ถ้า $\log^* n \leq k$ $\log \log \dots \log n \leq 1$

- สำหรับ node u กับ rank ของ u คือ จำนวนหลังจาก u ถึง node ที่เป็น parent



พิจารณากรณีต่อไปนี้ Find(m) ถ้า m เป็น偶数 ให้พิจารณาหา $\frac{m}{2}$ จำนวนเส้นเชื่อมที่จะมีเพิ่มขึ้นทีละหนึ่งตัว

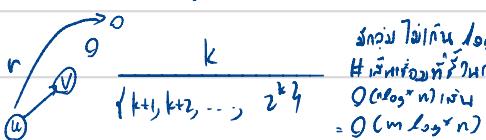
จำนวนเส้นเชื่อม

= # จำนวนตัวอย่าง (u, v) ที่ v เป็น root \Rightarrow จำนวนตัวอย่าง

จำนวนตัวอย่าง (u, v) ที่ $\log^* r(u) < \log^* r(v)$

จำนวนตัวอย่าง (u, v) ที่ $\log^* r(u) = \log^* r(v)$

จำนวนตัวอย่าง (u, v) ที่ $\log^* r(u) = \log^* r(v)$ แต่ v ไม่เป็น root



\therefore Edge รวมเข้า一起去

Find จำนวน $m > n$ ครั้ง

จำนวน $O(m \log^* n)$ เส้น

\therefore amortized cost ของ Find = $O(\log^* n)$

Group Theory - abelian

(S, \oplus)

1.) closure

$$\forall a, b \in S \Rightarrow a \oplus b \in S$$

2) Identity

$$\exists e \in S \Rightarrow \forall a \in S, a \oplus e = a$$

3) Associative

$$a, b, c \in S \quad a \oplus (b \oplus c) = (a \oplus b) \oplus c$$

4) Inverse

$$a \in S \quad \exists b \text{ unique one to one}$$

$$a \oplus b = e$$

Ex $(\mathbb{Z}_n, +)$

$$\forall a, b, c \in \mathbb{Z}_n \Rightarrow a +_n b \in \mathbb{Z}_n$$

$$a + e = a \Rightarrow a + 0 = a$$

$$a + (b + c) \sim (a + b) + c$$

$$a + (n-a)_n = n_n = 0 \Rightarrow 0 < n-a < n$$

$$\mathbb{Z}_n^* = \{ [a]_n \in \mathbb{Z}_n : \gcd(a, n) = 1 \} \quad \text{prime prime}$$

$$\mathbb{Z}_{15}^* = \{ 1, 2, 4, 7, 8, 11, 13, 14 \}$$

$$\phi(n) = n \prod_{\substack{p \mid n \\ p \in P}} \left(1 - \frac{1}{p}\right) = |\mathbb{Z}_n^*|$$

NP - Complete

Solve optimization

- $\max_{\bar{N}}$

- objective function

if k in range

Goal minimize $\max_{\bar{N}}$ in \bar{N} such that objective function $\leq k$

binary search

Solve minimization (Decision problem)

Get input x , some parameter

Answer \Rightarrow True/No Yes/No

True if A returns $x \in X$

$\neg A(x)$ means Yes

Shortest Path

Given directed graph $G = (V, E)$, $s \in V$, $t \in V$, \leq $E \in E$, $d \in k$

Answer: \exists path $s \rightarrow t$ in G in cost $\leq d$ or \neg

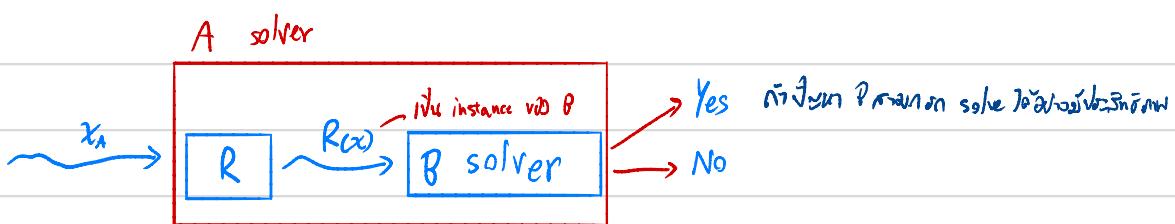
Note: If A is algorithm A and B is algorithm B then $A \leq_p B$ if A runs in polynomial time

Polynomial time Reduction (Conversion)



Given Decision Problem $A \leq_p B$ if algorithm R takes instance x of A and return instance y of B in

$\forall x \in X_A$; $A(x)$ means Yes iff $R(x)$ means Yes. (algo R is polynomial time algorithm)



Given A has a polynomial time reduction to B then B is at least as hard as A for the poly-time
reductions. Given A has a polynomial time reduction to B then B is at least as hard as A for the poly-time
 $A \leq_p B$

ថាមរយៈ Independent Set (IS)

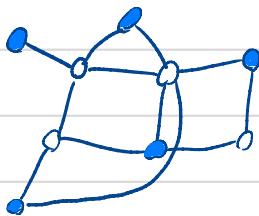
ឲ្យឱ្យ undirected graph $G = (V, E)$, និងការសរុបតូច k , G តើ independent set នូវការសរុបតូច k គឺជាដី?

independent set តើ $S \subseteq V$ ដូចខាងក្រោម $u, v \in S$ $(u, v) \notin E$

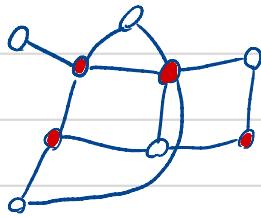
ថាមរយៈ Vertex Cover (VC)

ឲ្យឱ្យ undirected graph $G = (V, E)$, int k G តើ Vertex Cover នូវការសរុបតូច k នឹងតូច?

Vertex Cover តើ $S \subseteq V$ ដូចខាងក្រោម $\forall (u, v) \in E, u \in S \vee v \in S$



~IS នូវ 5 node



~VC នូវ 4 node

Theorem: សំខាន់ស្ថិករបស់ $G = (V, E)$, $S \subseteq V$ នឹង independent set iff $V-S$ នឹង vertex cover

proof : 1) ពី S នឹង independent set $\rightarrow V-S$ នឹង vertex cover

2) ពី $V-S$ នឹង vertex cover $\rightarrow S$ នឹង independent set

1. ពី S នឹង independent set ដឹងថាប្រាក់នៃកំណត់ទីនៅ $(u, v) \in E$ សម្រាប់ u និង v ទូទៅជាជួយ S អំពីលើកនៃកំណត់ទីនៅ

ទេន័ែម ទាំងពីរ និងទាំងពីរ នឹង $V-S$ $\therefore V-S$ នឹង vertex cover

2. ពី S ជាដី independent set ដឹងថាប្រាក់នៃកំណត់ទីនៅ $(u, v) \in E$ ត្រូវបាន u ឬ v និង u ឬ v នឹង S

ប៉ុណ្ណោះ $V-S$ នឹង vertex cover (ដឹងថា $(u, v) \in E$) $\therefore V-S$ ជាដី vertex cover

Theorem Independent Set \leq_p Vertex Cover.

proof ទម្រង់សេចក្តីផ្តើម reduce ឲ្យរយៈ IS ទៅវា VC តួន្ផៅ

$G = (V, E)$
 k



$G = (V, E)$
 $n-k$

instance នៃ IS

instance នៃ VC

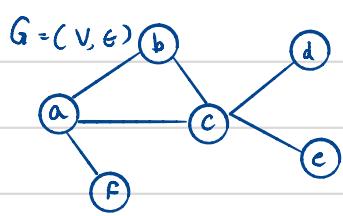
ຈິງກາ Set Cover

ຖີ່ $\bigcup S = \{e_1, \dots, e_n\}$, $S_1, \dots, S_k \subseteq \bigcup S$, ບັນດາໃຫຍ່ ຂະໜາມການ
ເລືອງ ພົມມາກ ເຊົາກາງຕາ S_i ທັງນີ້ ກ່ອນ ສູນພວນອັນທີ ເລືອງໄດ້ຈຳນົດ $\bigcup S$ ໄກສະແດງ

Theorem : Vertex Cover \leq_p Set Cover

proof : ໂກງະໂຮງ ຮັດຂອງ $VC \rightarrow SC$

ສໍາຜົນ instance ວົດ VC , $G = (V, E)$ ໂກງະໂຮງ ຈີນ instance VCD



$$\bigcup E = E$$

$$S_a = \{(a, b), (a, c), (a, f)\}$$

$$S_b = \{(a, b), (b, c)\}$$

⋮

$$\bigcup E = E$$

- ສໍາຜົນ node $u \in V$

ສໍາຜົນ $S_u = \{e \in E : e \text{ ລັບຕິດກັບ } u\}$

- k (ຕົວລີ່ມ)

- ໄກສະວ່າ ສໍາຜົນ instance ວົດ VC ຖ້າວັນ Yes ແກ້ວຂຶ້ນ instance ວົດ SC ຖ້າວັນ Yes ດັວດເກີນກັນ

- ໄກສະວ່າ ສໍາຜົນ instance ວົດ VC ຖ້າວັນ No ແກ້ວຂຶ້ນ instance ວົດ SC ຖ້າວັນ No ດັວນ

↪ ໄກສະວ່າ SC ອະນຸຍາຍພລາກວນ Yes ແກ້ວຂຶ້ນ instance ວົດ VC ອະນຸຍາຍພລາກວນ Yes ຕ້ອນ

ຈິງກາ 3SAT

ຖີ່ boolean x_1, \dots, x_n C_1, \dots, C_n ອີ່ clause $C_i = (u \vee v \vee w)$ ໃນີ້ u, v, w ຕີ່ x_i ແລ້ວ \bar{x}_i

ຕາມ: ພົມມາກ assign ຄ່າຄວາມຕິດຕັ້ງທີ່ x_1, \dots, x_n ທີ່ທີ່ C_1, \dots, C_n ເປົ້າອີ່ງພວດກັນທີ່ຂອງເຊື້ອໄວ

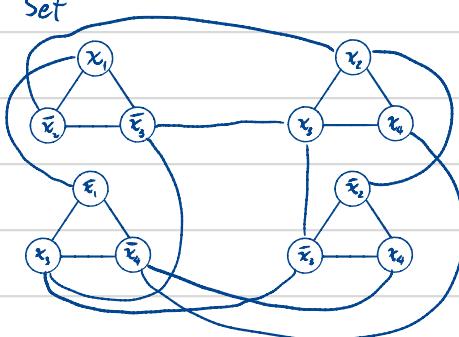
Theorem 3SAT \leq_p Independent Set

$$C_1 = (x_1 \vee \bar{x}_2 \vee \bar{x}_3)$$

$$C_2 = (x_2 \vee x_3 \vee x_4)$$

$$C_3 = (\bar{x}_1 \vee x_3 \vee \bar{x}_4)$$

$$C_4 = (\bar{x}_2 \vee \bar{x}_3 \vee x_4)$$



3SAT \leq_p IndependentSet \equiv_p VertexCover \leq_p SetCover

คลาส P คือชั้นของปัญหาที่หาคำตอบได้ใน polynomial-time ของขนาด input

ลักษณะปัญหา $x \in P$ ให้ตัวอ่อน $A_x(s) = \text{Yes}$ iff $s \in X$ และ A_x ใช้เวลาเป็น polynomial time

คลาส NP คือ ชั้นของปัญหาที่สักดิ์ตรวจสอบคำตอบได้ใน polynomial-time

พิจารณา [การตรวจสอบคำตอบ] ให้ X เป็น Decision problem เช่น กว่าจะว่า อิเล็กทรอนิกส์ V ผู้ใด วัดอุณหภูมิห้องที่ตรวจสอบคำตอบของ X

เมื่อ - ถ้าครับ $s \in X$ จะว่า string t ตอบนั้น 1 ถึงที่ $V(s, t)$ ต้อง Yes

- ถ้าครับ $s \notin X$ สำหรับ string t ใดๆ $V(s, t)$ ต้อง No

3SAT $s \in 3SAT$ ก็ต่อเมื่อ s คือ string ที่ represent instance ของปัญหา 3SAT ที่ซึ่งแต่ละเงื่อนไข clause ใน s จะเป็นที่หนึ่ง

Verifier von 3SAT

รับ s เป็น input ของ 3SAT, รับ string t

if t represent assignment ของ x_1, \dots, x_n ให้ assign ให้ x_1, \dots, x_n ใน s ตามที่ t ระบุ ยัง clause ใน s ไม่เป็นที่หนึ่ง

Return Yes [Accept]

else Return No [Reject] $\therefore 3SAT \in \text{NP}$

Verifier von Independent Set

รับ G, k , รับ t

if t represent set ของ node ของ V ที่รวมกันเป็น k ไม่เก็บ independent set

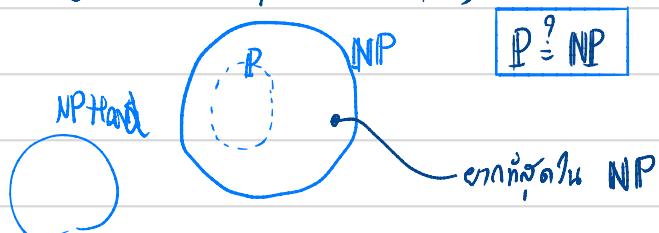
Return Yes

else Return No

Theorem $P \subseteq \text{NP}$

proof ให้ $x \in P$ ให้ A 为 poly-time algo ที่ solve X

Verifier รับ s และ t ให้ Return $A(s)$



พิจารณา ปัญหา $X \in \text{NP}$ ที่ NP-complete ต้องมีอย่างน้อย 1 ปัญหา $Y \in \text{NP}$ ที่ $Y \leq_p X$

NP-complete ปัญหานี้คือ CircuitSAT

CircuitSAT \leq_p 3SAT \leq_p IS $=_p$ VC \leq_p SC

การพิสูจน์ฝั่งปัญหา X เป็น NP-complete ทำได้ดังนี้

1. แสดงว่า $X \in \text{NP}$

2. แสดงว่า $Y \leq_p X$ สำหรับทุกปัญหา $Y \in \text{NP}$ ที่เป็น NP-complete

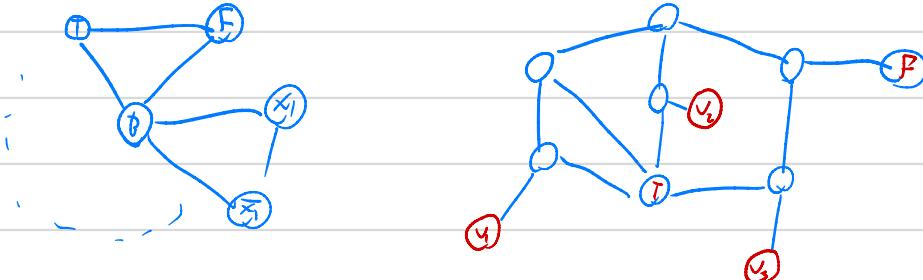
ปัญหา 3-color

ให้ undirected graph $G = (V, E)$ ตาม: สามารถจะบานสีในชุดเดียว 3 สีได้หรือไม่?

Theorem 3 Color is NP-complete

proof ปี 1970 แสดงว่า 3-color $\in \text{NP}$, ลักษณะว่าตัวมี certificate เป็นการระบุสีที่ให้ต่อ node ใดก็ได้ 3 ตัว ตรวจสอบว่า 3 ตัวนี้ไม่ใช้สีเดียวกัน กระบวนการตรวจสอบคือต้อง check ว่าแต่ละ node ไม่ใช้สีเดียวกัน ทั้งหมด 3 ตัว

ต่อไปจะแสดงว่า 3-color เป็น NP-hard (\exists NP-Complete problem X ที่ $X \leq_p 3\text{color}$)

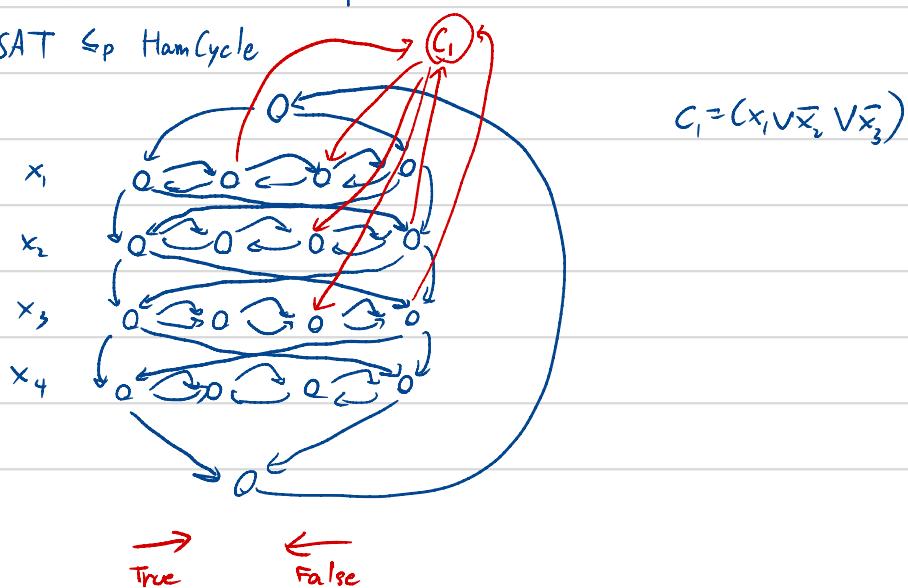


ปัญหา Hamiltonian Cycle

ให้ directed graph $G = (V, E)$, ถาม: มี cycle 9 ใน G ที่ผ่านครบทุก node node ละ 1 ครั้งเท่านั้นหรือไม่?

Theorem HamCycle is NP-complete

3SAT \leq_p HamCycle



o

TSP (Travelling Salesman Problem)

gū complete directed graph $G = (V, E)$ gū $c_e \forall e \in E$, gūn k

ərə: ər hamiltonian cycle n̄ cost ərələrən k uşələrəq

TSP is NP-complete

edge n̄ ər 1, edge n̄ ər 2 z output cost $k = N$