Vlist

DATA STRUCTURES

Vlist คือ

วีลิสต์ (Vlist) เป็นโครงสร้างข้อมูลที่ดัดแปลงจาก Linked list ทางเดียว หรือ Singly-Linked list และใช้อาเรย์เป็นตัวเก็บข้อมูลแทนการเก็บข้อมูลแบบทั่วไป ทำให้การเข้าถึงข้อมูลที่ตำแหน่งใด ๆ ทำได้เร็วกว่า Linked list ทั่วไป

Vlist คือ

วีลิสต์ (Vlist) เป็นการรวมกันของโครงสร้างอาเรย์กับลิงค์ลิตส์

อาเรย์

ข้อดี

1. เข้าถึงข้อมูลได้ง่ายและรวดเร็ว
2. กำหนดข้อมูลได้ง่าย

Vlist คือ

อาเรย์

ข้อเสีย

1. ขนาดของอาเรย์นั้นตายตัวไม่สามารถปรับเปลี่ยนได้
2. การ insert และ delete ข้อมูล จำเป็นต้องย้ายข้อมูลจำนวนมาก
3. อาเรย์มีการจองพื้นที่หน่วยความจำให้สมาชิกทุกตัวติดกันเป็นบล็อค

Vlist คือ

ลิงค์ลิสต์

ข้อดี

1. ขนาดของลิงค์ลิงค์ลิสต์สามารถปรับเปลี่ยนได้อย่างอิสระ
2. การเก็บข้อมูลของลิสต์ไม่ต้องจองพื้นที่หน่วยความจำติดกัน เพราะมีลิงค์เชื่อมต่อไปยังข้อมูลตัวถัดไปอยู่แล้ว
3. การเพิ่มและลบข้อมูลทำได้ง่าย

Vlist คือ

ลิงค์ลิสต์

ข้อเสีย

1. การเข้าถึงข้อมูลไม่สามารถข้ามได้ ต้องเข้าถึงเป็นลำดับไปเรื่อย ๆ จนถึงข้อมูลที่ต้องการเข้าถึง

2. มีความซับซ้อนในการกระทำต่าง ๆ กว่าอาเรย์

3. ต้องใช้พื้นที่หน่วยความจำเก็บค่าของตัวชี้ด้วย

ประโยชน์ Vlist

วีลิสต์ (Vlist) ใช้ในการเขียนโปรแกรมเชิงฟังก์ชัน(functional programming languages) และนำไปใช้ในการสร้างโครงสร้างข้อมูลแบบ persistent data structure

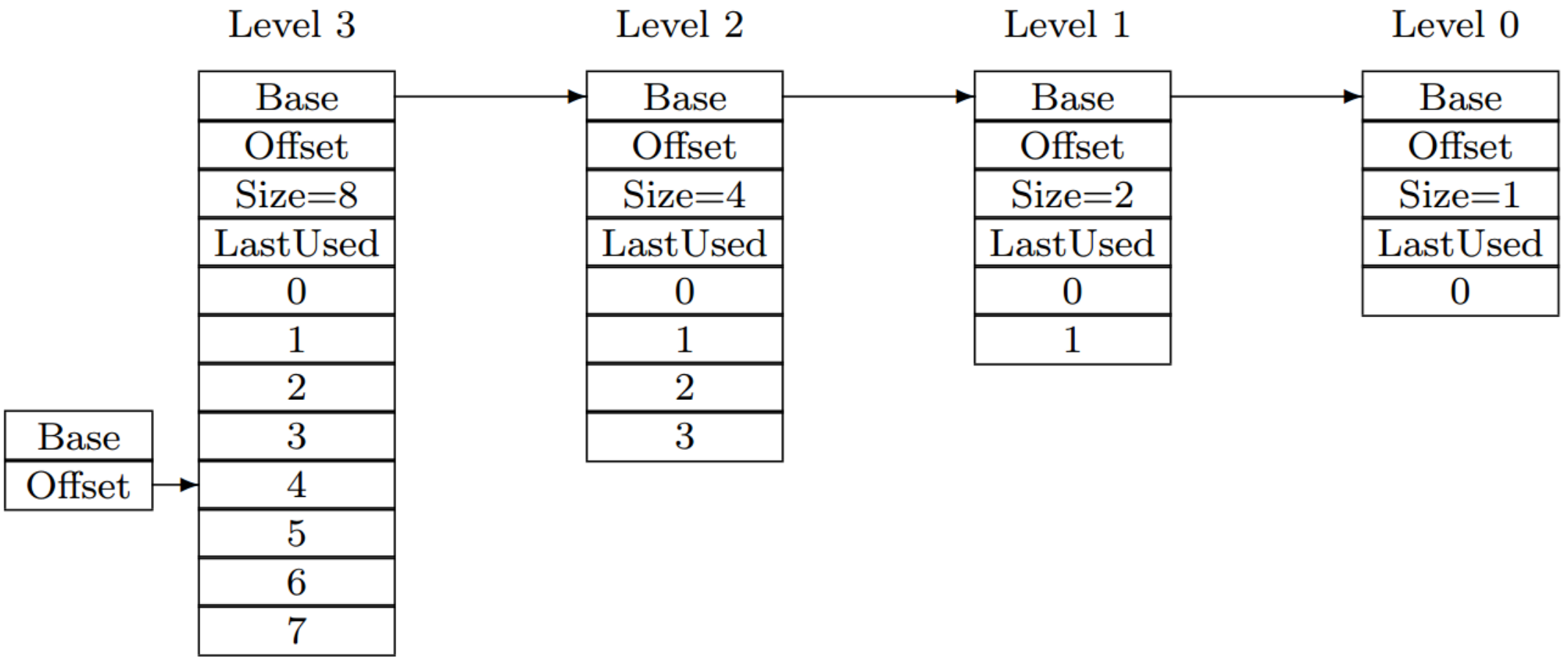
หลักการ Vlist

วีลิสต์ (Vlist) แทนที่จะโยงต่อกันด้วยโหนดตัวเดียวแบบลิงค์ลิสต์ วีลิสต์ใช้การโยงของก้อนข้อมูล (memory block) ซึ่งประกอบด้วยอาเรย์ที่สามารถเก็บข้อมูลได้หลายตัวมาโยงต่อกัน โดยมีฐาน (base) เป็นตัวชี้ (pointer) ไปยังก้อนข้อมูลก่อนหน้า และมีออฟเซ็ต (offset) เป็นตัวอ้างอิงตำแหน่งปัจจุบันของข้อมูลที่เทียบจากรายการและตำแหน่งล่าสุด (last used) ซึ่งเทียบตำแหน่งของข้อมูลจากอาเรย์ปัจจุบันที่ข้อมูลตัวนั้นอยู่

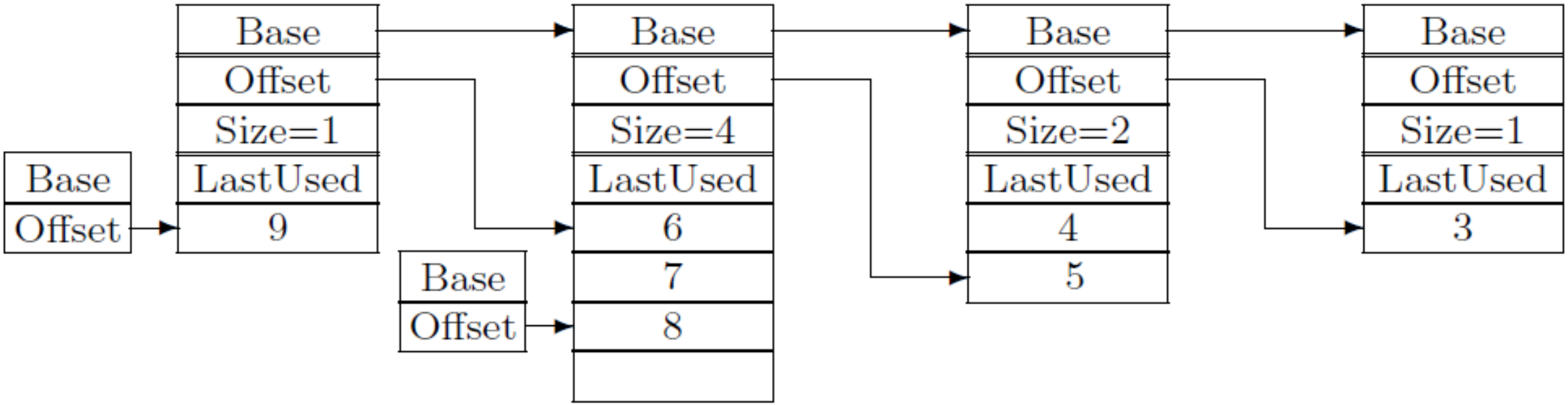
หลักการ Vlist

ก้อนข้อมูลยังมีตัวแปรเก็บขนาดสูงสุดของอาเรย์ปัจจุบัน และทุก ๆ ครั้งที่ข้อมูลเต็มก้อนข้อมูลอันเก่า เมื่อเพิ่มก้อนข้อมูลใหม่ ก้อนข้อมูลใหม่จะมีขนาดเป็น **r** เท่าของก้อนข้อมูลเดิม และก้อนข้อมูลก่อน ๆ จะมีขนาดลดลงเป็น **r** เท่า

หลักการVlist



หลักการVlist



การเพิ่มข้อมูล Vlist

การเพิ่มข้อมูลเข้าที่ตำแหน่งหน้าสุดของวีลิสต์ (ตำแหน่งสุดท้ายของรายการ) จะต้องตรวจสอบว่าอาเรย์ของข้อมูลในก้อนข้อมูล (memory block) ปัจจุบันนั้นเต็มหรือยัง ถ้าไม่เต็มก็เพิ่มข้อมูลลงในตำแหน่งถัดไป และเพิ่มค่าของตำแหน่งล่าสุด (last used) ในก้อนข้อมูลปัจจุบัน

การเพิ่มข้อมูล Vlist

ถ้าอาเรย์ของข้อมูลเต็มแล้วต้องทำการสร้างก้อนข้อมูลตัวใหม่ที่มีอาเรย์เป็นขนาด **r** เท่าของก้อนข้อมูลตัวเดิม แทรกเข้าไปที่ตำแหน่งหน้าของก้อนข้อมูลตัวก่อน (ตำแหน่งท้ายสุดของรายการ) และให้ออฟเซ็ต (offset)ของก้อนข้อมูลตัวล่าสุดเก็บตำแหน่งสุดท้ายของรายการจากก้อนข้อมูลอันก่อน

การเพิ่มข้อมูล Vlist

การเพิ่มข้อมูลเข้าไปสู่ตำแหน่งอื่นที่ไม่ใช่ตำแหน่งหน้าสุดของวีลิสต์ จำเป็นต้องทำการสร้างวีลิสต์ตัวใหม่ที่ชี้ไปยังตำแหน่งที่ต้องการจะเพิ่มข้อมูล เมื่อเพิ่มข้อมูลต้องเพิ่มข้อมูลที่เหลือจากวีลิสต์อันเก่าเข้าไป

การลบข้อมูล Vlist

ในการลบข้อมูลจากตำแหน่งหน้าสุดของวีลิสต์ (ตำแหน่งท้ายสุดของรายการ) ทำได้โดยลบตำแหน่งล่าสุด (last used) ที่อ้างอิงอาเรย์ของข้อมูลลงหนึ่งค่า ซึ่งการลบข้อมูลด้วยวิธีนี้จะไม่คืนหน่วยความจำที่จองไว้ แต่เมื่อมีการเพิ่มข้อมูลตัวใหม่จะเขียนทับตำแหน่งเดิม เมื่อตำแหน่งอ้างอิงของอาเรย์ข้อมูลในก้อนข้อมูลปัจจุบันมีค่าติดลบ การให้ตัวชี้ชี้ไปยังก้อนข้อมูล (memory block) ถัดไป ก้อนข้อมูลที่ถูกลบนั้นจะถูกเก็บกวาดด้วยตัวเก็บขยะ (Garbage Collection)

การลบข้อมูล Vlist

ในการลบข้อมูลที่ตำแหน่งใดๆที่ไม่ใช่ตำแหน่งหน้าสุดต้องมีการสร้างวีลิสต์ตัวใหม่ให้ชี้ไปยังข้อมูลตำแหน่งก่อนหน้าข้อมูลที่จะทำการลบ แล้วเพิ่มข้อมูลจากวีลิสต์ตัวเก่าโดยไม่เพิ่มข้อมูลจากตำแหน่งที่ต้องการลบ

การเข้าสู่ตำแหน่งใด ๆ Vlist

การเข้าสู่ตำแหน่งที่ n ใดๆของวีลิสต์ จะเริ่มด้วยการนำ n ลบกับออฟเซ็ต (offset) ของก้อนข้อมูลปัจจุบันถ้าเป็นบวก แสดงว่าตำแหน่งนั้นอยู่ที่ตำแหน่งของอาเรย์ตำแหน่งที่ n - offset ถ้า n ลบกับออฟเซ็ตเป็นลบ แสดงว่าตำแหน่งนั้นอยู่ในอาเรย์ข้อมูลของก้อนข้อมูล (memory block) ถัดๆไป ซึ่งเราสามารถหาได้โดยลบกับออฟเซ็ตของก้อนข้อมูลถัดไปๆ เรื่อยๆ จนลบแล้วได้ค่าของ n ลบกับออฟเซ็ตแล้วเป็นบวก ตำแหน่งนั้นคือตำแหน่งของอาเรย์ที่ n - offset ของก้อนข้อมูลตัวนั้น

ประสิทธิภาพการทำงาน Vlist

- เนื้อที่ในการเก็บตัวชี้จะใช้เนื้อที่ O(log n) เพราะการโยงข้อมูลของแต่ละก้อนข้อมูลใช้ตัวชี้เพียงตัวเดียว และข้อมูลได้อยู่เป็นกลุ่มในแต่ละก้อนข้อมูลลดลงกันไปก้อนละ r เท่า
- การเพิ่มข้อมูลข้างหน้าของวีลิสต์ใช้เวลา O(1)
- การลบข้อมูลที่อยู่ข้างหน้าของวีลิสต์ใช้เวลา O(1)
- การนับจำนวนข้อมูลในวีลิสต์ใช้เวลา O(log n)
- การเข้าสู่ตำแหน่งใดๆของวีลิสต์ใช้เวลาเฉลี่ย O(1) ในกรณีที่ช้าที่สุดใช้เวลา O(log n)

ประสิทธิภาพการทำงาน Vlist

เนื่องจาก 50% ของข้อมูลทั้งหมดอยู่ที่ก้อนข้อมูลอันแรกแล้ว 75% อยู่ในก้อนข้อมูลอันแรกและอันที่สองรวมกัน ซึ่งในกรณีที่ช้าที่สุดคือตำแหน่งที่ต้องการอยู่ในก้อนข้อมูลอันสุดท้าย ต้องผ่านก้อนข้อมูลไปจำนวน n/2^i เมื่อค่า r คือ 2 นั้นก็คือ log n เมื่อคิดในกรณีเฉลี่ย การเข้าสู่ตำแหน่งใดๆ จะได้ตามสมการนี้

$$\sum_{i=1}^{\lceil log_2 n \rceil} \frac{i-1}{2^i} < \sum_{i=1}^{\infty} \frac{i-1}{2^i} = 1$$

การนำไปใช้สร้างโครงสร้างข้อมูลแบบอื่น ๆ

Vlist สามารถนำไปใช้เพื่อสร้างโครงสร้างข้อมูลแบบอื่น

- ตารางแฮช (Hash Table) โดยการแบ่งก้อนข้อมูลในมีส่วนของข้อมูลและส่วนของตารางแฮช โดยส่วนที่เป็นข้อมูลจะมีการโยงถึงตำแหน่งและข้อมูลก่อนหน้า เช่นเดียวกับส่วนที่เป็นตารางแฮชก็จะมีการโยงกับตารางแฮชก่อนหน้า ด้วยประสิทธิภาพของตารางแฮชทำให้การหาข้อมูลทำได้โดยเวลาคงที่

- แถวลำดับพลวัต (dynamic array)

- แถวคอยสองหน้า (Double-ended queue)

Vlist