**Hashing Search**

• การค้นหาข้อมูลด้วยวิธีนี้ สามารถเข้าถึงตําแหน่งของข้อมูลได้โดยตรง

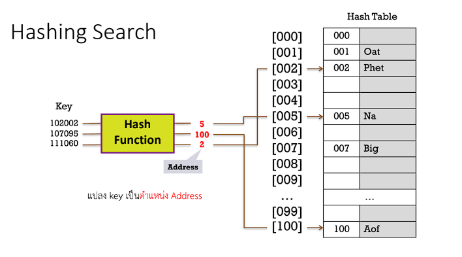
• สามารถค้นหาตําแหน่งข้อมูลที่ต้องการได้ภายในครั้งเดียว

• ประสิทธิภาพของการค้นหาจึงเป็นค่าคงที่ O(1) ต่างจาก..

• Sequential Search ประสิทธิภาพของการค้นหาคือ O(n)

• Binary Search ประสิทธิภาพของการค้นหาคือ O(log n)

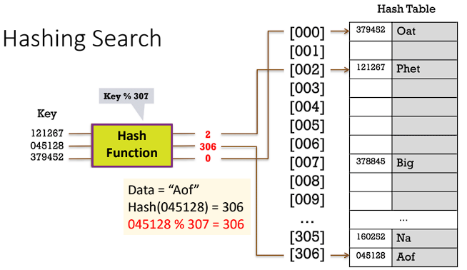
• หลักในการค้นหาข้อมูล มีดังนี้ Key คือ ข้อมูลที่ใช้ในการค้นหา

• Hash Function คือแปลง key เป็นตําแหน่ง Address เพื่อใช้เข้าถึงตําแหน่ง

ของข้อมูลใน Hash Table

• Hash Table คือ ตารางที่ใช้สําหรับเก็บข้อมูล

•วิธีแปลงค่าคีย์มาเป็น Address ใช้ขนาดของ List

****มาเป็น ขนาดของ Hash Table ให้ Hash Table มีขนาดใหญ่กว่า

จํานวน Key ที่มีอยู่

• ตัวอย่าง: ใช้วิธีการ mod

Address = key mod tableSize

Collision Resolution

• การชนกันของคีย์• เป็นข้อเสียของ Hashing

• ขึ้นอยู่กับขนาดของ Hash table• ขึ้นอยู่กับการออกแบบ Hash Function

• ความสมํ่าเสมอของการกระจายกันของคีย์•แนวทางการแก้ไขเมื่อมีการชนกันของคีย์

• Open Addressing•Linear probe• Quadratic probe•Linked Lists• Buckets

**Open Addressing: Linear Probe**

•f(i) = i เมื่อ f เป็น Linear function ของ i

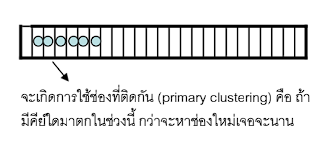
• เป็นวิธีที่มีรูปแบบไม่ซับซ้อน ดําเนินการเมื่อไม่สามารถจัดเก็บข้อมูลหลังจากการ Hash ได้

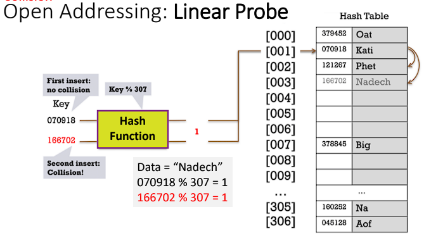
• เป็นการแก้ไขการชนกันด้วยการนํา Address ปัจจุบันมาบวกเพิ่มอีก 1 ตําแหน่งไปเรื่อยๆ

จนกว่าจะพบตําแหน่งที่ว่าง (ในบางกรณีอาจจะ เพิ่ม 2, ลบออก 2, เพิ่ม 3, ลบออก 3)

ข้อดี 1.ง่ายต่อการพัฒนาใช้งาน 2.ข้อมูลที่แทรกเข้าไปใหม่จะอยู่ใกล้กับ Address จริง

ข้อเสีย 1.การกระจายคีย์ใน Hash Table ไม่สมํ่าเสมอ

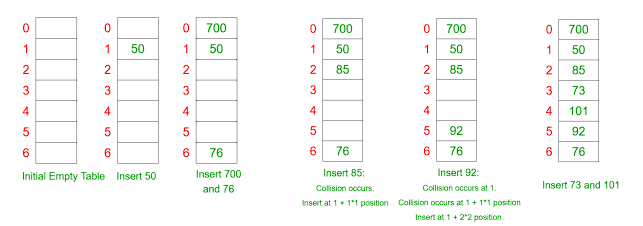
2.เวลาในการหาช่องว่างจะนาน เพราะในการหาแต่ละครั้ง ต้องเริ่มจาก f(1), f(2), f(3), ...

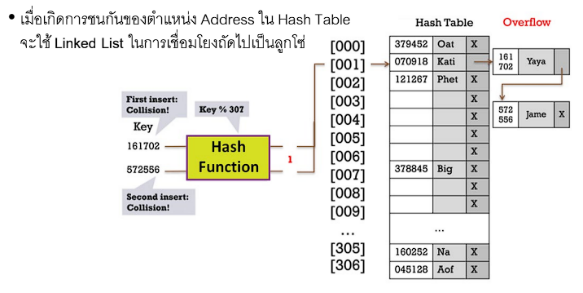
3.เกิดการเกาะกลุ่มกันอย่างมาก (primary clustering)

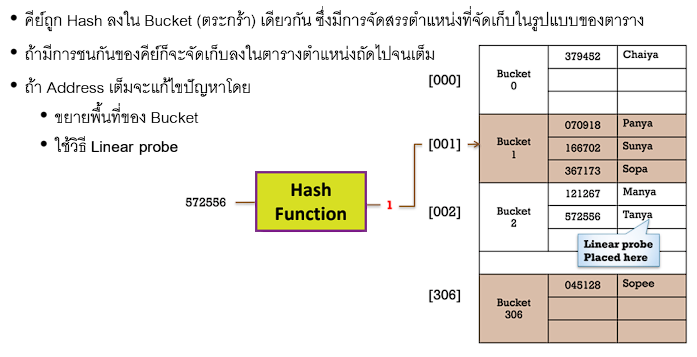
**Open Addressing: Quadratic Probe**

• วิธีนี้ช่วยกําจัด Primary Clustering

•hi(x) = (hash(x) + f(i)) % tableSize เมื่อ f(i) = i2, i แทนจํานวนครั้งที่หา

• For example: Let us consider a simple hash function as “key mod 7” and sequence of keys as 50, 700, 76, 85, 92, 73, 101

**Linked Lists / Chaining**

**Bucket Hashing**

Dictionary เป็นหนึ่งในคอลเล็กชันของไพธอนที่มีประโยชน์ที่สุดbDictionary เป็นชนิดข้อมูลที่เก็บคู่ของ Keyและ Value ที่สัมพันธ์กัน Key ถูกใช้ในการค้นหาค่าข้อมูลที่สัมพันธ์กันเรามักอ้างถึงแนวคิดนี้ว่าแผนที่ นายแผนเป็นไม่ซ้ำกันทั้งหมดดังนั้นจึงมีความสัมพันธ์หนึ่งต่อหนึ่งระหว่างคีย์และค่าข้อมูลที่ข้อมูลมีการนิยามดังต่อไปนี้โครงสร้างเป็นคอลเล็กชันที่ไม่เรียงลำดับของคู่สัมพันธ์ระหว่างคีย์และค่าข้อมูล

คำสั่งในการใช้ dictionary

Create-dict()

เข้าถึง-get()

เข้าถึงkey-dict.keys()

เพิ่มข้อมูล-ชื่อ dict [keys]=value

เช็คว่าคีย์อยู่ในdict- if keys in dict

เปลี่ยน value – dict [keys]=value(ที่จะเปลี่ยน)

อัปเดต