

Data Structure

- PART FINAL -



Patiwit Wuttisarnwattana

ผู้สอน 21 ชั่วโมง ·

แนวข้อสอบ มีทั้งหมด 24 ข้อ

- ข้อ 1: Java Programming of Find operation (ที่ไม่มีใน Quiz 2) (10คะแนน)
 - ข้อ 2: Binary Search Tree Delete (15 คะแนน)
 - ข้อ 3: Java Programming of Node Rotations (20 คะแนน)
 - ข้อ 4-5: AVL Rebalancing, Balance Factor (20 คะแนน)
 - ข้อ 6: AVL Tree Merging (10 คะแนน)
 - ข้อ 7-9: Splay Tree operations (30 คะแนน)
 - ข้อ 10-11: B-Tree Insert and Delete (35 คะแนน)
 - ข้อ 12: Binary Heap Implementation using Array (15 คะแนน)
 - ข้อ 13: Priority Queue Implementations (20 คะแนน)
 - ข้อ 14: Stock Market Simulation using Binary Heaps (20 คะแนน)
 - ข้อ 15-16: Separate Chaining, Linear Probing, Quadratic Probing, Hash Table Basics (35 คะแนน)
 - ข้อ 17-19: Hash Table Insert and Delete for Hexadecimal Keys, Open Addressing, Collision Analysis (30 คะแนน)
 - ข้อ 20-21: Runtime Analysis of Graph Algorithms (20 คะแนน)
 - ข้อ 22: Graph Representations (20 คะแนน)
 - ข้อ 23: Java Programming of finding the shortest route (20 คะแนน)
 - ข้อ 24: Bonus (25 คะแนน)
- นศ สามารถจดข้อความเข้าได้ 1 กระดาษ A4 และสามารถนำกระดาษคิดเลขเข้าได้อีก 3 แผ่น รวม 4 แผ่นครับ
- ข้อสอบจะมีการตระหนักรูปด้วย เนื้อหาเยอะมาก ไปซ้อมมาด้วยครับ
- ข้อสอบมี Java Programming แล้ว 3 ข้อครับ
- ขอให้โชคดีทุกคนครับ

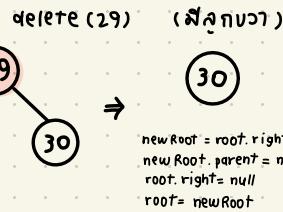
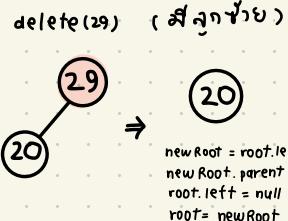
Programming

- BST Find Operation
- AVL Node Rotation
- Graph Find shortest route

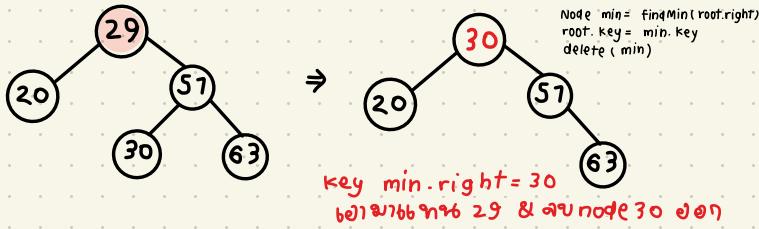
For the rest
= Understanding & Draw

BST Delete Operation

1. Empty Tree } ERROR
 2. Key not found }
 3. Root មានវិជ្ជាគារ child "លប់តែងលើ"
Delete (29) (29) → ទូរសព្ទ root = null



5. Root ដែល child 2 នៅ
"មិន key min. right តាមអាជីវកិច្ច key នៃ root
នៅក្នុង Node min. right គឺ"



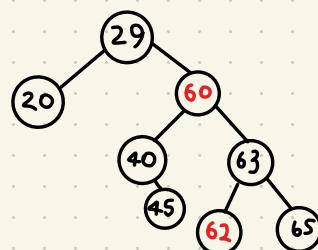
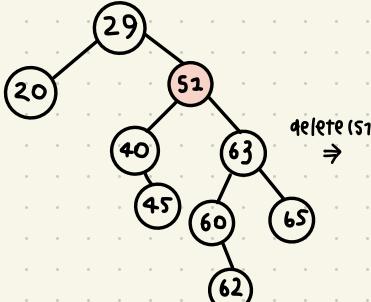
6. Node Կամ child "սեղմ"



7. Node վեց child ենթագլուխ "promote նպատակ" առաջարկություն



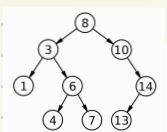
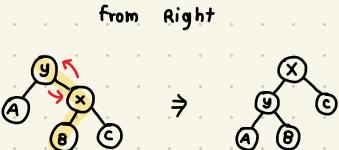
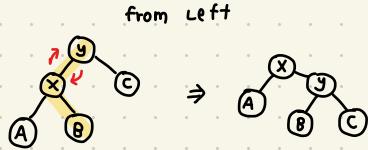
8. Node ມີ child 2 ຕອນ
“ເຊົາ key min.right ພາໄຫວ່າ key vo) Node
ໄລ້ຈະນີ້ Node min.right oon”



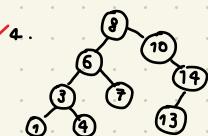
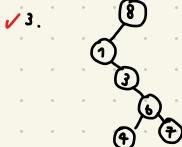
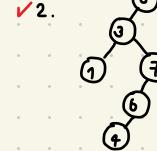
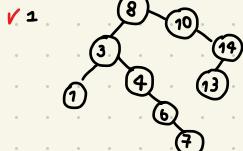
Rotation concept នៅក្នុង AVL មិន

Single Rotation

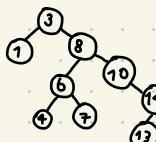
"គឺការចូលដោលទិន្នន័យនៃ parent"



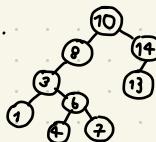
- SingleRotateFromLeft(6)
- SingleRotateFromRight(6)
- SingleRotateFromLeft(3)
- SingleRotateFromRight(3)
- SingleRotateFromLeft(8)
- SingleRotateFromRight(8)



✓ 5.



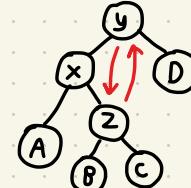
✓ 6.



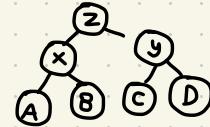
Double Rotation

→ Inner case ឬចាប់ផ្តើមជាមួយ

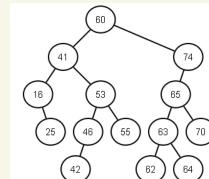
107 = ពីរ



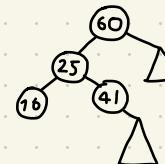
⇒



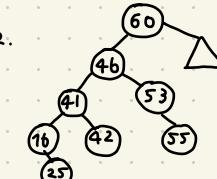
- DoubleRotateFromInnerLeft(41)
- DoubleRotateFromInnerRight(41)
- DoubleRotateFromInnerLeft(60)
- DoubleRotateFromInnerRight(60)
- DoubleRotateFromOuterLeft(60)



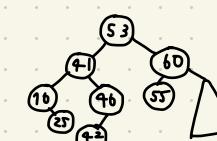
1.



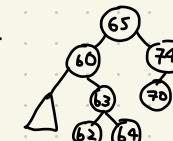
2.



3.



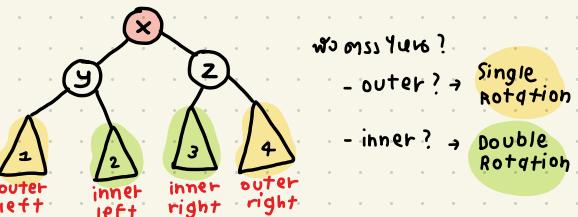
4.



Apply Rotation for Rebalancing!

ພົບຕາວອງ ຄະນະ AVL

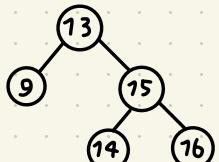
- what if node x ຝຟ?



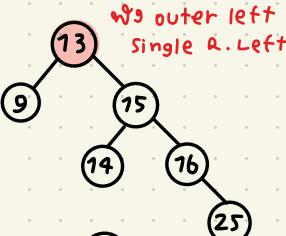
Insert & Delete

→ ຍັງ node → ພ່ມເຫດຢູ່ກີ່ງ root ຕ່າງໆ unbalanced? : { rebalance } : { 1ຍ້າ parent }

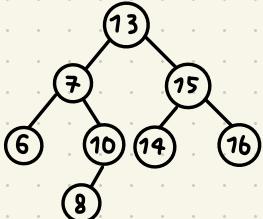
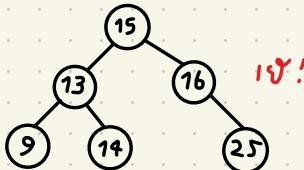
Ex.



insert(25)
⇒

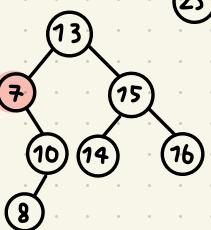


Single R.
from Left
⇒

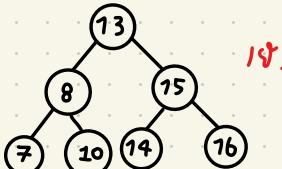


delete(6)
⇒

wົງ
inner
Right



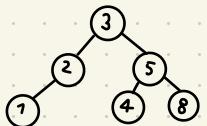
Double R.
from Right
⇒



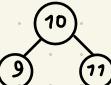
AVL Merge

- ເຊື້ອກໍານົງ mergeable ພົບ?
 - $\max(R_1) < \min(R_2)$
- $|R_1 - R_2| \leq 1$ Merge ເລຍ!

Ex

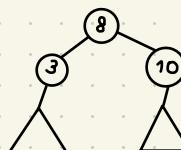


merge



⇒

10 8 ສາມເປົ້າrootຫຸ້ນ



- else → ອຸປ່ານ R_1 , ນັ້ນ R_2 ສູງກວ່າ
 - R_2 ສູງກວ່າ \Rightarrow ທີ່ເຖິງ $R_1.right$ ທີ່ $|R_1.right - R_2| \leq 1$ ມາ merge ດີຍ
 - R_2 ສູງກວ່າ \Rightarrow ທີ່ເຖິງ $R_2.left$ ທີ່ $|R_2.left - R_1| \leq 1$ ມາ merge ດັງ

Splay Tree

1985 By Sleator & Tarjan

- Node មុនទៀតខ្លះ = ចាយអារ៉ាប់ចង់បន្ត
- ផលកប្រភេទទិន្នន័យ ទីន្នន័យ

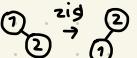
"ការ zig" ~ rotation



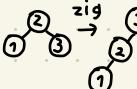
"Insert" ឈរឲ្យនៃ BST នៃក្នុង splay រាងកំroot

ឈរឲ្យ = 0(1) នឹងកំ 0(n)

Ex. ① insert(2)
⇒



② insert(3)
⇒



pro & cons.
+ ឈរឲ្យ = 0(log N)
- ឈរឲ្យនៃ AVL
- ស្របមួលឱ្យបែកបង្ហាញ

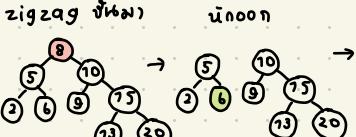
"Delete"

splay រាងកំroot → ឱ្យកូនភ្លាម Splay max/left រាងកំroot នៃកូន

Ex.

delete(8)
⇒

zigzag នៅក្បាស

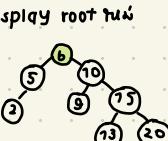


ឱ្យកូន

→

ឱ្យកូន

splay root នៃកូន



B-Tree

ការងារ $\log N$, balance តាមតាមទេ, hard - drive

- 1 Node ដើម្បីស្ថានលាយ key (min) M/2 ប៉ុណ្ណោះ - 1
(max) M - 1

- 1 Node ដើម្បីស្ថានលាយកន្លែង (min) M/2 ប៉ុណ្ណោះ
(max) M

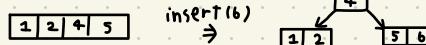
- Node អីមី k Keys គឺដឹងត្រូវ k+1 លាយនៅលើ

គាមត្រួត តួន 0($\log_{M/2} N$)

"Insert" ឈរឲ្យ node តិចនៅមី? → មី : មី, លើ

→ តិចមី : 10. node តារាងការងារបានបង្ហាញ

Ex.

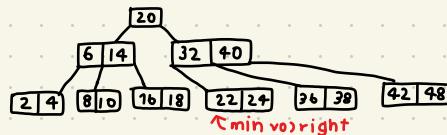


"Delete"

Key ឈូនីឱ្យបែក \rightarrow ស្រប ឱ្យកូនដែល underflow មី?

Key ឈូនីឱ្យបែក \rightarrow 10. min val right ពីកូនមាត្រាខាងក្រោមខាងក្រោម underflow

Ex.



↑ min val right

delete(20)

- 10. 22 គាន់ក្រោម
→ underflow ប៉ុណ្ណោះទៅ
= 6បុ



underflow = 6បុ

→

underflow = 6បុ

Priority Queue

Q. ស្ថាមារកសង្គម priQ. និត្តធម៌ប៉ារ៉ា

Ans.

1. multiple queue

- មែនជាផ្លូវការ នៅក្នុងការសង្គម
- នៅក្នុងការសង្គមមិនមែន priority ដើម្បី
- push obj. នៅលើព្យូរុណី ដើម្បីការសង្គមpriority
- top, pop នៅក្នុងក្នុងព្យូរុណីការសង្គមpriority ខ្លាំងនៅ

ចំណាំ : ផែនក្នុងព្យូរុណី
ទំនួរ : និង priority នៃក្នុង

2. AVL Tree

- នៅក្នុង node មិនមែន priority នៅក្នុង
- insert នឹងបញ្ជាការ priority ក្នុងប៊ូន
- top, pop នឹងចាប់បានក្នុងក្នុងសំណួល

ចំណាំ : ឈប់ priority នឹងប៊ូនក្នុងមាតិការងារក្នុងក្នុង

ទំនួរ : ចំណាំ មិនបានគ្មាន ទូទៅលទេនៅលើក្នុងក្នុង

3. Heap

- Binary Tree ដើម្បី root មិនមែន \leq key នៃ subtree នៅក្នុង
- នូវក្នុង subtree កិច្ចនូវ Binary Min Heap តួនាទី

ចំណាំ : ក្នុងក្នុងក្នុងក្នុងក្នុង H (1)

ទំនួរ : នឹងបានលើក្នុងក្នុងក្នុងក្នុងក្នុង AVL Tree

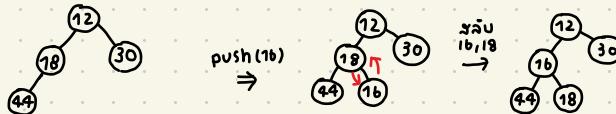
Binary Heap

complete tree
តើងតាកបអោលំដាប់, ចាប់បាន

"Push"

→ នៅលើលូវ complete tree
→ (min) swap ក្នុង key ដើម្បី $<$ parent

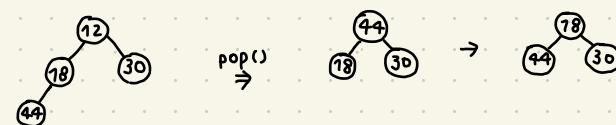
Ex.



"Pop"

→ ឈប់តួចកាប់ស្ថុតាមវិធីលើក្នុង root
→ swap នូវក្នុង

Ex.



Using Array

- Travel នូវ BST
- ឡើងក្នុង index 1
- ត្រូវការងារចំណែក $2k, 2k+1$, និងការចំណែក $\frac{k}{2}$ នៅរវាង

Push = $O(\log N)$, Pop = $O(\log N)$, Top = $O(1)$

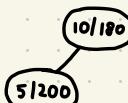
Merge = $O(N \log N)$

Stock Market

using priority queue

PTT	351.00		+4.00(+1.15%)
Volume	Bid	Offer	Volume
229,800	350.00	351.00	17,700
129,300	349.00	352.00	733,200
75,800	348.00	353.00	36,700
113,800	347.00	354.00	190,000
80,400	346.00	355.00	104,700

market.submitSellOrder("Paint", "sc8", 5, 200);
 market.submitSellOrder("Meen", "sc8", 10, 180);



market.submitBuyOrder("Gift", "sc8", 6, 250); MATCH!!
 market.submitBuyOrder("Kob", "sc8", 6, 250); MATCH!!



Hash Table

set, map

- สร้าง hash table "Chaining"
 - 1 ถ้า obj. มาก hash ย่อ index 10 ไปต่อหาง List
 - กันขบวน
 - รูปแบบที่ซ้ำกัน
- $O(n+m)$, ความเร็ว $O(c+1)$
 γ cardinality → hash แม่นยำมาก
- Universal Family → ต้องการขนาด $\leq \frac{1}{m}$
 $O(\alpha + 1)$; $\alpha = \frac{n}{m}$
 \downarrow ควรอยู่ประมาณ 0.5 - 1.0

Separate Chaining

- array ของ Linked List
- ใช้ || จัดต่อเนื่อง ยุ่งยาก
- ตอบด้วย Load Factor (α) ค่าต่ำ ($\alpha > 1$)

Open Address

Linear
Quadratic

- array เหมือน
- ใช้ ค่าปิดช่อง
- Search $O(1)$ หมายความคือ $\alpha = 1$!

Linear Probing

จะ → ค่าปิดช่อง คือ

- "Insert" hash นำ index แล้ว 10 ไปจุ
- อะไร ? นับ 8 ถูกต้องไปเรื่อยๆ

- "Rehash" หรือค่าเริ่มต้น เป็น 0 ก็จะ

- เพิ่ม ขนาด array #2
- ติดครั้ง hash ใหม่
- copy ข้อมูลเก่า → hash → 10 ใช้ตารางใหม่

การแก้ collision

ข้อมูล ↑ การค้น ↑

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	96	97	98	99	100																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																									
6801	6242	2077	6861	54	6281	2474	6023	540	1643	6022	54	2047	6455	54	2449	6874	54	2851	7295	54	3253	7714	54	3655	8133	54	4057	8552	54	4459	8971	54	4861	9390	54	5263	9809	54	5665	10228	54	6067	10647	54	6469	11066	54	6871	11485	54	7273	11904	54	7675	12323	54	8077	12742	54	8479	13161	54	8881	13580	54	9283	14000	54	9685	14419	54	10087	14838	54	10489	15257	54	10891	15676	54	11293	16095	54	11695	16514	54	12097	16933	54	12499	17352	54	12891	17771	54	13293	18190	54	13695	18609	54	14097	19028	54	14499	19447	54	14891	19866	54	15293	20285	54	15695	20704	54	16097	21123	54	16499	21542	54	16891	21961	54	17293	22380	54	17695	22799	54	18097	23218	54	18499	23637	54	18891	24056	54	19293	24475	54	19695	24894	54	20097	25313	54	20499	25732	54	20891	26151	54	21293	26570	54	21695	26989	54	22097	27408	54	22499	27827	54	22891	28246	54	23293	28665	54	23695	29084	54	24097	29503	54	24499	29922	54	24891	30341	54	25293	30760	54	25695	31179	54	26097	31598	54	26499	32017	54	26891	32436	54	27293	32855	54	27695	33274	54	28097	33693	54	28499	34112	54	28891	34531	54	29293	34950	54	29695	35369	54	30097	35788	54	30499	36207	54	30891	36626	54	31293	37045	54	31695	37464	54	32097	37883	54	32499	38302	54	32891	38721	54	33293	39140	54	33695	39559	54	34097	39978	54	34499	40397	54	34891	40816	54	35293	41235	54	35695	41654	54	36097	42073	54	36499	42492	54	36891	42911	54	37293	43330	54	37695	43749	54	38097	44168	54	38499	44587	54	38891	45006	54	39293	45425	54	39695	45844	54	40097	46263	54	40499	46682	54	40891	47101	54	41293	47520	54	41695	47939	54	42097	48358	54	42499	48777	54	42891	49196	54	43293	49615	54	43695	50034	54	44097	50453	54	44499	50872	54	44891	51291	54	45293	51710	54	45695	52129	54	46097	52548	54	46499	52967	54	46891	53386	54	47293	53805	54	47695	54224	54	48097	54643	54	48499	55062	54	48891	55481	54	49293	55900	54	49695	56319	54	50097	56738	54	50499	57157	54	50891	57576	54	51293	57995	54	51695	58414	54	52097	58833	54	52499	59252	54	52891	59671	54	53293	60090	54	53695	60509	54	54097	60928	54	54499	61347	54	54891	61766	54	55293	62185	54	55695	62604	54	56097	63023	54	56499	63442	54	56891	63861	54	57293	64280	54	57695	64709	54	58097	65128	54	58499	65547	54	58891	65966	54	59293	66385	54	59695	66804	54	60097	67223	54	60499	67642	54	60891	68061	54	61293	68480	54	61695	68900	54	62097	69319	54	62499	69738	54	62891	70157	54	63293	70576	54	63695	71095	54	64097	71514	54	64499	71933	54	64891	72352	54	65293	72771	54	65695	73190	54	66097	73609	54	66499	74028	54	66891	74447	54	67293	74866	54	67695	75285	54	68097	75704	54	68499	76123	54	68891	76542	54	69293	76961	54	69695	77380	54	70097	77799	54	70499	78218	54	70891	78637	54	71293	79056	54	71695	79475	54	72097	79894	54	72499	80313	54	72891	80732	54	73293	81151	54	73695	81570	54	74097	81989	54	74499	82408	54	74891	82827	54	75293	83246	54	75695	83665	54	76097	84084	54	76499	84503	54	76891	84922	54	77293	85341	54	77695	85760	54	78097	86179	54	78499	86598	54	78891	87017	54	79293	87436	54	79695	87855	54	80097	88274	54	80499	88693	54	80891	89112	54	81293	89531	54	81695	89950	54	82097	90369	54	82499	90788	54	82891	91207	54	83293	91626	54	83695	92045	54	84097	92464	54	84499	92883	54	84891	93302	54	85293	93721	54	85695	94140	54	86097	94559	54	86499	94978	54	86891	95397	54	87293	95816	54	87695	96235	54	88097	96654	54	88499	97073	54	88891	97492	54	89293	97911	54	89695	98330	54	90097	98749	54	90499	99168	54	90891	99587	54	91293	99906	54	91695	100325	54	92097	100744	54	92499	101163	54	92891	101582	54	93293	101901	54	93695	102320	54	94097	102739	54	94499	103158	54	94891	103577	54	95293	103996	54	95695	104415	54	96097	104834	54	96499	105253	54	96891	105672	54	97293	106091	54	97695	106510	54	98097	106929	54	98499	107348	54	98891	107767	54	99293	108186	54	99695	108605	54	100097	109024	54	100499	109443	54	100891	109862	54	101293	110281	54	101695	110700	54	102097	111119	54	102499	111538	54	102891	111957	54	103293	112376	54	103695	112795	54	104097	113214	54	104499	113633	54	104891	114052	54	105293	114471	54	105695	114890	54	106097	115309	54	106499	115728	54	106891	116147	54	107293	116566	54	107695	116985	54	108097	117404	54	108499	117823	54	108891	118242	54	109293	118661	54	109695	119080	54	110097	119500	54	110499	120000	54	110891	120500	54	111293	121000	54	111695	121500	54	112097	122000	54	112499	122500	54	112891	123000	54	113293	123500	54	113695	124000	54	114097	124500	54	114499	125000	54	114891	125500	54	115293	126000	54	115695	126500	54	116097	127000	54	116499	127500	54	116891	128000	54	117293	128500	54	117695	129000	54	118097	129500	54	118499	130000	54	118891	130500	54	119293	131000	54	119695	131500	54	120097	132000	54	120499	132500	54	120891	133000	54	121293	133500	54	121695	134000	54	122097	134500	54	122499	135000	54	122891	135500	54	123293	136000	54	123695	136500	54	124097	137000	54	124499	137500	54	124891	138000	54	125293	138500	54	125695	139000	54	126097	139500	54	126499	140000	54	126891	140500	54	127293	141000	54	127695	141500	54	128097	142000	54	128499	142500	54	128891	143000	54	129293	143500	54	129695	144000	54	130097	144500	54	130499	145000	54	130891	145500	54	131293	146000	54	131695	146500	54	132097	147000	54	132499	147500	54	132891	148000	54	133293	148500	54	133695	149000	54	134097	149500	54	134499	150000	54	134891	150500	54	135293	151000	54	135695	151500	54	136097	152000	54	136499	152500	54	136891	153000	54	137293	153500	54	137695	154000	54	138097	154500	54	138499	155000	54	138891	155500	54	139293	156000	54	139695	156500	54	140097	157000	54	140499	157500	54	140891	158000	54	141293	158500	54	141695	159000	54	142097	159500	54	142499	160000	54	142891	160500</td

Quadratic Probing

→ କାହିଁ ଲେଖି ଏବଂ କଥାରେ କଥାରେ

"Insert" hash นา index แล้ว เติมปะ
- ชาด ? ลง ช่อง ก็ (bin + K) / M

Ex. hash 11 กว่า 5 816 → $5+1 = 6$ 816
 จำนวนที่มากกว่า 5
 $6+2 = 8$ 816
 $8+3 = 11$ 816
 $11+4 = 15$ 816
 $15+5 = 20$ 1.16 = 4 816
 2 * 16 หนึ่ง!

2^x ወገኑ!

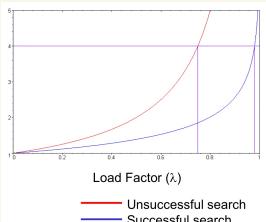
and it will visit every index

"Rehash" ແບກສັນເກີນ

"Delete" Lazy Delete!

- ດີ່ນເພີ້ນ `isDeleted` ລາຍກູບ ຂະໜາກູວ່າ `true`
 - ເປັນຕົວກຸ່ມອົບ ໃຫ້ອົບ ໄກສະໝັກ

Run-time Analysis



- $\frac{1}{1-\lambda}$ မျှ၍လဲ၏၂၀
 - $\ln(\frac{1}{1-\lambda})/\lambda$ မျှ၏၂၀

ବେଳୁଳାଙ୍କ କୌଣସି ବେଳୁଳାଙ୍କ କୌଣସି collision

Logg factor ↑ collision ↑

๖ วันที่ ๓ ๑ มกราคม พ.ศ. ๒๕๖๔

ප්‍රේඛන තොගවල්ලිසන

Linear probing

Unsuccessful search

Successful search

Quadratic probing

Unsuccessful search

Successful search

សារិយាយនវិភាគ ចំណាំរក្សា

Digitized by srujanika@gmail.com

Digitized by srujanika@gmail.com

Load Factor (λ)

- 9.6 Load Factor กี่วันต่อเดือน กี่%

ເບີໂທສາງການຄົງກົງ ສູ່ເວລັກສູ່ສົມບັດ ສະຫຼຸບມະຫາວິທະຍາ

cache (in CPU) - સર્વાંગી ગ્રાડ. probing

Graph

tree ສູ່ subset ວົງການ

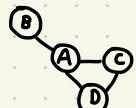
- ມີ cycle \rightarrow graph, ໂດຍມີ cycle \rightarrow tree

vertex ຫຼື edge ເນື້ອ

Graph Representation

- Edge List
- Adjacency Matrix
- Adjacency List *

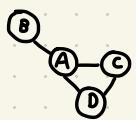
Edge List



Edge:

(A,B), (A,C), (A,D), (D,D)

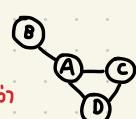
Adjacency Matrix



	A	B	C	D
A	0	1	1	1
B	1	0	0	0
C	1	0	0	1
D	1	0	1	0

Adjacency List

ໃຊ້ mem ຂອຍກວ່າ



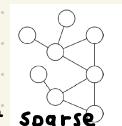
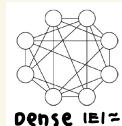
A	Ad to	B, C, D	$A \rightarrow B \rightarrow C \rightarrow D$
B	"—"	A	$B \rightarrow A$
C	"—"	A, D	$C \rightarrow A \rightarrow D$
D	"—"	A, C	$D \rightarrow A \rightarrow C$

Run-time Analysis

Op	hasEdge(V1, V2)	find all Neighbors(V)	Memory
Edge List	$O(E)$	$O(E)$	$O(E)$
Adj Matrix	$O(1)$	$O(V)$	$O(V ^2)$
Adj. List	$O(V)$	$O(V)$	$O(V + E)$
Adj. List (if deg << V)	$O(\text{deg})$	$O(\text{deg})$	$O(V + E)$

ດາວໂຫຼວງ ວົງການ depends on V, E

"Density" ດາວໂຫຼວງ ນິກົດ



Dense $|E|=|V|^2$

Sparse

DFS

- Set visited ໃບຕົວ false \rightarrow loop ພົບຖານ
- visit neighbor $O(|E|)$
- Runtime $O(|V| + |E|)$

BFS

- ດັບເຫັນ neighbor ຖກຈຳກົດ
- Length = # of edge
- Distance = Length ທີ່ສົ່ງກຶ່ສູດ
(∞ = ມີເລື່ອ, 0 = ວິທະຍາມທີ່ຕໍ່ຕົ້ນ)
- Shortest Path $O(|V| + |E|)$