บทที่ 3

โครงสร้างข้อมูล สำหรับการวิเคราะห์ถัวเฉลี่ย

ในบทนี้จะกล่าวถึงโครงสร้างข้อมูล ที่ใช้เป็นกรณีศึกษาสำหรับการวิเคราะห์ถัวเฉลี่ย ซึ่งจะใช้วิธีศักย์ในการวิเคราะห์ มีดังนี้คือ

1. ตารางไดนามิก (Dynamic Table)
2. ไบโนเมียลฮีพ (Binomial Heap)
3. เลซี่ ไบโนเมียลฮีพ (Lazy Binomial Heap)
4. ฟิโบนักชี่ฮีพ (Fibonacci Heap)
5. สคิวฮีพ (Skew Heap)

1. ตารางไดนามิก

ตารางไดนามิก คือ ตารางจัดเก็บข้อมูลที่ขนาดของตารางสามารถขยายและยุบตัวได้ตามจำนวนข้อมูล ถ้าในตารางนั้นมีการใส่ข้อมูลจนเต็มตาราง การจะใส่ข้อมูลเข้าไปอีกจะต้องมีการสร้างตารางใหม่ที่มีขนาดใหญ่กว่าเดิม และข้อมูลทั้งหมดที่ถูกจัดเก็บในตารางเดิม ก็จะต้องถูกคัดลอกไปไว้ในตารางใหม่ ซึ่งเรียกว่า มีการขยายตาราง ทำนองเดียวกันถ้าข้อมูลหลายๆตัวถูกลบออกจากตารางมากๆ ก็จะต้องมีการสร้างตารางใหม่ที่มีขนาดเล็กลงกว่าเดิม และข้อมูลทั้งหมดที่ถูกจัดเก็บในตารางเดิม ก็จะต้องถูกคัดลอกไปไว้ในตารางใหม่ ซึ่งเรียกว่า มีการยุบตาราง[14]

การขยายตาราง

เมื่อถึงจุดหนึ่งที่จะต้องมีการขยายตาราง เช่นเมื่อช่องทั้งหมดในตารางมีข้อมูลเต็มอยู่เป็นต้น และต้องการเพิ่มข้อมูลลงไปอีก ก็จะมีการสร้างตารางใหม่ที่มีขนาดใหญ่กว่าเดิม พร้อมทั้งคัดลอกข้อมูลในตารางเดิมสู่ตารางใหม่ โดยทั่วไปการสร้างตารางใหม่จะสร้างให้มีขนาดเป็น 2 เท่าของตารางเดิม [14]

ให้ T เป็น ตาราง

$table[T]$ เป็น พอยเตอร์ที่จะชี้ไปยังตาราง

$num_i[T]$ เป็น จำนวนข้อมูลในตารางในการดำเนินงานที่ i

$size_i[T]$ เป็น จำนวนช่องข้อมูลทั้งหมดในตารางในการดำเนินงานที่ i

Φ_i เป็น ฟังก์ชันศักย์ของการดำเนินงานที่ i

สำหรับการวิเคราะห์ในแบบถัวเฉลี่ย จะขอใช้วิธีศักย์ กำหนดให้ Φ_i ที่จะมีค่าเริ่มต้นเป็น 0 และไม่มีค่าติดลบ $\Phi_i = 2num_i[T] - size_i[T]$

ให้ c_i เป็นต้นทุนจริงของการเพิ่มที่ i โดยที่ $i = 1, 2, ..., n$ และให้การเพิ่มข้อมูล 1 ตัวลงในตารางใช้ 1 หน่วย

การหาต้นทุนถัวเฉลี่ยของการเพิ่มข้อมูลลงในตารางไดนามิก แบ่งได้เป็น 2 กรณี ดังนี้คือ

กรณีที่ 1 ถ้าการดำเนินงานการเพิ่มที่ i ไม่ได้ส่งผลให้เกิดการขยายตาราง $size_i[T] = size_{i-1}[T]$ และ ต้นทุนถัวเฉลี่ยของการดำเนินงานจะเป็น

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (2num_i[T] - size_i[T]) - (2num_{i-1}[T] - size_{i-1}[T])$$

$$1 + (2num_i[T] - size_i[T]) - (2(num_i[T] - 1) - size_i[T]) = 3$$

กรณีที่ 2 ถ้าการดำเนินงานการเพิ่มที่ i ส่งผลให้เกิดการขยายตาราง $size_i[T] / 2 = size_{i-1}[T] = num_i[T] - 1$ และต้นทุนถัวเฉลี่ยของการดำเนินงาน จะเป็น

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = num_i[T] + (2num_i[T] - size_i[T]) - (2num_{i-1}[T] - size_{i-1}[T])$$

$$= num_i[T] + (2num_i[T] - (2num_i[T] - 2)) - (2(num_i[T] + 1) - (num_i[T] + 1)) = 3$$

นั่นคือ ตารางจะมีการขยายหรือไม่ก็ตาม ต้นทุนถัวเฉลี่ย ($\hat{c}_i$) อย่างมากเท่ากับ 3 และจาก 2 กรณีข้างต้นสรุปว่า ต้นทุนถัวเฉลี่ยต่อการเพิ่ม 1 ครั้งมีค่าคงที่[14]

การยุบตาราง

ในการยุบตารางจะยุบเมื่อมีที่ว่างมากเกินไป เช่นถ้ามีจำนวนข้อมูลน้อยกว่าเศษหนึ่งส่วนสี่ของจำนวนช่องข้อมูลทั้งหมดในตาราง เป็นต้น จะต้องมีการจัดหาตารางใหม่ให้มีขนาดเล็กลงกว่าเดิมครึ่งหนึ่งพร้อมทั้งคัดลอกข้อมูลจากตารางเก่าสู่ตารางใหม่ ส่วนตารางเก่าก็สามารถคืนให้กับระบบจัดการหน่วยความจำได้[14]

สำหรับการวิเคราะห์ในถัวเฉลี่ย จะใช้วิธีศักย์เริ่มด้วยการนิยามฟังก์ชันศักย์ Φ_i ที่มีค่าเริ่มต้นเป็น 0 และไม่มีค่าติดลบ ซึ่งก็คือ [14]

$$\Phi_i = \begin{cases} 2num[T]_i - size_i[T] & if \alpha_i \geq 1/2 \\ size_i[T]/2 - num_i[T] & if \alpha_i < 1/2 \end{cases}$$

โดยที่ α_i เป็นภาระของตาราง หรือก็คือจำนวนข้อมูลในตาราง หารด้วยจำนวนช่องข้อมูลทั้งหมดในตาราง ให้ c_i เป็นต้นทุนจริงของการลบที่ i โดยที่ $i = 1, 2, ..., n$ และให้การลบข้อมูล 1 ตัวใช้ 1 หน่วย สำหรับการหาต้นทุนถัวเฉลี่ยของการลบข้อมูลในตารางไดนามิก แบ่งได้เป็น 3 กรณี ดังนี้คือ [14]

กรณีที่ 1 ถ้าการดำเนินงานการลบข้อมูลที่ i ไม่ได้ส่งผลให้เกิดการยุบตาราง โดยที่ $\alpha_{i-1} < 1/2$ ต้นทุนถัวเฉลี่ย เป็น [14]

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (size_i[T] / 2 - num_i[T]) - (size_{i-1}[T] / 2 - num_{i-1}[T])$$

$$= 1 + (size_i[T] / 2 - num_i[T]) - (size_i[T] / 2 - (num_i[T]+1)) = 2$$

กรณีที่ 2 ถ้าการดำเนินงานการลบข้อมูลที่ i ไม่ได้ส่งผลให้เกิดการยุบตาราง โดยที่ $\alpha_{i-1} \geq 1/2$ ต้นทุนถัวเฉลี่ย เป็น [14]

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1} = 1 + (2num_i[T] - size_i[T]) - (2num_{i-1}[T] - size_{i-1}[T])$$
$$= 1 + (2num_i[T] - size_i[T]) - (2(num_i[T] + 1) - size_i[T]) = -1$$

หรือ

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= 1 + ((size_i[T] / 2) - num_i[T]) - (2num_{i-1}[T] - size_{i-1}[T])$$
$$1 + ((num_i[T] + 1) - num_i[T]) - (2(size_{i-1}[T] / 2) - size_{i-1}[T])$$
$$= 1 + 1 = 2$$

กรณีที่ 3 ถ้าการดำเนินงานการลบข้อมูลที่ i ส่งผลให้เกิดการยุบตาราง โดยที่ $\alpha_{i-1} < 1/2$ ต้นทุนจริงของการดำเนินงานเป็น $c_i = num_i[T] + 1$ เพราะว่า มีการลบข้อมูล 1 ตัว และมีการย้ายข้อมูลจำนวน $num_i[T]$ ตัว โดยที่ $size_i[T] / 2 = size_{i-1}[T] / 4 = num_i[T] + 1$ และต้นทุนถัวเฉลี่ยของการดำเนินงาน จะเป็น [14]

$$\hat{c}_i = c_i + \Phi_i - \Phi_{i-1}$$
$$= (num_i[T] + 1) + (size_i[T] / 2 - num_i[T]) - (size_{i-1}[T] / 2 - num_{i-1}[T])$$
$$= (num_i[T] + 1) + ((num_i[T] + 1) - num_i[T]) - ((2num_i[T] + 2) - (num_i[T] + 1))$$
$$= 1$$

จาก 3 กรณีข้างต้น สรุปว่าต้นทุนถัวเฉลี่ยต่อการลบ 1 ครั้ง มีค่าคงที่

สำหรับอัลกอริทึมการเพิ่มข้อมูลในตาราง และการลบข้อมูลในตารางเป็นดังนี้ [14]

TABLE-INSERT(T,x)

```
1.  if size[T] = 0
2.      then allocate table[T] with 1 slot
3.           size[T] ← 1
4.  if num[T] = size[T]
5.      then allocate new-table with 2size[T] slots
6.           insert all items in table[T] into new-table
7.           free table[T]
8.           table[T] ← new-table
9.           size[T] ← 2size[T]
10. insert x into table[T]
11. num[T] ← num[T]+1
```

TABLE-DELETE(T,x)

1. *delete x from table[T]*
2. *num[T]* ← *num[T] - 1*
3. *if num[T] = size[T] / 4*
4. *then allocate new-table with size[T] / 2 slots*
5. *insert all items in table[T] into new-table*
6. *free table[T]*
7. *table[T]* ← *new-table*
8. *size[T]* ← *size[T] / 2*

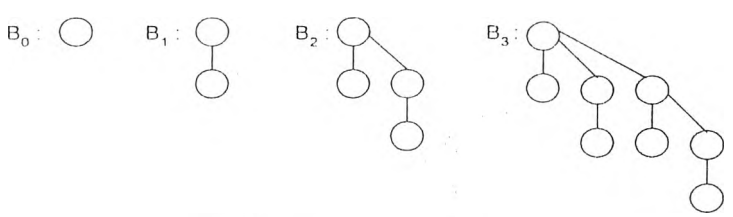
2. ไบโนเมียลฮีพ

ก่อนอื่นขอนิยามต้นไม้ไบโนเมียล B_k ดังนี้ [14]

นิยาม ต้นไม้ไบโนเมียล B_k เป็นต้นไม้ที่มีการเรียงลำดับ โดยที่

- ต้นไม้ไบโนเมียล B_0 จะประกอบด้วยโหนดโหนดเดียว
- ต้นไม้ไบโนเมียล B_k จะประกอบด้วยต้นไม้ไบโนเมียล B_{k-1} จำนวน 2 ต้นเชื่อมต่อกัน โดยรากของ B_{k-1} ต้นหนึ่งจะมีลูกทางขวาสุดเป็นรากของ B_{k-1} อีกต้นหนึ่ง

ต้นไม้ไบโนเมียล B_0 ถึง B_3 แสดงได้ดังรูปที่ 3.1



รูปที่ 3.1 แสดงต้นไม้ไบโนเมียล B_0, B_1, B_2 และ B_3

ต้นไม้ไบโนเมียล B_k มีความสูง k โดยมีจำนวนโหนดที่ระดับ i เท่ากับ $\binom{k}{i}$ ดังนั้น B_k มีจำนวนโหนดทั้งสิ้น 2^k โหนด

นิยาม ไบโนเมียล ฮีพ H เป็นรายการสะสมของต้นไม้ไบโนเมียล ที่มีคุณสมบัติ ดังนี้ [14]

- ข้อมูลในโหนดต่างๆของต้นไม้ไบโนเมียลแต่ละต้นในฮีพ H จะต้องมีการเรียงลำดับค่าแล้ว กล่าวคือค่าของโหนดใดๆ จะต้องมีค่ามากกว่าหรือเท่ากับค่าของโหนดพ่อของโหนดนั้นๆ
- มีต้นไม้ไบโนเมียล B_k สำหรับ k ใดๆ ได้อย่างมากเพียงหนึ่งต้นในฮีพ H

สำหรับการวิเคราะห์ถัวเฉลี่ยจะใช้วิธีศักย์ โดยที่ฟังก์ชันศักย์คิดจากจำนวนต้นไม้ในฮีพ และต้นทุนจริงเป็นสัดส่วนโดยตรงกับจำนวนการเชื่อมต้นไม้ที่มีความสูงเท่ากัน

การปฏิบัติการบนไบโนเมียล ฮีพ

1. การหาโหนดที่มีค่าน้อยที่สุด

การหาโหนดที่มีค่าน้อยที่สุดใช้เวลา O(1) โดยการจำและปรับแปลี่ยนตำแหน่งของโหนด ที่มีค่าน้อยสุดไว้ตลอดเวลา เมื่อมีการเปลี่ยนแปลงเกิดขึ้นระหว่างการปฏิบัติการต่างๆ[14]

2. การรวมฮีพ

การรวมไบโนเมียลฮีพ 2 ฮีพ มีขั้นตอนดังนี้

2.1 เชื่อมรากของต้นไม้ในฮีพทั้ง 2 เข้าด้วยกัน

2.2 ถ้าในฮีพมีต้นไม้ที่มีความสูงเท่ากัน เชื่อมต้นไม้นั้นเข้าด้วยกัน โดยให้รากที่มีค่าน้อยกว่า เป็นรากของต้นไม้ที่เชื่อมรวมกันแล้ว ดังนั้นต้นไม้ $B_{(k-1)}$ จำนวน 2 ต้น จะรวมเป็นต้นไม้ B_k ที่มีรากเป็นค่าที่น้อยกว่า

2.3 ทำ 2.2 ไปเรื่อยๆจนกว่าจะไม่มีต้นไม้ที่มีความสูงเท่ากัน

เวลาการรวมจะขึ้นอยู่กับจำนวนต้นไม้ในฮีพ ซึ่งจะมีอย่างมากสุด $\log n$ ต้น ดังนั้น การรวมฮีพ 2 ฮีพ ต้นทุนถัวเฉลี่ยในการรวมเท่ากับ $O(\log n)$ [14]

อัลกอริทึมการรวมมีดังต่อไปนี้ [14]

BINOMAIL-HEAP-UNION (H_1 , H_2)

```
1.  H ← MAKE-BINOMAIL-HEAP( )
2.  head[H] ← BINOMAIL-HEAP-MERGE ( H1 , H2 )
3.  free the objects H1 and H2 but not the lists they point to
4.  if head[H] = NIL
5.     then return H
6.  prev-x ← NIL
7.  x ← head[H]
8.  next-x ← sibling[x]
9.  while next-x ≠ NIL
10.    do if (degree[x] ≠ degree[next-x]) or (sibling[next-x]) ≠ NIL and degree
                 [sibling[next-x]] = degree[x])
11.          then prev-x ← x                          -- Case 1 and 2
12.               x ← next-x                          -- Case 1 and 2
13.          else if key[x] ≤ key[next-x]
14.                  then sibling[x] ← sibling[next-x]   -- Case 3
```

15. *BINOMAIL-LINK(next-x,x)* -- *Case 3*

16. *else if prev-x = NIL* -- *Case 4*

17. *then head[H]* ← *next-x* -- *Case 4*

18. *else sibling[prev-x]* ← *next-x* -- *Case 4*

19. *BINOMAIL-LINK(x,next-x)* -- *Case 4*

20. *x* ← *next-x* -- *Case 4*

21. *next-x* ← *sibling[x]*

22. *return H*

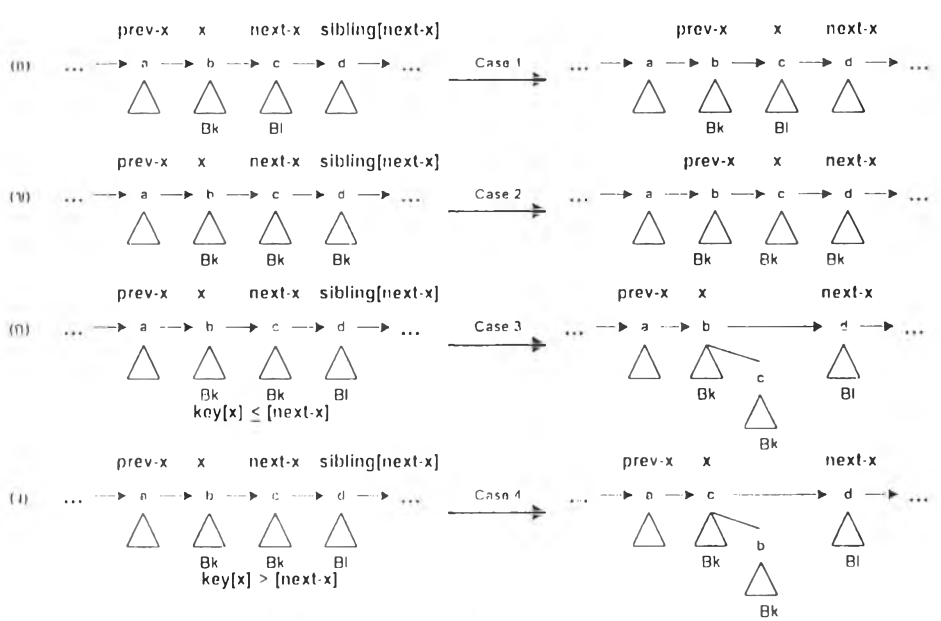
BINOMAIL-HEAP-MERGE (H_1 , H_2)

1. *H* ← *MAKE-BINOMIAL-HEAP( )*
2. *Concatenate the root lists of H_1 and H_2 into a new root list H*
3. *Free the objects H_1 and H_2*
4. *Return H*

BINOMIAL-LINK(y,z)

1. *p[y]* ← *z*
2. *sibling[y]* ← *child[z]*
3. *child[z]* ← *y*
4. *degree[z]* ← *degree[z]+1*

ที่ซึ่งอัลกอริทึมนี้ ทำงานดังแสดงในรูปที่ 3.2



รูปที่ 3.2 แสดงกรณีที่จะเกิดขึ้นในการรวมไบโนมียลฮีพ โดย x เป็นรากของต้นไม้ B_k และ $i > k$

โดยในกรณีที่ 1 degree[x] ≠ degree[next-x] กรณีที่ 2 degree[x] = degree[next-x] = degree[sibling[next-x]] กรณีที่ 3 degree[x] = degree[next-x] ≠ degree[sibling[next-x]] และ key [x] ≤ key[next-x] กรณีที่ 4 degree[x] = degree[next-x] ≠ degree[sibling[next-x]] และ key[x] ≥ key[next-x] [14]

3. การเพิ่ม

การเพิ่มข้อมูลในฮีพ มีขั้นตอนดังนี้ [14]

3.1 สร้างฮีพใหม่ที่ประกอบด้วยโหนดใหม่ที่มาเพิ่ม ที่ซึ่งใช้เวลา O(1)

3.2 นำฮีพใหม่ที่สร้างขึ้น รวมเข้ากับฮีพเดิมที่มีอยู่ ใช้เวลาในการรวมO(log n)

ดังนั้นหากกำหนดให้ฟังก์ชันศักย์เป็นจำนวนต้นไม้ในฮีพ ต้นทุนถัวเฉลี่ยจะเท่ากับต้นทุนจริงบวกกับจำนวนต้นไม้ที่เปลี่ยนแปลง ถ้าต้นทุนจริงในการเพิ่มเป็น v แสดงว่าต้นไม้ B_0, B_1, ... , $B_{(v-2)}$ เดิมในฮีพ เพื่อรวมกับต้นไม้ B_0 ในฮีพใหม่ จะกลายเป็นต้นไม้ $B_{(v-1)}$ 1 ต้น จำนวนต้นไม้ที่เปลี่ยนแปลงจะเท่ากับ 1-(v-1) ดังนั้น ต้นทุนถัวเฉลี่ยในการเพิ่มเท่ากับ v+(1-(v-1)) = 2 = O(1) [14]

อัลกอริทึมการเพิ่ม มีดังต่อไปนี้ [14]

BINOMIAL-HEAP-INSERT (H,x)

1. *H' ← MAKE-BINOMIAL-HEAP( )*
2. *p[x] ← NIL*
3. *child[x] ← NIL*
4. *sibling[x] ← NIL*
5. *degree[x] ← 0*
6. *head[H'] ← x*
7. *H ← BINOMIAL-HEAP-UNION(H,H')*

4. การลบโหนดที่มีค่าน้อยที่สุด

การลบโหนดที่มีค่าน้อยที่สุด มีขั้นตอนดังนี้ [14]

4.1 หาโหนดที่เป็นรากของต้นไม้ ที่มีค่าน้อยสุด แล้วลบออก

4.2 สร้างฮีพใหม่ ซึ่งประกอบไปด้วยรายการของต้นไม้ย่อยของโหนดที่ถูกลบออก

4.3 รวมฮีพใหม่นี้กับฮีพเดิมที่ได้นำต้นไม้ที่มีโหนดที่รากมีค่าน้อยที่สุดออกแล้ว

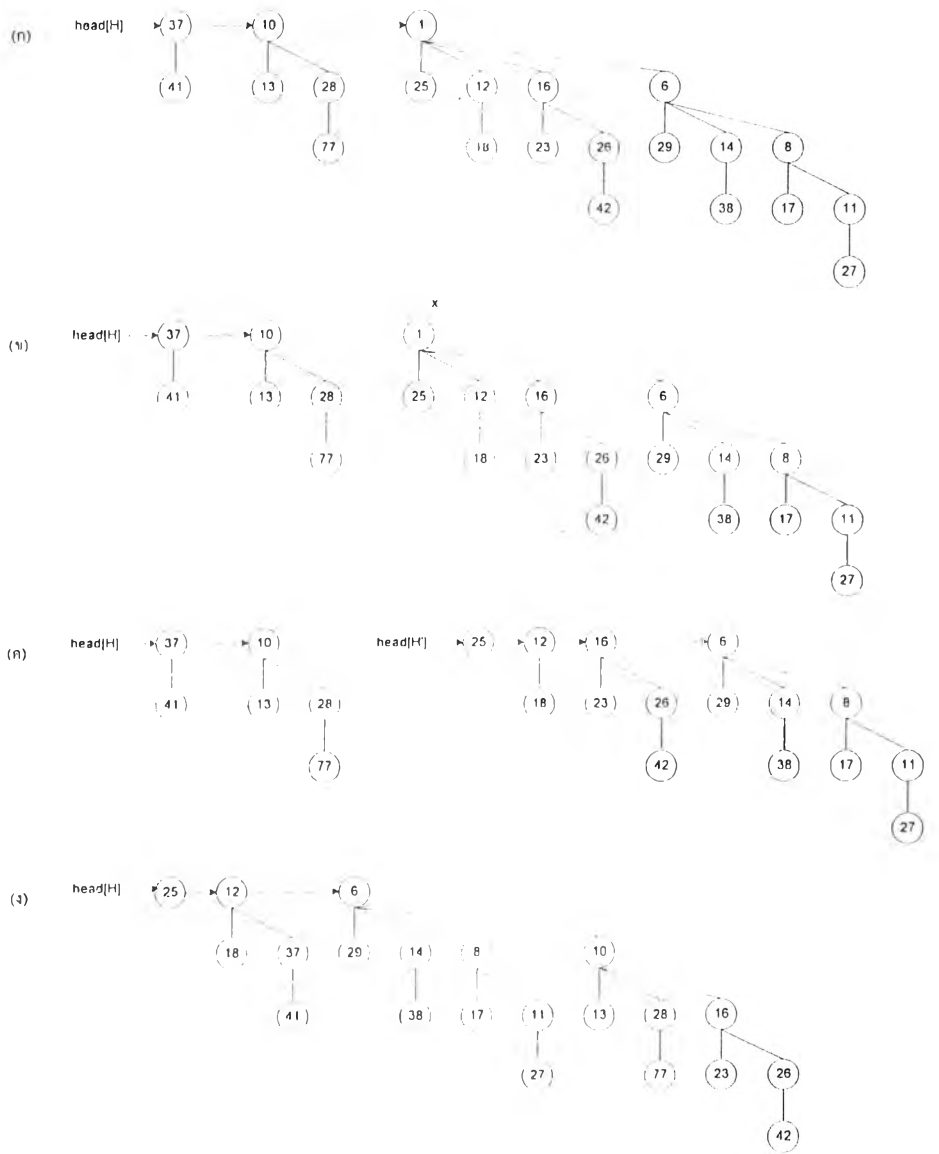
การหาโหนดที่เป็นรากของต้นไม้ ที่มีค่าน้อยสุด ใช้เวลา O(1) ส่วนการลบรากของต้นไม้ซึ่งมีค่าน้อยสุด ใช้เวลา O(1) การสร้างฮีพใหม่ใช้เวลา O(log n) และการรวมฮีพเข้าด้วยกัน ใช้เวลา O(log n) ดังนั้นเวลาที่ใช้ในการลบโหนดที่มีค่าน้อยที่สุดเป็น O(log n) [14]

อัลกอริทึมการลบโหนดที่มีค่าน้อยที่สุดมีดังต่อไปนี้ [14]

BINOMIAL-HEAP-EXTRACT-MIN(H)

1. *find the root x with the minimum key in the root list of H, and remove x from the root list of H*
2. *H' ⟵ MAKE-BINOMIAL-HEAP( )*
3. *H ⟵ BINOMIAL-HEAP-UNION(H,H')*
4. *return x*

อัลกอริทึมนี้ทำงานดังแสดงในรูปที่ 3.3 [14]



รูปที่ 3.3 แสดงการลบโหนดที่มีค่าน้อยที่สุด ของไบโนเมียลฮีพ // (ก) ก่อนทำการลบโหนด (ข) โหนดที่มีค่าน้อยที่สุด คือ โหนดที่ถูกชี้ด้วย x จะถูกลบออก (ค) หลังการลบโหนดได้ฮีพใหม่ คือ ฮีพ //' (ง) ผลลัพธ์ของการรวมฮีพ // และ ฮีพ //'

5. การลดค่าของโหนด

การลดค่าของโหนดเป็นดังนี้ โดยสมมติให้โหนด x เป็นโหนดที่จะถูกลดค่า [14]

1. ทำการลดค่าของโหนด x
2. ถ้าการลดค่าก่อให้เกิดการเปลี่ยนแปลงลำดับในฮีพ กล่าวคือถ้าโหนด x มีค่าน้อยกว่าโหนดพ่อของโหนด x แล้ว
   2.1 ให้สลับ โดยให้โหนด x ขึ้นไปแทนที่พ่อของโหนด x ส่วนพ่อของโหนด x ลงมาแทนที่โหนด x
   2.2 ย้อนกลับไปทำข้อ 2 จนกระทั่งโหนด x เป็นรากของต้นไม้ หรือโหนด x มีค่ามากกว่าโหนดพ่อของโหนด x

การลดค่าของโหนดใช้เวลา $O(\log n)$ เนื่องจากต้นไม้สูงอย่างมาก $\lfloor \log n \rfloor$ ดังอัลกอริทึมต่อไปนี้ โดยสมมติให้ ค่าของโหนด x จะถูกลดค่าเป็นค่า k [14]

อัลกอริทึมการลดค่าโหนดมีดังต่อไปนี้ [14]

BINOMIAL-HEAP-DECREASE-KEY(H, x, k)

1. *key[x]* $\leftarrow k$
2. *y* $\leftarrow x$
3. *z* $\leftarrow p[y]$
4. *while z* $\neq$ *NIL and key[y] < key[z]*
5. *do exchange key[y] ↔ key[z]*
6. *y* $\leftarrow z$
7. *z* $\leftarrow p[y]$
8. *end do*

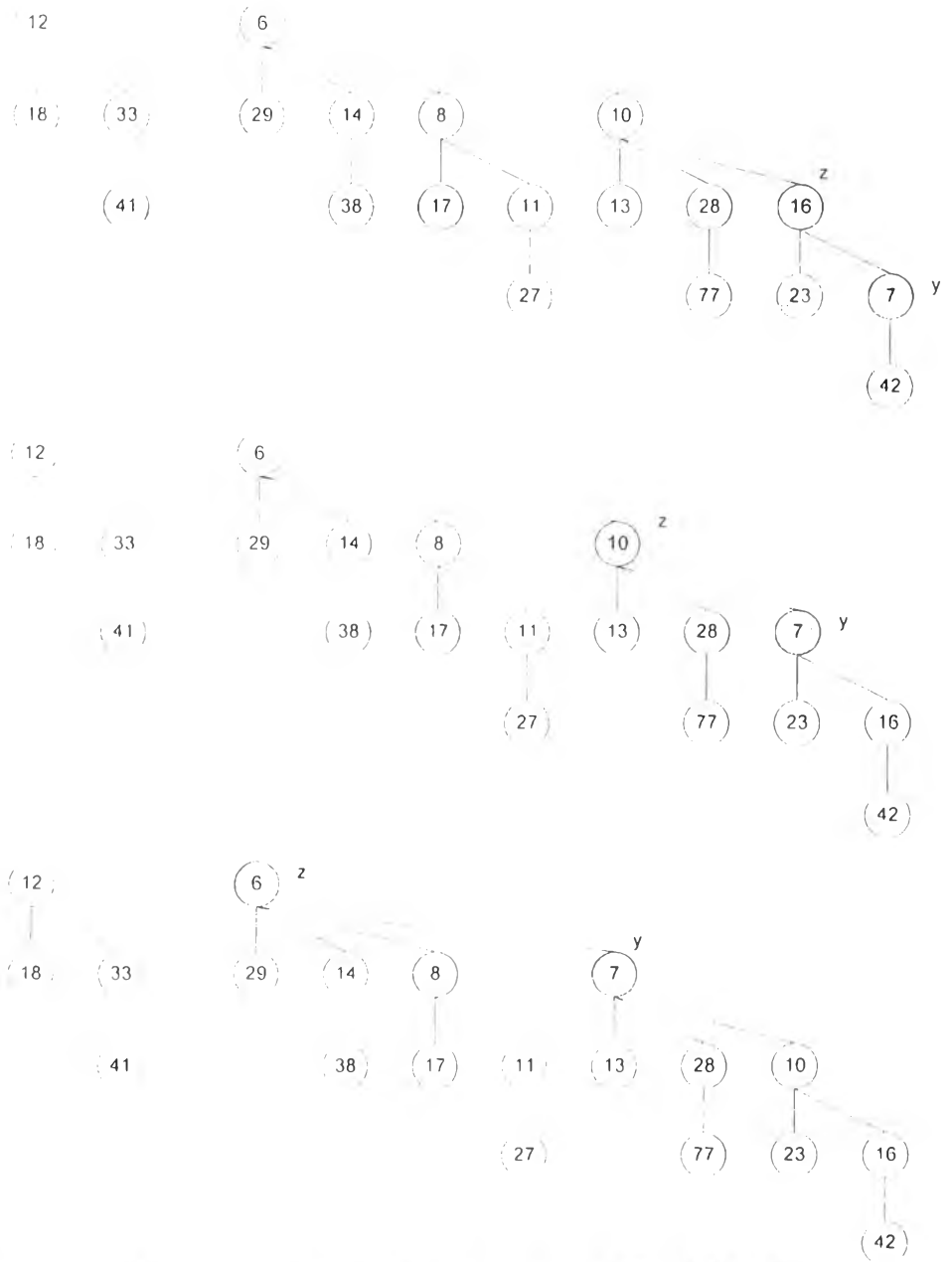
อัลกอริทึมนี้ทำงานดังแสดงในรูปที่ 3.4

3. เลซี่ ไบโนเมียล ฮีพ

เลซี่ไบโนเมียลฮีพ มีลักษณะการจัดเก็บ และการจัดการคล้ายกับไบโนเมียลฮีพ คือเป็นรายการของต้นไม้ไบโนเมียล แต่ต่างกันตรงที่เลซี่ไบโนเมียลฮีพ สามารถมีต้นไม้ที่มีความสูงเท่ากันได้มากกว่า 1 ต้น

นิยาม เลซี่ ไบโนเมียล ฮีพ // มีคุณสมบัติ ดังนี้

- ต้นไม้ไบโนเมียลแต่ละต้นในฮีพ // จะต้องมีการเรียงลำดับค่าแบบฮีพแล้ว กล่าวคือค่าของโหนดใดๆ จะต้องมีค่ามากกว่า หรือเท่ากับค่าของโหนดพ่อของโหนดนั้นๆ
- เลซี่ไบโนเมียลฮีพ // สามารถมีต้นไม้ไบโนเมียลที่มีความสูงเท่ากันได้มากกว่า 1 ต้น



รูปที่ 3.4 แสดงการลดค่าโหนด ของไบโนมียลฮีพ (ก) ค่าของโหนด y ถูกลดค่าให้เป็น 7 ที่ซึ่งมีค่าน้อยกว่าค่าของพ่อของโหนด y ซึ่งคือโหนด z (ข) ค่าของโหนด y สลับกับค่าของโหนด z แล้วตั้งให้โหนด y และโหนด z จะเลื่อนขึ้นไป 1 ระดับในต้นไม้ แต่ค่าของโหนด y ยังมีค่าน้อยกว่าค่าของโหนด z (ค) ค่าของโหนด y สลับกับค่าของโหนด z แล้วตั้งให้โหนด y และโหนด z จะเลื่อนขึ้นไป 1 ระดับในต้นไม้ โดยค่าของโหนด y มีค่ามากกว่าค่าของโหนด z จึงหยุด

สำหรับการวิเคราะห์ถัวเฉลี่ยจะใช้วิธีศักย์ โดยที่ฟังก์ชันศักย์คิดจากจำนวนของต้นไม้ ส่วนต้นทุนจริงเป็นสัดส่วนโดยตรงกับการเชื่อมต้นไม้ที่มีความสูงเท่ากัน

การปฏิบัติการของเลซี่ ไบโนมียล ฮีพ

1. การหาโหนดที่มีค่าน้อยที่สุด

การหาโหนดที่มีค่าน้อยที่สุดใช้เวลา O(1) โดยการจำและปรับเปลี่ยนตำแหน่งของโหนด ที่มีค่าน้อยสุดไว้ตลอดเวลา เมื่อมีการเปลี่ยนแปลงเกิดขึ้นระหว่างการปฏิบัติการต่างๆ เช่นเดียวกับไบโนมียลฮีพ [9]

2. การรวมฮีพ

การรวมฮีพทำได้โดยการสร้างฮีพใหม่ ที่ประกอบไปด้วยต้นไม้ในฮีพแต่ละฮีพมาต่อกัน โดยในการรวมฮีพในแอพลิเคชันไบโนมียลฮีพนั้นไม่จำเป็นต้องทำการเชื่อมต้นไม้ที่มีดีกรีเท่ากัน ดังนั้นจึงใช้เวลาเพียง O(1) [9]

อัลกอริทึมการรวมมีดังต่อไปนี้ [9]

LAZY-BINOMIAL-HEAP-UNION (H_1 , H_2)

1. $H \leftarrow$ MAKE-BINOMAIL-HEAP( )

2. head[H] $\leftarrow$ BINOMAIL-HEAP-MERGE (H_1 , H_2)

3. free the objects H_1 and H_2 but not the lists they point to

3. การเพิ่มข้อมูล

การเพิ่มข้อมูล ทำได้โดยการสร้างฮีพใหม่ที่มีข้อมูลเพียง 1 โหนด จากนั้นนำไปต่อกับรายการต้นไม้ของฮีพเดิม จึงใช้เวลาในการเพิ่มเพียง O(1) [9]

อัลกอริทึมการเพิ่ม มีดังต่อไปนี้ [9]

LAZY-BINOMIAL-HEAP-INSERT (H,x)

1. H' $\leftarrow$ MAKE-BINOMIAL-HEAP( )

2. p[x] $\leftarrow$ NIL

3. child[x] $\leftarrow$ NIL

4. sibling[x] $\leftarrow$ NIL

5. degree[x] $\leftarrow$ 0

6. head[H'] $\leftarrow$ x

7. return H'

4. การลบโหนดที่มีค่าน้อยที่สุด

การลบโหนดที่มีค่าน้อยที่สุด มีขั้นตอนดังนี้ [9]

1. ลบโหนดที่มีค่าน้อยสุดออก
2. สร้างฮีพใหม่ ซึ่งประกอบไปด้วยรายการของต้นไม้ย่อยของโหนดที่ถูกลบออก

3. ทำการเชื่อมรายการของต้นไม้ย่อยที่มีความสูงเท่ากัน เพื่อให้ได้ผลลัพธ์เป็นสิพที่ไม่มีต้นไม้ความสูงเท่ากันเกิน 1 ต้น

ให้ r เป็น ระดับความสูงของต้นไม้ที่มีโหนดที่มีค่าน้อยที่สุด

t เป็น จำนวนต้นไม้ในสิพ

ดังนั้น ฟังก์ชันศักย์ก่อนการดำเนินงานการลบเป็น t หลังจากทำการลบโหนดที่มีค่าน้อยที่สุดออก มีต้นไม้ย่อยที่แตกออกมา $r+r$ ต้น ซึ่งในการรวมต้นไม้เหล่านี้ ต้นทุนจริงเป็น $t+r+\log n$ ฟังก์ชันศักย์จะเพิ่มขึ้นอย่างมาก $(\log n)-t$ ต้นทุนถัวเฉลี่ยเป็น $t+r+\log n+\log n-t = 2(\log n)+r$ ที่ซึ่ง $r \leq \log n$ นั่นคือ ต้นทุนถัวเฉลี่ยเป็น $O(\log n)$ [11]

อัลกอริทึมการลบมีดังต่อไปนี้ [14]

LAZY-BINOMIAL-HEAP-EXTRACT-MIN(H)

1. z ← MIN[H]
2. if z ≠ NIL
3. then for each child x of z
4. do add x to the root list of H
5. p[X] ← NIL
6. remove z from the root list of H
7. if z = right[Z]
8. then min[H] ← NIL
9. else min[H] ← right[Z]
10. CONSOLIDATE(H)
11. n[H] ← n[H] − 1
12. return z

CONSOLIDATE(H)

1. for i ← 0 to D(n[H])
2. do A[i] ← NIL
3. for each node (O) in the root list of H
4. do X ← O
5. d ← degree[X]
6. while A[d] ≠ NIL
7. do Y ← A[d]
8. if key[X] > key[Y]
9. then exchange X <---> Y

10. *LAZY-BINOMIAL-HEAP_LINK(H, Y, X)*

11. *A[d]* $\leftarrow$ *NIL*

12. *d* $\leftarrow$ *d + 1*

13. *A[d]* $\leftarrow$ *x*

14. *min[H]* $\leftarrow$ *NIL*

15. *for I* $\leftarrow$ *0 to D(n[H])*

16. *do if A[I]* $\neq$ *NIL*

17. *then add A[I] to the root list of H*

18. *if min[H] = NIL or key[A[I]] < key[min[H]]*

19. *then min[H]* $\leftarrow$ *A[I]*

LAZY-BINOMIAL–HEAP–LINK(H,Y,X)

1. remove Y from the root list of H

2. make Y a child of X, incrementing degree[X]

3. mark[Y] $\leftarrow$ *FALSE*

5. การลดค่าของโหนด

การลดค่าของโหนดเป็นดังนี้ โดยสมมติให้โหนด x เป็นโหนดที่จะถูกลดค่า [14]

1. ทำการลดค่าของโหนด x
2. ถ้าการลดค่าก่อให้เกิดการเปลี่ยนแปลงลำดับในฮีพ กล่าวคือถ้าโหนด x มีค่าน้อยกว่าโหนดพ่อของโหนด x แล้ว
   - 2.1 ให้สลับ โดยให้โหนด x ขึ้นไปแทนที่พ่อของโหนด x ส่วนพ่อของโหนด x ลงมาแทนที่โหนด x
   - 2.2 ย้อนกลับไปทำข้อ 2 จนกระทั่งโหนด x เป็นรากของต้นไม้ หรือโหนด x มีค่ามากกว่าโหนดพ่อของโหนด x

การลดค่าของโหนดใช้เวลา O(log *n*) เนื่องจากต้นไม้สูงอย่างมาก $\lfloor \log n \rfloor$ ดังอัลกอริทึมต่อไปนี้ โดยสมมติให้ ค่าของโหนด x จะถูกลดค่าเป็นค่า k [14]

อัลกอริทึมการลดค่าโหนดมีดังต่อไปนี้ [14]

LAZY-BINOMIAL-HEAP-DECREASE-KEY(H, x, k)

1. *key[x]* $\leftarrow$ *k*

2. *y* $\leftarrow$ *x*

3. *z* $\leftarrow$ *p[y]*

4. *while z* $\neq$ *NIL and key[y] < key[z]*

5. *do exchange key[y] ... key[z]*

6. $y \leftarrow z$

7. $z \leftarrow p[y]$

8. *end do*

4. ฟิโบนักชี่สีพ

ฟิโบนักชี่สีพ จัดได้ว่าเป็นส่วนขยายของเลขชี่ไบโนมียลสีพ กล่าวคือเป็นรายการของต้นไม้ที่เรียงลำดับแบบสีพ โดยต้นไม้แต่ละต้นมีรากเป็นโหนดที่มีค่าน้อยที่สุด และต้นไม้ในฟิโบนักชี่สีพจะไม่มีข้อจำกัดดังเช่นต้นไม้ไบโนมียล นอกจากนี้ในฟิโบนักชี่สีพยังสามารถมีต้นไม้ที่มีความสูงเท่ากันได้หลายต้น [14]

สำหรับในการวิเคราะห์ถัวเฉลี่ย จะใช้วิธีศักย์ โดยสมมติให้ฟิโบนักชี่สีพ H มีจำนวนต้นไม้ $t(H)$ ต้น และมีจำนวนโหนดที่ถูกมาร์ค $m(H)$ โหนด ดังนั้นฟังก์ชันศักย์ ($\Phi(H)$) เป็น [14]

$$\Phi(H) = t(H) + 2m(H)$$

การปฏิบัติการของฟิโบนักชี่สีพ

1. การหาโหนดที่มีค่าน้อยที่สุด

การหาโหนดที่มีค่าน้อยที่สุดใช้เวลา O(1) โดยการจำและปรับเปลี่ยนตำแหน่งของโหนด ที่มีค่าน้อยสุดไว้ตลอดเวลาเมื่อมีการเปลี่ยนแปลงเกิดขึ้นระหว่างการปฏิบัติการต่างๆ [14]

2. การเพิ่มข้อมูล

การเพิ่มข้อมูล ทำได้โดยการสร้างสีพใหม่ที่มีข้อมูลเพียง 1 โหนด จากนั้นนำไปต่อกับรายการต้นไม้ของสีพเดิม เวลาที่ใช้ในการเพิ่มจึงคงที่เพียง O(1) สามารถคิดต้นทุนที่ถัวเฉลี่ย($\hat{c}_i$) ได้ดังนี้

$$\hat{c}_i = 1 + (t(H) + 1 + 2m(H)) - (t(H) + 2m(H)) = 1 + 1 = 2$$

หลังจากการเพิ่มโหนดเข้าไปในสีพ ต้นไม้จะเพิ่มขึ้นในสีพ 1 ต้น แต่จำนวนโหนดที่ถูกมาร์คจะไม่มีการเปลี่ยนแปลง [14]

อัลกอริทึมการเพิ่มข้อมูลเป็นดังนี้ [14]

FIB–HEAP–INSERT (H,X)

1. *degree [X]* $\leftarrow 0$
2. *p[X]* $\leftarrow NIL$
3. *child [X]* $\leftarrow NIL$
4. *left [X]* $\leftarrow X$
5. *right [X]* $\leftarrow X$

6. *mark [X]* $\leftarrow$ *FALSE*
7. *concatenate the root list containing X with root list H*
8. *if min [H] = NIL or key[X] < key[min[H]]*
9. *then min[H]* $\leftarrow X$
10. *n[H]* $\leftarrow n[H] + 1$

3. การรวมสีพ

การรวมสีพ เช่นเดียวกับแซ่ไบโนมียลสีพ ทำได้โดยการสร้างสีพใหม่ที่ประกอบไปด้วยต้นไม้ในสีพแต่ละสีพมาต่อกัน โดยในการรวมสีพในฟิโบนักชี่สีพนั้นไม่จำเป็นต้องทำการเชื่อมต้นไม้ที่มีดีกรีเท่ากัน ใช้เวลาเพียง O(1) สามารถคิดต้นทุนถัวเฉลี่ยได้ดังนี้

$$\hat{c}_i = 1 + (t(H) + 2m(H)) - (t(H_1) + 2m(H_1) + (t(H_2) + 2m(H_2))$$

เนื่องจาก $t(H_1) = t(H_2)$ และ $m(H) = m(H_1) + m(H_2)$ ดังนั้น $\hat{c}_i = 1 + 0 = 1$ นั่นคือ หลังจากการรวมสีพ 2 สีพ จำนวนต้นไม้ และจำนวนโหนดที่ถูกมาร์คจะไม่มีการเปลี่ยนแปลง สังเกตหลังจากการรวมสีพแล้ว จะมีต้นไม้ที่มีความสูงเท่ากันอยู่หลายต้น [14]

อัลกอริทึมการรวมเป็นดังนี้ [14]

FIB–HEAP–UNION (H1, H2)

1. *H* $\leftarrow$ *MAKE FIB HEAP ( )*
2. *Min[H]* $\leftarrow$ *min[H$_1$]*
3. *Concatenate the root list of H$_2$ with the root list of H*
4. *if (min[H$_1$] = NIL) or (min[H$_2$] $\neq$ NIL and min[H$_2$] < min[H$_1$]*
5. *then min[H]* $\leftarrow$ *min[H$_2$]*
6. *n[H]* $\leftarrow$ *n[H$_1$]+n[H$_2$]*
7. *free the objects H$_1$ and H$_2$*
8. *return H*

4. การลบโหนดที่มีค่าน้อยที่สุด

การลบโหนดที่มีค่าน้อยที่สุด มีขั้นตอนดังนี้ [14]

1. ลบโหนดที่มีค่าน้อยสุดออก
2. สร้างสีพใหม่ ซึ่งประกอบไปด้วยรายการของต้นไม้ย่อยของโหนดที่ถูกลบออก
3. ทำการเชื่อมรายการของต้นไม้ย่อยที่มีความสูงเท่ากัน เพื่อให้ได้ผลลัพธ์เป็นสีพที่ไม่มีต้นไม้ความสูงเท่ากันเกิน 1 ต้น

ให้ r เป็น ระดับความสูงของต้นไม้ที่มีโหนดที่มีค่าน้อยที่สุด

t เป็น จำนวนต้นไม้ในฮีพ

ฟังก์ชันศักย์ก่อนการดำเนินงานการลบเป็น $t+2m$ หลังจากทำการลบโหนดที่มีค่าน้อยที่สุดออก มีต้นไม้ย่อยที่แตกออกมา $r+r$ ต้น ซึ่งในการรวมต้นไม้เหล่านี้ ต้นทุนจริงเป็น $t+r+\log n$ ฟังก์ชันศักย์จะเพิ่มขึ้นอย่างมาก $((\log n)+2m)-(t+2m) = \log n - t$ ต้นทุนถัวเฉลี่ยเป็น $t+r+\log n+\log n-t = 2(\log n)+r$ สำหรับฟิโบนักชี่นั้น $r \leq \log n$ นั่นคือ ต้นทุนถัวเฉลี่ยเป็น $O(\log n)$ [14]

อัลกอริทึมการลบเป็นดังนี้ [14]

FIB-HEAP-EXTRACT-MIN(H)

```
1.  Z ← MIN[H]
2.  if Z ← NIL
3.        then for each child x of z
4.                 do add x to the root list of H
5.                        p[X] ← NIL
6.                 remove z from the root list of H
7.                 if z = right[Z]
8.                        then min[H] ← NIL
9.                        else min[H] ← right[Z]
10.                              CONSOLIDATE(H)
11.                       n[H] ← n[H]  1
12. return z
```

CONSOLIDATE(H)

```
1.  for I ← 0 to D(n[H])
2.        do A[I] ← NIL
3.  for each node (O in the root list of H
4.        do X ← (O
5.           d ← degree[X]
6.           while A[d] ≠ NIL
7.                 do Y ← A[d]
8.                    if key[X] > key[Y]
9.                           then exchange X <---> Y
10.                   FIB  HEAP- LINK(H, Y, X)
11.                   A[d] ← NIL
12.                   d ← d + 1
```

13. $A[d] \leftarrow x$

14. $min[H] \leftarrow NIL$

15. for $i \leftarrow 0$ *to* $D(n[H])$

16. do if $A[i] \neq NIL$

17. then add A[i] to the root list of H

18. if min[H] = NIL or key[A[i]] < key[min[H]]

19. then $min[H] \leftarrow A[i]$

FIB-HEAP-LINK(H,Y,X)

1. remove Y from the root list of H

2. make Y a child of X, incrementing degree[X]

3. $mark[Y] \leftarrow FALSE$

5. การลดค่าของโหนด

การลดค่าของโหนด เป็นดังนี้ โดยสมมติให้โหนด X เป็นโหนดที่จะถูกลดค่า [14]

1. ทำการลดค่าของโหนด
2. ถ้าการลดค่าก่อให้เกิดการเปลี่ยนแปลงลำดับในฮีพ กล่าวคือถ้าโหนด X มีค่าน้อยกว่าโหนดพ่อของโหนด X แล้วให้ทำการตัดโหนด X ออกจากโหนดพ่อของโหนด X
3. โหนด X จะกลายเป็นรากในรายการของต้นไม้ในฮีพ
4. ถ้า X เป็นลูกตัวแรกที่ถูกตัดจากโหนดพ่อของโหนด X แล้วทำการมาร์คโหนดพ่อของโหนด X
5. ถ้า X เป็นไม่ใช่ลูกตัวแรกที่ตัดออกจากโหนดพ่อของโหนด X กล่าวคือ โหนดพ่อของโหนด X เป็นโหนดที่ถูกมาร์คก่อนแล้ว จะต้องทำการตัดโหนดพ่อของโหนด X ออก แล้วโหนดพ่อของโหนด X จะกลายเป็นราก เป็นต้นไม้ใหม่ แล้วทำการยกเลิกการมาร์คที่โหนดพ่อของโหนด X
6. ย้อนกลับไปทำข้อ 5 จนกระทั่งโหนด X เป็นรากในรายการของต้นไม้ในฮีพ

ในการวิเคราะห์ถัวเฉลี่ย สมมติให้ Cut เป็น จำนวนการตัดโหนดขณะทำการลดค่าโหนด $t(H)$ เป็นจำนวนต้นไม้ในฮีพ $m(H)$ เป็นจำนวนโหนดที่ถูกมาร์ค ต้นทุนจริง (c) เป็นสัดส่วนโดยตรงกับจำนวนการตัดบวก 1 นั่นคือ $c = Cut + 1$ สำหรับฟังก์ชันศักย์นั้น ในแต่ละครั้งของการตัดโหนดนั้นมีการสร้างต้นไม้ใหม่ $(t(H)+Cut)$ และในแต่ละครั้งของการตัด โหนดที่ถูกมาร์คอยู่จะกลายเป็นโหนดที่ไม่ถูกมาร์คยกเว้นในการตัดครั้งสุดท้าย โดยในการตัดครั้งสุดท้ายนั้นจะมีการเพิ่ม

โหนดที่ถูกมาร์คอีกครั้งหนึ่งด้วย $(m(H)-(Cut-1)+1 = m(H)-Cut+2)$ นั่นคือความแตกต่างของฟังก์ชันศักย์ $(\Phi_i - \Phi_{i-1})$ เป็น

$$((t(H)+Cut)+2(m(H)-Cut+2))-(t(H)+2m(H)) = 4-Cut$$

ต้นทุนถัวเฉลี่ย จึงเป็น

$$Cut+1+4-Cut = 5$$

นั่นคือ เวลาที่ใช้ในการลดค่าของโหนด เป็น O(1) [14]

อัลกอริทึมการลดค่าของโหนดเป็นดังนี้ [14]

FIB–HEAP–DECREASE–KEY (H,X,K)

1. if K > key[X]
2. then error "new key is greater than current key"
3. key[X] ← K
4. Y ← p[X]
5. if Y ≠ NIL and key[X] < key[Y]
6. then CUT (H, X, Y)
7. CASCADING–CUT (H, Y)
8. if key[X] < key[min[H]]
9. then min[H] ← X

CUT (H, X, Y)

1. remove X from the child list of Y, decrementing degree[Y]
2. add X to the root list of H
3. p[X] ← NIL
4. mark[X] ← FALSE

CASADING–CUT (H, Y)

1. Z ← p[Y]
2. if Z ≠ NIL
3. then if mark[Y] = FALSE
4. then mark[Y] ← TRUE
5. else CUT (H, Y, Z)
6. CASADING–CUT (H, Z)

5. สถิติวิจัย

สลิวสีพ เป็นต้นไม้ไบนารีที่เรียงลำดับแบบสีพ และต้นไม้ในสีพจะไม่มีข้อจำกัดในด้านโครงสร้าง

นิยาม โหนด p จะเป็นโหนดเฮฟวี่ (heavy) ถ้าจำนวนลูก ๆ ทางขวาของโหนด p มีจำนวนอย่างน้อยครึ่งหนึ่งของจำนวนลูกทั้งหมดของโหนด p ในทางกลับกัน ถ้าจำนวนลูก ๆ ทางซ้ายของโหนด p มีจำนวนอย่างน้อยครึ่งหนึ่งของจำนวนลูกทั้งหมดของโหนด p แล้วโหนด p จะเป็นโหนดไลท์ (light)

ในการวิเคราะห์ถัวเฉลี่ยจะใช้วิธีศักย์ โดยให้ฟังก์ชันศักย์เป็นสัดส่วนโดยตรงกับจำนวนโหนดเฮฟวี่

การปฏิบัติการของสลิวสีพ

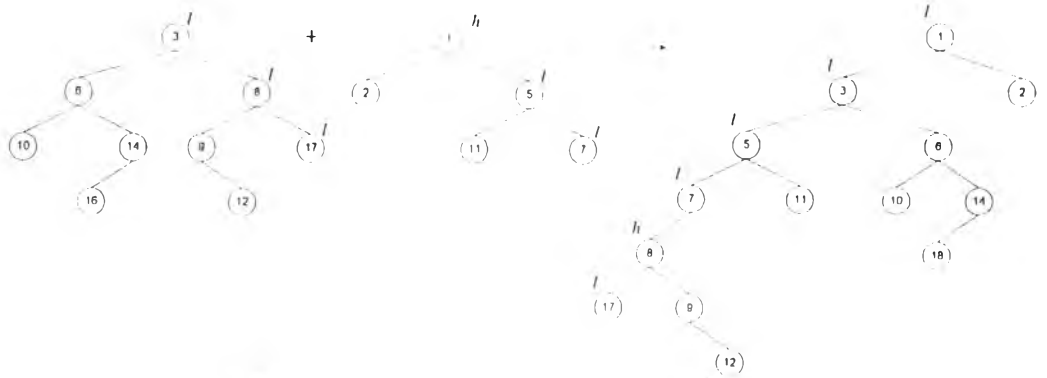
1. การรวมสีพ

การรวมของ 2 สลิวสีพ ทำได้โดย

1. ให้สีพหนึ่งไปรวมจากการลงล่างไปตามส่วนทางขวาของต้นไม้ในอีกสีพหนึ่ง โดยให้มีการเรียงลำดับ
2. ทำการสลับส่วนทางขวาให้ไปอยู่ทางซ้าย และส่วนทางซ้ายให้ไปอยู่ส่วนทางขวา โดยไล่สลับจากล่างขึ้นบนไปตามส่วนทางขวาของต้นไม้ ยกเว้นโหนดขวาสุด ไม่ต้องสลับเพราะจะเป็นโหนดที่ไม่มีลูกทางขวาให้สลับ

เนื่องจากสลิวสีพเป็นต้นไม้ที่ไม่มีข้อจำกัดในด้านโครงสร้าง จึงเป็นไปได้ที่ทุกๆ โหนดในสีพเอียงไปอยู่ทางด้านขวาของต้นไม้ กรณีนี้ถือเป็นกรณีร้ายแรงสุดที่เวลาที่ใช้ในการรวมสีพเป็น $O(n)$

ในการวิเคราะห์ถัวเฉลี่ย สมมติให้ H_1 และ H_2 เป็นสีพที่มี n_1 และ n_2 โหนดตามลำดับ โดยสมมติว่า H_1 มีโหนดไลท์อยู่ทางขวา l_1 โหนด และมีโหนดเฮฟวี่ อยู่ทางขวาเช่นกัน h_1 โหนด และ h_2 มีโหนด ไลท์ อยู่ทางขวา l_2 โหนด และมีโหนด เฮฟวี่ อยู่ทางขวาเช่นกัน h_2 โหนด นั่นคือ ต้นทุนจริงของการรวมเป็น $l_1+l_2+h_1+h_2$ ในส่วนทางขวา ถ้าโหนดใดเป็นโหนดเฮฟวี่แล้วหลังจากการรวม โหนดนั้นจะกลายเป็นโหนดไลท์ ส่วนโหนดอื่นๆที่เป็นโหนดไลท์หลังจากการรวมแล้วโหนดนั้นอาจจะเป็นหรือไม่เป็น โหนดเฮฟวี่ก็ได้ แต่ในกรณีร้ายแรงสุดโหนดที่เป็นโหนดไลท์หลังจากการรวมแล้วกลายเป็นโหนดเฮฟวี่ ซึ่งจะเป็นการเพิ่มฟังก์ชันศักย์ นั่นคือจำนวนของโหนดเฮฟวี่ จะเพิ่มขึ้นเป็น $l_1+l_2+h_1+h_2$ ดังนั้น ต้นทุนถัวเฉลี่ยเป็น $(l_1+l_2+h_1+h_2)+(l_1+l_2-h_1-h_2)=2(l_1+l_2)$ และเนื่องจาก l_1 และ l_2 เป็นจำนวนโหนดไลท์ในส่วนทางขวา และในจำนวนโหนดไลท์จะมีจำนวนโหนดน้อยกว่าครึ่งหนึ่งของจำนวนโหนดทั้งหมด ดังนั้น $l_1+l_2=\log n_1+\log n_2$ นั่นคือเวลาที่ใช้ในการรวมเป็น $O(\log n)$ ตัวอย่างดังรูปที่ 3.5



รูปที่ 3.5 แสดงการเปลี่ยนสถานะหลังการรวม โหนดเฮฟวี่/โหนดไลท์

อัลกอริทึมการรวมฮีพ เป็นดังนี้

MERGE (H1, H2)

// H_1 and H_2 are skew heaps

// Merge returns the merged heap, destroying H_1 and H_2

if H_1 is empty then return H_2

else if H_2 is empty then return H_1

if the root of $H_2 \leq$ the root of H_1 then swap H_1 and H_2

// now root(H_1) $\leq$ root(H_2)

if right(H_1) = nil

then right(H_1) $\leftarrow$ (H_2)

else

right(H_1) $\leftarrow$ Merge(right(H_1), H_2)

swap left and right children of H_1

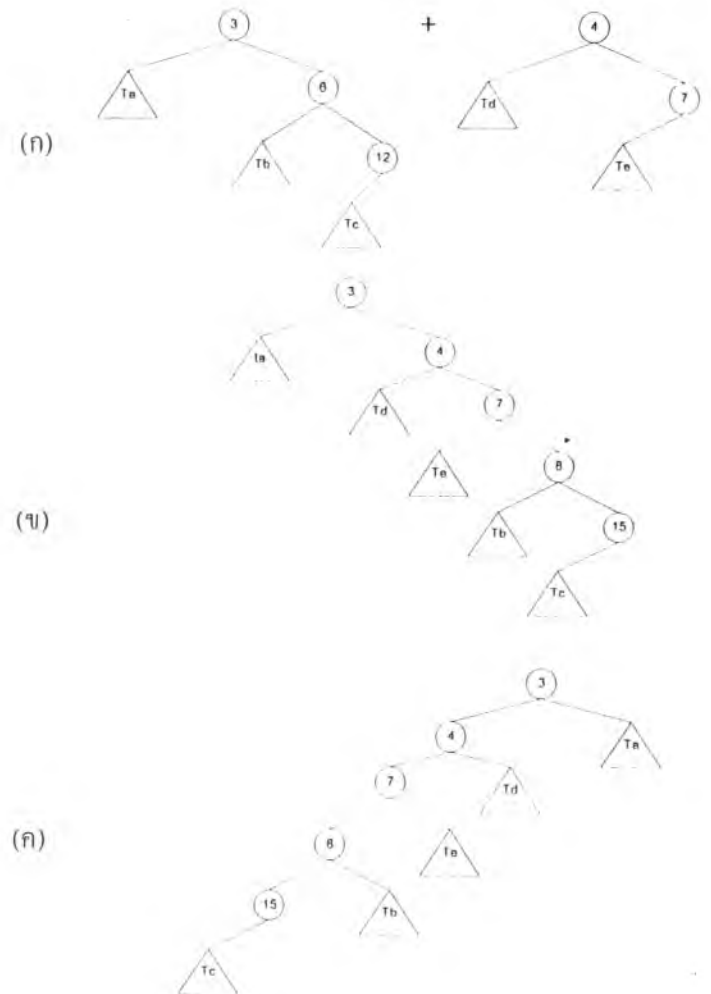
return H_1

ที่ซึ่งอัลกอริทึมนี้ ทำงานดังแสดงในรูปที่ 3.6

2. การเพิ่ม

การเพิ่มจะเป็นลักษณะเดียวกันกับการรวม คือจะเพิ่มโหนดในสคิวฮีพทางขวาแล้วไล่จากบนลงล่างไปตามส่วนทางขวาของต้นไม้ให้มีการเรียงลำดับ แล้วทำการสลับลูกโดยไล่สลับจากล่างขึ้นบนไปตามส่วนทางขวาของต้นไม้ ยกเว้นโหนดขวาสุดไม่ต้องสลับ เพราะเป็นโหนดที่ไม่มีลูกทางขวา สำหรับเวลาที่ใช้การรวม เป็น O(log n) เช่น เดียวกับการรวม[9]

3. การลบโหนดที่มีค่าน้อยที่สุด



รูปที่ 3.6 แสดงการรวมของ 2 สคิวฮีพ (ก) แสดง 2 สคิวฮีพก่อนที่จะทำการรวม (ข) แสดงผลหลังจากการรวม ก่อนที่จะมีการสลับ (ค) แสดงผลสุดท้ายของการรวม

การลบโหนดที่มีค่าน้อยที่สุด เนื่องจากสคิวฮีพ เป็นต้นไม้ไบนารี ดังนั้นการลบจะทำให้ต้นไม้แตกออกเป็น ต้นใหม่ 2 ต้น ซึ่งจะทำการรวมดังที่ได้กล่าวข้างต้น สำหรับการวิเคราะห์ถัวเฉลี่ยของการลบ เป็น $O(\log n)$ เช่นเดียวกับการรวม [9], [11]

5. การลดค่าของโหนด

การลดค่าของโหนดเป็นดังนี้ โดยสมมติให้โหนด x เป็นโหนดที่จะถูกลดค่า [14]

1. ทำการลดค่าของโหนด x
2. ถ้าการลดค่าก่อให้เกิดการเปลี่ยนแปลงลำดับในฮีพ กล่าวคือถ้าโหนด x มีค่าน้อยกว่าโหนดพ่อของโหนด x แล้ว
    2.1 ให้สลับ โดยให้โหนด x ขึ้นไปแทนที่พ่อของโหนด x ส่วนพ่อของโหนด x ลงมาแทนที่โหนด x

2.2 ย้อนกลับไปทำข้อ 2 จนกระทั่งโหนด x เป็นรากของต้นไม้ หรือโหนด x มีค่ามากกว่าโหนดพ่อของโหนด x

การลดค่าของโหนดใช้เวลา $O(\log n)$ เนื่องจากต้นไม้สูงอย่างมาก $\lfloor \log n \rfloor$ ดังอัลกอริทึมต่อไปนี้ โดยสมมติให้ ค่าของโหนด x จะถูกลดค่าเป็นค่า k [14]

อัลกอริทึมการลดค่าโหนดมีดังต่อไปนี้ [14]

SKEW-DECREASE-KEY(H, x, k)

1. *key[x]* ← *k*
2. *y* ← *x*
3. *z* ← *p[y]*
4. *while z* ≠ *NIL and key[y] < key[z]*
5. *do exchange key[y] — key[z]*
6. *y* ← *z*
7. *z* ← *p[y]*
8. *end do*