โครงสร้างข้อมูล

แบบ

HASHING

สมาชิก

นางสาว ชัญญา กลางแก้ว ปวค.59/2 055950201069-2

นาย รพีภัทร หลงวิจิตร ปวค.59/2 055950201084-1

HASH

- เกี่ยวกับการจัดการข้อความหรือข้อมูลให้เป็นดัชนี เพื่อใช้อ้างอิงตำแหน่งการเก็บข้อมูลของอาร์เรย์หรือฐานข้อมูล เรียกรูปแบบการหาคีย์ดัชนีว่า การทำแฮช (Hash)
- เมื่อต้องการค้นหาข้อมูลให้เจอเพียงครั้งเดียว สามารถใช้รูปแบบการอ้างอิงตำแหน่งในการเก็บข้อมูลด้วยดัชนีเพื่อที่จะสามารถเข้าถึงข้อมูลได้เพียงครั้งเดียว โดยการหาคีย์ดัชนีเพื่อใช้อ้างอิงตำแหน่งในเก็บข้อมูล
- เมื่อพิจารณาอาร์เรย์ที่มีขนาด n ข้อมูล พบว่า ในแต่ละเรคอร์ดของอาร์เรย์สามารถเก็บข้อมูลได้เพียงหนึ่งข้อมูล ในการหาตำแหน่งเพื่อเพิ่มข้อมูลเข้าไปในอาร์เรย์เรียกว่า “การคำนวณหาแอดเดรส” (Address Calculator)
- แฮชฟังก์ชัน (Hash function) เป็นแนวความคิดในการคำนวณหาแอดรเดรสในการจัดการข้อมูลในอาร์เรย์และเรียกอาร์เรย์ในการเก็บข้อมูลแฮชว่า แฮชเทเบิล (Hash table)

ตัวอย่างการของแฮชฟังก์ชัน

- เมื่อโรงพยาบาลต้องการเก็บข้อมูลเบอร์โทรศัพท์บุคคลที่เข้ามารักษาพยาบาล แล้วกำหนดให้ดัชนีเพื่อใช้อ้างอิงตำแหน่งในการเก็บข้อมูลในอาร์เรย์ คือหมายเลขโทรศัพท์แทนด้วยตัวแปร num จะได้ว่าตำแหน่งในการเก็บข้อมูลบุคคลที่เข้ามารักษาพยาบาลคือ table[num]
- ใช้หมายเลขโทรศัพท์ 123-4567 เก็บข้อมูลในตำแหน่ง table[1234567] ซึ่งในกรณีนี้ต้องใช้การจองพื้นที่ในการเก็บข้อมูลทั้งหมดสิบล้านข้อมูลในการประกาศอาร์เรย์ table
- แต่ถ้าไม่ต้องการใช้พื้นที่ในหน่วยความจำเพื่อเก็บข้อมูลถึงสิบล้านข้อมูล และเมื่อพิจารณาเฉพาะหมายเลขโทรศัพท์เพียง 4 ตัวหลังมาเป็นตำแหน่งในการเก็บข้อมูล เช่นหมายเลขโทรศัพท์คือ 123-4567 จะเก็บในอาร์เรย์ตำแหน่งที่ table[4567] แทน ทำให้พื้นที่ในการเก็บข้อมูลเหลือเพียง 10,000 ข้อมูลในการเก็บข้อมูล

ตัวอย่างการของแฮชฟังก์ชัน

- การเปลี่ยนข้อมูลหมายเลขโทรศัพท์ในการเก็บข้อมูลในอาร์เรย์ในตำแหน่ง 1234567 เป็นการอ้างอิงตำแหน่งในการเก็บข้อมูลในอาร์เรย์ตำแหน่ง 4567 ในรูปแบบนี้เป็นตัวอย่างในการใช้แฮชฟังก์ชัน (Hash function)

- สัญลักษณ์ h แทนแฮชฟังก์ชัน และตัวแปร x คือ ตำแหน่งแอดเดรสที่ได้จากแฮชฟังก์ชันในการอ้างอิงตำแหน่งแอดเดรสในอาร์เรย์
- มีหมายเลขโทรศัพท์สองเบอร์คือ 1234567 และ 1114567 ตำแหน่งในการเก็บข้อมูลในอาร์เรย์อยู่ที่ตำแหน่งเดียวกันเนื่องจากกำหนดให้ตัวเลขสี่ตัวหลังเป็นตำแหน่งในการอ้างอิงตำแหน่งแอดเดรส คือ ตำแหน่งแอดเดรส 4567 ซึ่งเรียกรูปแบบการมีแอดเดรสที่ซ้ำกันนี้ว่า การชนกันข้อมูล (Collision data)

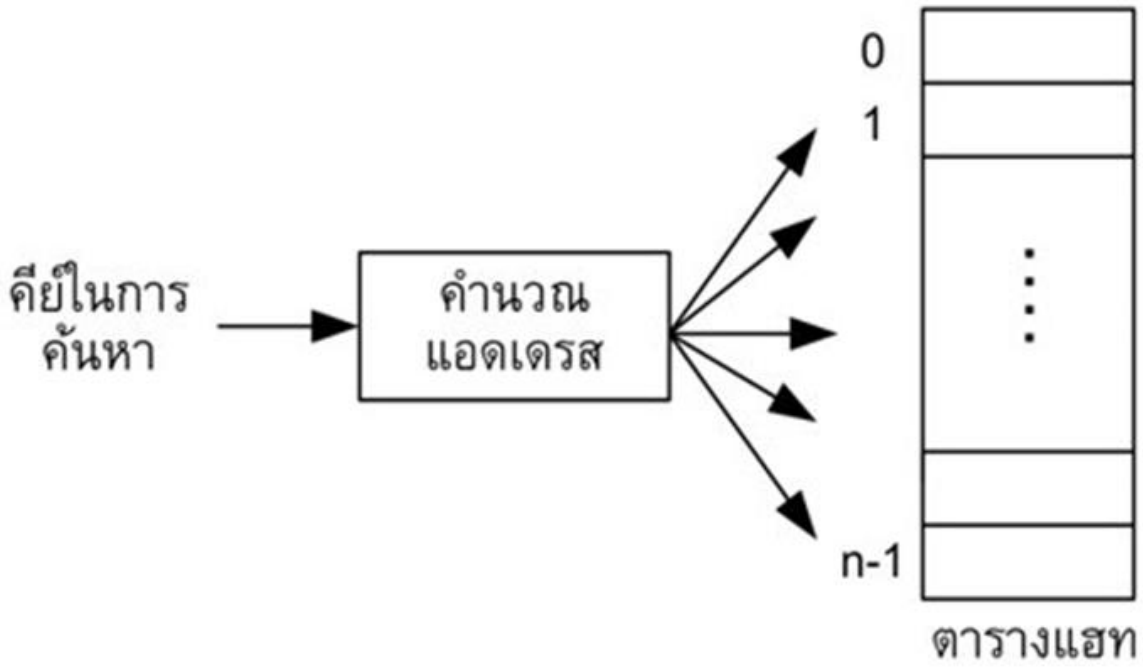
แฮชฟังก์ชัน (HASH FUNCTIONS)

แฮชฟังก์ชัน (Hash Functions) เป็นการจัดการเกี่ยวกับตัวเลขจำนวนเต็ม เช่น การเปลี่ยนตัวเลขที่หลากหลายให้เป็นตัวเลขที่อยู่ช่วงที่กำหนด, การกำหนดให้ข้อมูลที่หลากหลายให้มาเก็บอยู่ในช่วง 0 ถึง 100 เป็นต้น แต่ถ้าคีย์ในการค้นหาไม่ใช่ตัวเลขจำนวนเต็มก็สามารถเปลี่ยนรูปแบบให้เป็นเลขจำนวนเต็มได้เช่นกัน ซึ่งเป็นหลักการของการทำแฮชฟังก์ชัน

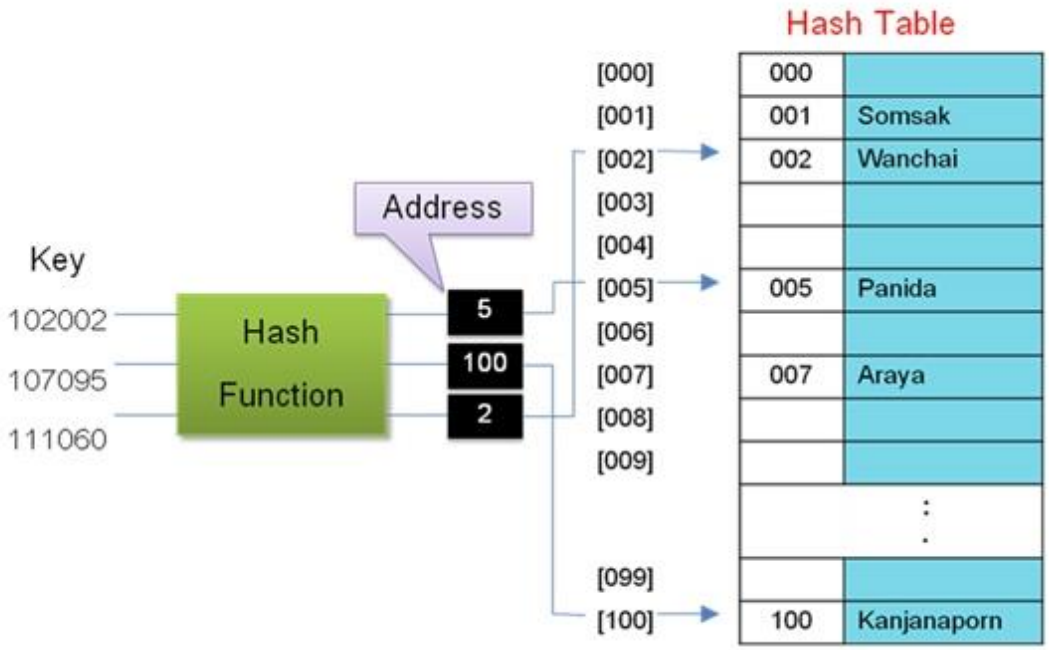
คุณสมบัติของ HASH FUNCTION ที่ดี

1. data เดียวกัน จะได้ค่า hash ที่เหมือนกัน
2. data ที่ต่างกันแม้เพียงเล็กน้อย จะได้ค่า hash ที่ต่างกันมาก
3. การคำนวณทำได้ง่าย ใช้เวลาคำนวณน้อย (เป็น function ที่ไม่ซับซ้อน)
4. ไม่สามารถคำนวณย้อนกลับจากค่า hash กลับมาเป็น data ได้

ภาพแสดงการนำคีย์มาผ่านฟังก์ชันแฮชเพื่อแอดเดรสซึ่งเป็นตำแหน่งที่อยู่ของข้อมูล



ตัวอย่างหลักการของแฮช (HASH CONCEPT)



ในการศึกษาเกี่ยวกับการค้นหาข้อมูลด้วยวิธีแฮชชิงนั้นจำเป็นจะต้องรู้ความหมายของคำศัพท์พื้นฐานต่อไปนี้ให้เข้าใจเสียก่อน ซึ่งประกอบด้วย

ตัวอย่างหลักการของแฮช (HASH CONCEPT) (ต่อ)

- คีย์ (Key) คือข้อมูลที่ต้องการนำไปค้นหา ซึ่งค่าของคีย์จะไม่มีการซ้ำกัน
- ฟังก์ชันแฮช(Hash Function) คือสูตรหรือฟังก์ชันที่ใช้สำหรับแปลงคีย์ให้เป็นตำแหน่งแอดเดรส เมื่อได้แอดเดรส แล้ว ก็สามารถนำแอดเดรสนี้เข้าถึงตำแหน่งข้อมูลที่ต้องการในตารางแฮชได้
- ตารางแฮช (Hash Table) คือตารางที่ใช้สำหรับเก็บข้อมูล

การเลือกหลัก (SELECTION DIGITS)

ถ้าคีย์ในการค้นหา คือ รหัสของลูกจ้างที่มีจำนวนเก้าหลัก เช่น 001364825 ถ้าเลือกหลักที่ 4 และหลักสุดท้าย จะได้เลข 35 เป็นตำแหน่งในอ้างอิงแอดเดรสในตารางแฮช h(001364825) = 35 (เลือกจากหลักที่สี่และหลักสุดท้าย)

ดังนั้น คีย์ในการค้นหาของข้อมูล 001354825 ในตารางแฮช คือ table[35]

ข้อควรระวังเกี่ยวกับการเลือกหลักที่นำมาทำแฮชฟังก์ชัน ตัวอย่างเช่น การเลือกหลักที่หนึ่ง และหลักที่สามมาเป็นแฮชฟังก์ชัน ซึ่งเป็นหลักที่มีหลายคนที่จะมีตัวเลขของหลักที่หนึ่งและหลักที่สามซ้ำกันได้เป็นจำนวนมาก

การบวกหลัก (FOLDING)

หลักการของการบวกหลัก คือ การเลือกหลักและนำตัวเลขในหลักนั้นมาบวกกัน จากตัวอย่างเช่น นำข้อมูลตัวตัวจากทุกหลักมาบวกกัน เช่น ข้อมูลคือ 001364825 จะได้ดังนี้

$0 + 0 + 1 + 3 + 6 + 4 + 8 + 2 + 5 = 29$

ดังนั้น คีย์ในการค้นหาของ 001364825 ในตารางแฮชคือ table[29]

การบวกหลัก (FOLDING) (ต่อ)

ข้อกำหนดของจากการบวกทุกหลักจากทั้งหมดเก้าหลักในแต่ละหลักจะมีข้อมูลได้คือ 0 ถึง 9 ดังนั้นตารางแฮชจะเก็บข้อมูลได้ระหว่าง 0 ถึง 81 ข้อมูล (0 < h(คีย์ในการค้นหา) < 81)

เมื่อเพิ่มขนาดของตารางแฮช จะต้องปรับรูปแบบของการบวก เช่น จัดกลุ่มของหลักและนำข้อมูลของหลักในแต่ละกลุ่มมาบวกกัน เป็นต้น ดังตัวอย่างข้อมูล 001364825 จะแบ่งข้อมูลออกเป็น 3 กลุ่ม แต่ละกลุ่มมี 3 หลัก และนำมาบวกกันได้ดังนี้

001 + 364 + 825 = 1,190

ดังนั้น แฮชฟังก์ชันจะเก็บข้อมูลอยู่ระหว่าง 0 < h(คีย์ในการค้นหา) < 3 * 999 = 2,997

การหารเอาเศษ (MODULATE ARITHMETIC)

หลักการหารเอาเศษเป็นรูปแบบที่ง่ายของแฮชฟังก์ชัน โดยมีรูปแบบ คือ

$$h(x) = x \bmod tableSize$$

เมื่อ tableSize เป็นขนาดของตารางแฮช table

ตัวอย่างเช่น ถ้า tableSize มีค่าคือ 101 จะได้ว่า $h(x) = x \bmod 101$ ทำให้ค่าตัวเลข x จะอยู่ในช่วง 0 ถึง 100 และจากตัวอย่างข้อมูล 001364825 จะทำให้มีค่าแฮชคือ 12

จากหลักการ $h(x) = x \bmod tableSize$ จะพบว่ามักจะมีข้อมูลแฮชในตำแหน่งที่ table[0] และตำแหน่งที่ table[1] ซ้ำกัน

การเปลี่ยนข้อความเป็นตัวเลข (CONVERTING A CHARACTER STRING TO AN INTEGER)

กรณีที่คีย์ในการค้นหาเป็นข้อความ เช่น การใช้ชื่อเป็นคีย์ในการค้นหา สามารถเปลี่ยนข้อความชื่อให้เป็นตัวเลขในแฮชฟังก์ชัน h(x) ได้ โดยสิ่งแรกที่ต้องทำคือเปลี่ยนตัวอักษรแต่ละตัวให้เป็นค่าตัวเลข ตัวอย่างเช่น ข้อความ “NOTE” เมื่อใช้รหัส ASCII ของตัวอักษรในการเปลี่ยนเป็นตัวเลข ได้เป็น 78, 79, 84 และ 69 ของตัวอักษร N, O, T และ E ตามลำดับ หรือสามารถใช้ค่าตัวเลขระหว่าง 1 ถึง 26 แทนตัวอักษร A ถึง Z ซึ่งจะได้ 14, 15, 20 และ 5 ของตัวอักษร N, O, T และ E

การเปลี่ยนข้อความเป็นตัวเลข

(CONVERTING A CHARACTER STRING TO AN INTEGER)

แทนตัวอักษร A ถึง Z ด้วยค่าตัวเลข 1 ถึง 26 จากข้อความ “NOTE” จะเปลี่ยนเป็นเลขฐานสองได้ดังนี้

N เท่ากับ 14 หรือ 01110 ในเลขฐานสอง

O เท่ากับ 15 หรือ 01111 ในเลขฐานสอง

T เท่ากับ 20 หรือ 10100 ในเลขฐานสอง

E เท่ากับ 5 หรือ 00101 ในเลขฐานสอง

เมื่อนำค่าตัวเลขฐานสองของข้อความ “NOTE” มาเรียงต่อกัน จะได้ 01110011111010000101 ซึ่งเท่ากับ 474,757 ในเลขฐานสิบ

การเปลี่ยนข้อความเป็นตัวเลข (CONVERTING A CHARACTER STRING TO AN INTEGER)

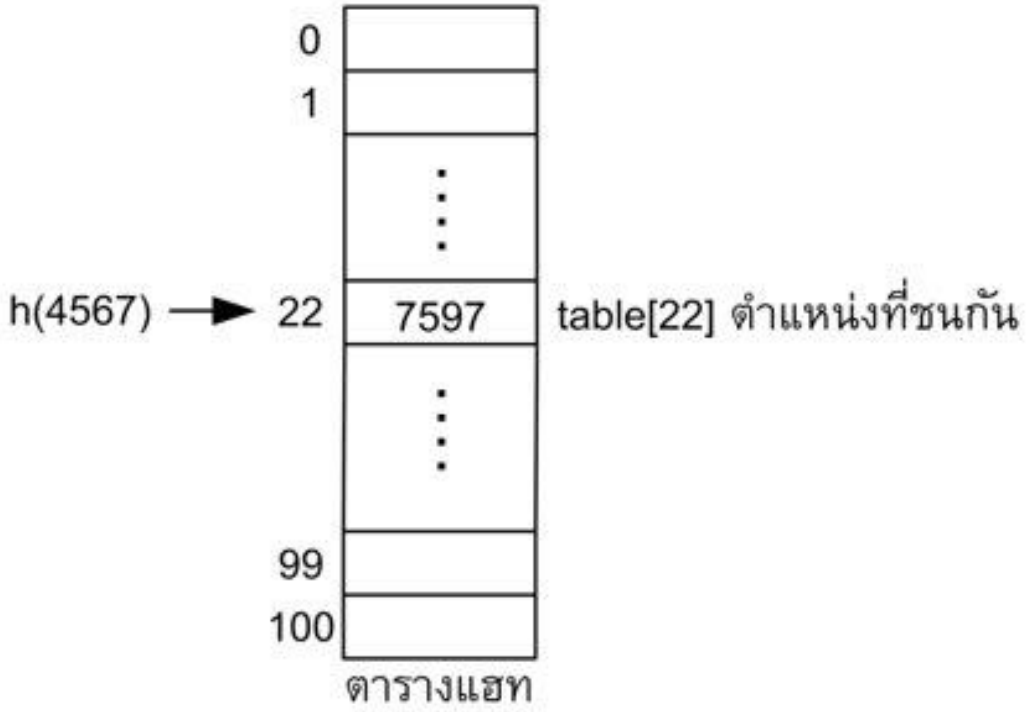
เป็นการยากในการเปลี่ยนข้อมูลเลขฐานสองจำนวนหลายหลักให้เป็นเลขฐานสิบ ดังนั้นได้มีหลักการของ Horner's rule ที่พิจารณาจำบิตของเลขฐานสองที่มีจำนวนบิตน้อยที่สุดมาใช้ในการเปลี่ยนข้อความเป็นตัวเลข (ตัวอักษร A ถึง Z เท่ากับ 26 ตัว) ซึ่งจะมีจำนวน 5 บิต ดังนั้น เลขฐานสองคือ 25 คือ 32 ดังนั้นเปลี่ยนตัวอักษรเป็นตัวเลขได้ดังนี้

$(14 * 323) + (15 * 322) + (20 * 321) + (5 * 320) = 474,757$

แล้วนำค่าเลขฐานสิบในการเปลี่ยนตัวอักษรเป็นตัวเลขไปใช้แฮชฟังก์ชัน $h(x) = x \bmod tableSize$ เพื่อหาค่าแอดเดรสในการอ้างอิงในตารางแฮช

การแก้ปัญหาการชนกันของแฮชคีย์ (RESOLVING COLLISION)

คีย์ในการค้นหาคือ 4567 เพิ่มเข้าไปในตารางแฮช table ที่มีขนาด tableSize เท่ากับ 101 (แฮชฟังก์ชันคือ h(x) = x mod 101) จะได้ตำแหน่งใหม่ในตารางแฮชเท่ากับ table[22] (ซึ่งได้จาก 4567 mod 101 เท่ากับ 22) ดังแสดงในรูป 5.2 และเมื่อเพิ่มคีย์ในการค้นหาคือ 7597 ตำแหน่งในการเพิ่มข้อมูลคือ table[22] เช่นกัน (ได้จาก 7597 mod 101 เท่ากับ 22) ซึ่งเป็นตำแหน่งที่ไม่อนุญาตให้เพิ่มข้อมูลในตารางแฮชเนื่องจากเป็นตำแหน่งที่ชนกัน



จากปัญหาการชนกันของแฮชคีย์ จะมีวิธีแก้ไขปัญหา 2 วิธี คือ

1.ใช้ตำแหน่งแอดเดรสถัดไปที่ใกล้เคียงกับคีย์แฮชที่หาได้จากฟังก์ชันแฮชมาเป็นตำแหน่งในการเพิ่มข้อมูล

2.เปลี่ยนโครงสร้างของตารางแฮชในตำแหน่ง table[i] ให้สามารถเก็บข้อมูลได้มากกว่าหนึ่งข้อมูล

แก้ปัญหาการชนกันด้วยการหาแอดเดรสถัดไปที่ใกล้เคียงที่สุด

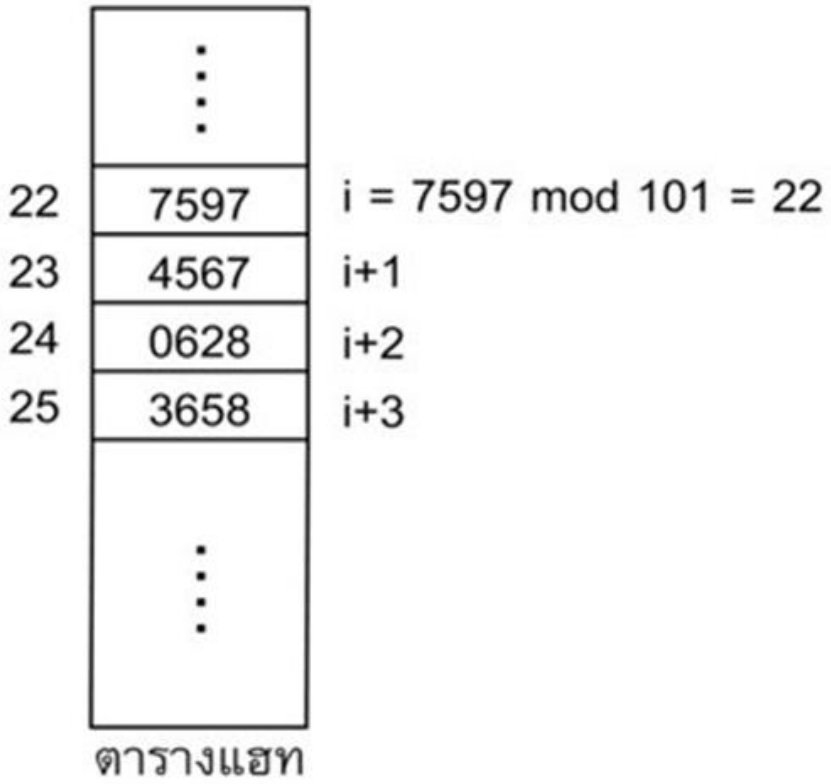
การแก้ปัญหาแบบลำดับ (Linear probing)

เป็นรูปแบบที่ง่ายที่สุดในการแก้ปัญหาการชนกันของคีย์แฮช

ใช้รูปแบบการเลื่อนตำแหน่งแอดเดรสไปยังตำแหน่งถัดไปในตารางแฮชมาเก็บข้อมูลแทนตำแหน่งแอดเดรสที่ชนกัน ซึ่งก็คือตำแหน่ง table[h(searchKey)+1], table[h(searchKey)+2] ตามลำดับ

จะเลื่อนตำแหน่งไปเรื่อยๆ จนกระทั่งเจอตำแหน่งที่ยังไม่เก็บข้อมูล

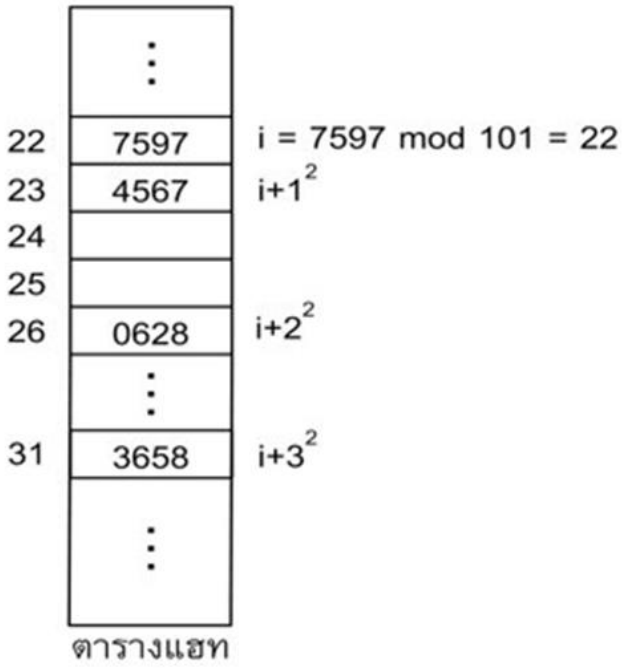
การแก้ปัญหาแบบลำดับ (LINEAR PROBING)



การแก้ปัญหาแบบกำลังสอง (QUADRATIC PROBING)

เป็นการแก้ไขปัญหาการชนกันของแฮชคีย์ด้วยการปรับโครงสร้างการแก้ปัญหาแบบลำดับ

แก้ปัญหาด้วยการนำคีย์ยกกำลังสอง เพื่อตรวจสอบตำแหน่ง table[h(serchKey)+12], table[h(serchKey)+22], table[h(serchKey)+32] จนกระทั่งเจอตำแหน่งที่ยังไม่เก็บข้อมูล



การทำแฮช 2 ครั้ง (DOUBLE HASHING)

การทำแฮช 2 ครั้งเป็นการแก้ปัญหาโดยใช้คีย์แบบไม่อิสระ

เป็นโครงสร้างที่ใช้หลักการแก้ไขปัญหาแบบลำดับ โดยเริ่มจากหาแอดเดรสที่ได้แฮชฟังก์ชันในลำดับที่หนึ่ง h1(key) ก่อนและเมื่อตำแหน่งที่จากแฮชฟังก์ชันครั้งที่ 1 มีข้อมูลอยู่แล้ว จะหาแฮชฟังก์ชันในลำดับที่สอง h2(key) เพื่อหาตำแหน่งแอดเดรสในตำแหน่งถัดไป โดยมีข้อกำหนดในการทำแฮช 2 ครั้งของแฮชฟังก์ชัน h1(key) และ h2(key) ดังนี้

$h2(key) \neq 0$

$h1 \neq h2$

การทำแฮช 2 ครั้ง (DOUBLE HASHING) (ต่อ)

ตัวอย่างเช่น กำหนดให้ h1 และ h2 เป็นลำดับที่หนึ่งและลำดับที่สองของแฮชฟังก์ชันดังนี้

h1(key) = key mod 11

h2(key) = 7 – (key mod 7)

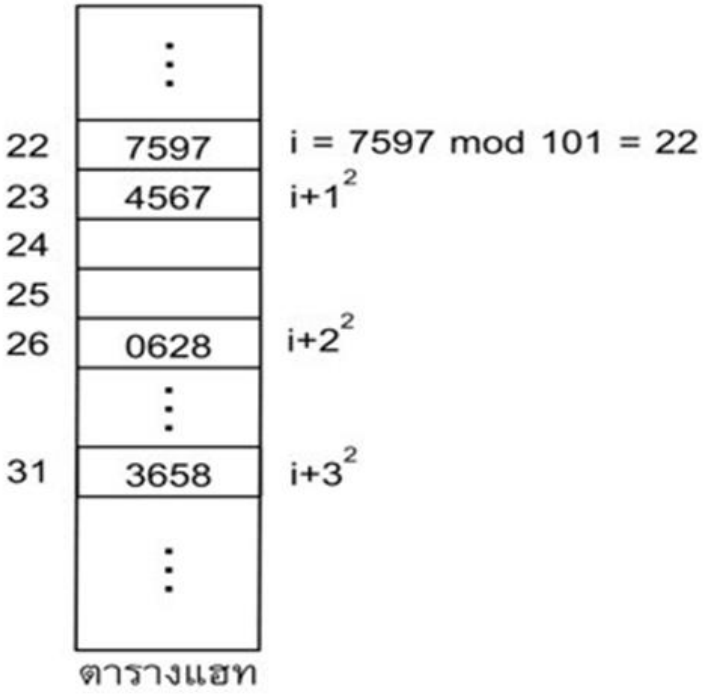
เมื่อตารางแฮชสามารถเก็บข้อมูลได้ 11 ข้อมูลและถ้า key = 58 ทำให้แฮชคีย์ h1 อยู่ในตำแหน่ง 3 (58 mod 11) และ h2 เป็นการแก้ไขปัญหาการชนกัน ด้วยการหาตำแหน่งแอดเดรสในลำดับถัดไปคือ 5 (7 – 58 mod 7)

การทำแฮช 2 ครั้ง (DOUBLE HASHING) (ต่อ)

หาแฮชคีย์ของคีย์ในการค้นหา 14 คือ h1(14) = 3 ซึ่งชนกับคีย์ 58 ดังนั้น ต้องหาแฮชคีย์ในลำดับที่ 2 คือ h2(14) = 7 จึงเพิ่มคีย์ในการค้นหาของ 14 ในตำแหน่ง 10 คือ table[3+7] หรือ table[10]

สุดท้ายคีย์ในการค้นหา 91 คือ h1(91) = 3 และ h2(91) = 7 แต่ table[3] และ table[10] มีข้อมูลที่ถูกครอบครองแล้ว สุดท้ายในการหาคีย์ค้นหา 91 กลับไปใช้ฟังก์แฮช h1 ใหม่อีกครั้งแต่เปลี่ยนคีย์ที่ได้จาก key = h1(91) + h2(91) + h2(91) ซึ่งจะได้ 3 + 7 + 7 = 17 ดังนั้น ตำแหน่งในการเพิ่มคีย์ในการค้นหา 91 คือ h1(key) = 17 mod 11 = 6 อยู่ใน table[6]

การแก้ปัญหาแบบกำลังสอง (QUADRATIC PROBING)



การแก้ปัญหาแบบกำลังสอง (QUADRATIC PROBING)

เป็นการแก้ไขปัญหาการชนกันของแฮชคีย์ด้วยการปรับโครงสร้างการแก้ปัญหาแบบลำดับ

แก้ปัญหาด้วยการนำคีย์ยกกำลังสอง เพื่อตรวจสอบตำแหน่ง

table[h(serchKey)+12],
table[h(serchKey)+22],
table[h(serchKey)+32] จนกระทั่งเจอตำแหน่งที่ยังไม่เก็บข้อมูล

การแก้ปัญหาการชนกันด้วยวิธีการปรับโครงสร้างตารางแฮช

การเก็บข้อมูลแบบกลุ่ม (Buckets)

ในแต่ละตารางแฮช table ของแต่ละตำแหน่ง table[i] จะมีอาร์เรย์ที่เรียกว่า Bucket (กลุ่มข้อมูล) ทำหน้าที่เก็บข้อมูลแฮชเข้าไปในแต่ละ table[i] ในอาร์เรย์

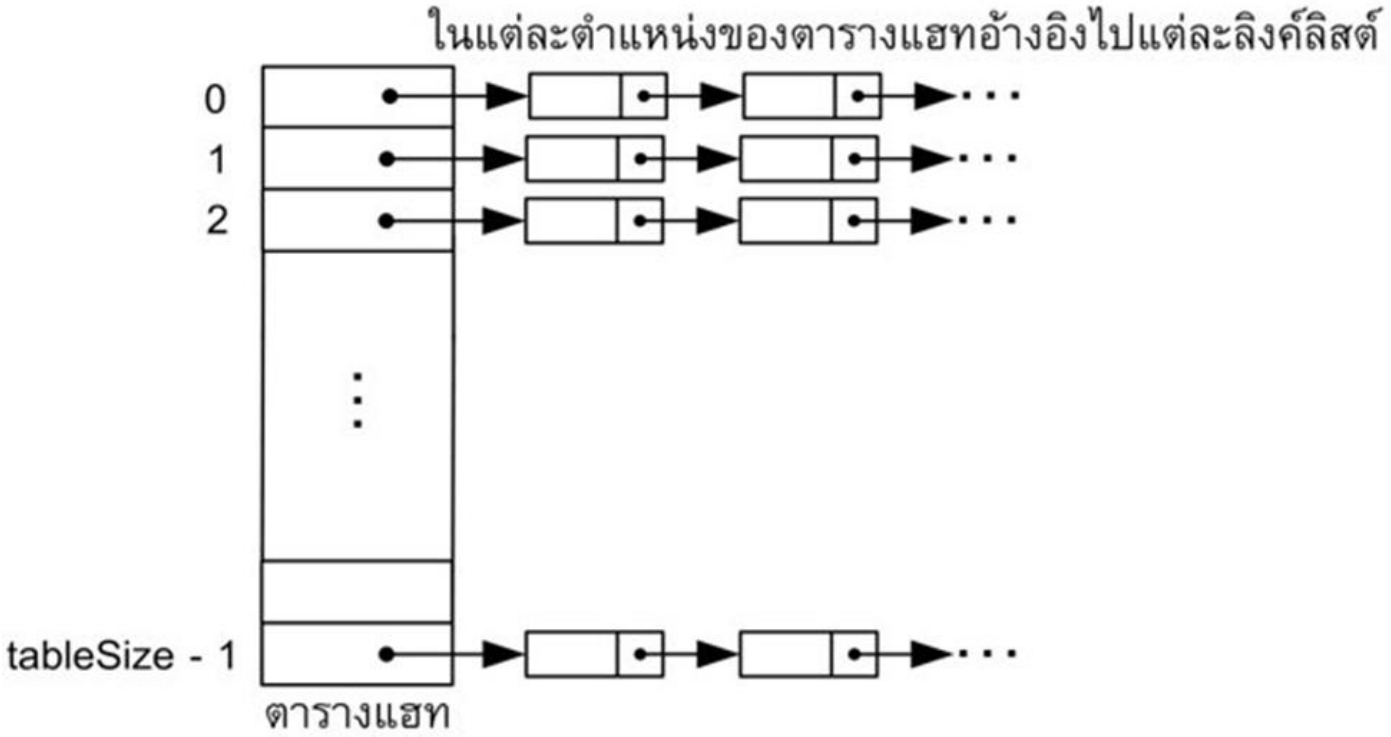
แต่ก็อาจจะเกิดปัญหาขึ้นถ้าขนาดของ Bucket ที่มีขนาดเท่ากับ B มีขนาดที่น้อยกว่าขนาดของข้อมูลที่จะเพิ่มเข้าไปในตารางแฮช ซึ่งจะเกิดการชนของข้อมูลใน Bucket ได้

ดังนั้น จะปรับให้ขนาดของ Bucket มีขนาดเป็น B + 1 ในตำแหน่งอาร์เรย์เดียวกัน ซึ่งเป็นความคิดที่ดีในการเก็บข้อมูลในตารางแฮช

การแยกออกจากกันด้วยการเชื่อมโยง (SEPARATE CHAINING)

การแยกออกจากกันด้วยการเชื่อมโยง (Separate Chaining) เป็นอีกทางเลือกหนึ่งในการออกแบบตารางแฮชด้วยการใช้อาร์เรย์ของลิงค์ลิสต์ในการเก็บข้อมูล

วิธีนี้เป็นรูปแบบการแก้ปัญหาการชนกันด้วยการกำหนดให้ในแต่ละ table[i] ทำหน้าที่อ้างอิงไปยัง ลิงค์ลิสต์เพื่อใช้ในการเชื่อมโยงข้อมูล


ในแต่ละตำแหน่งของตารางแฮทอ้างอิงไปแต่ละลิงค์ลิสต์
0
1
2
tableSize - 1
ตารางแฮท

ข้อสรุป HASHING

แฮชเป็นการจัดการข้อความหรือข้อมูลโดยการเปลี่ยนให้เป็นดัชนี (index) เพื่อใช้อ้างอิงข้อมูลในอาร์เรย์หรือฐานข้อมูล ซึ่งใช้แฮชฟังก์ชันในการเปลี่ยนข้อความหรือข้อมูลเป็นคีย์ในการอ้างอิงใน แฮทเทเบิล

แฮชฟังก์ชันมีหลายวิธีในเปลี่ยนตัวเลขให้เป็นคีย์ และเมื่อได้คีย์จากแฮชฟังก์ชันแล้วอาจจะมีคีย์ที่ชนกันได้

ข้อสรุป HASHING

การชนกัน คือ ตำแหน่งของคีย์ที่ได้จากแฮชฟังก์ชันมีข้อมูลอยู่แล้ว

การแก้ปัญหาการชนกันของคีย์มี 2 วิธี คือ

1.การหาตำแหน่งใหม่ที่ว่างแล้วนำข้อมูลนั้นไปเก็บไว้ในคีย์

2.การปรับโครงสร้างแฮทเทเบิลโดยนำอาร์เรย์มาเก็บคีย์และนำลิงค์ลิสต์มาเชื่อมโยงกับอาร์เรย์เพื่อทำหน้าที่เก็บข้อมูล

บรรณานุกรม

1. วิศวกรรมคอมพิวเตอร์ จุฬาฯ. “ตารางแฮช.”[ระบบออนไลน์]. แหล่งที่มา www.cp.eng.chula.ac.th/~vishnu/datastructure/aj.vishnu%20-%20hashing (17 ธันวาคม 2560).
2. มหาวิทยาลัยราชภัฏเพชรบูรณ์ สาขาเทคโนโลยีคอมพิวเตอร์อุตสาหกรรม คณะเทคโนโลยีการเกษตร.“การค้นหาข้อมูลแบบแฮชชิง (Hashing Search) .”[ระบบออนไลน์] .แหล่งที่มาhttp://agritech.pcru.ac.th/new/page/e-learningdata/9.6Hashing.php (17 ธันวาคม 2560)