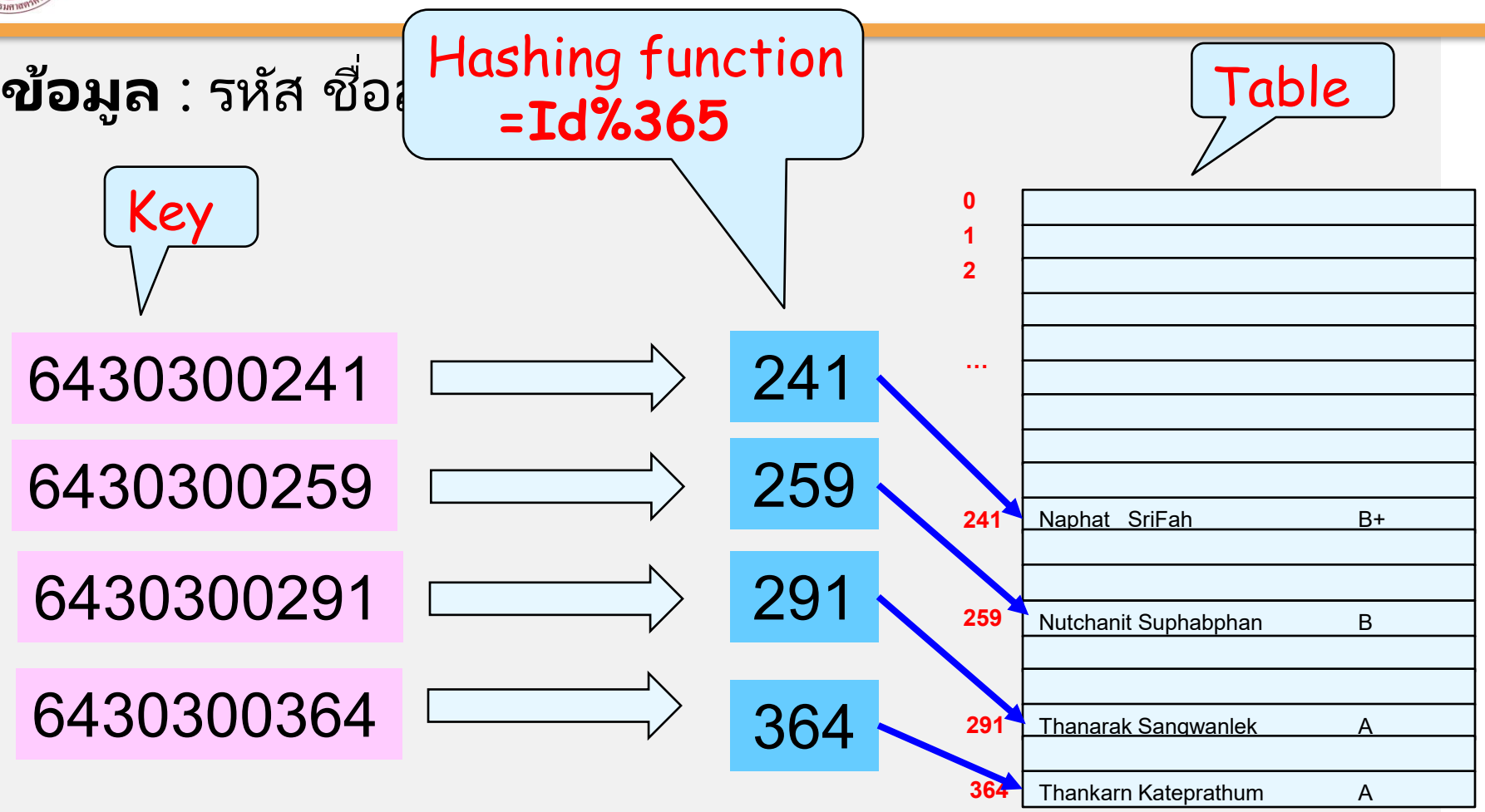
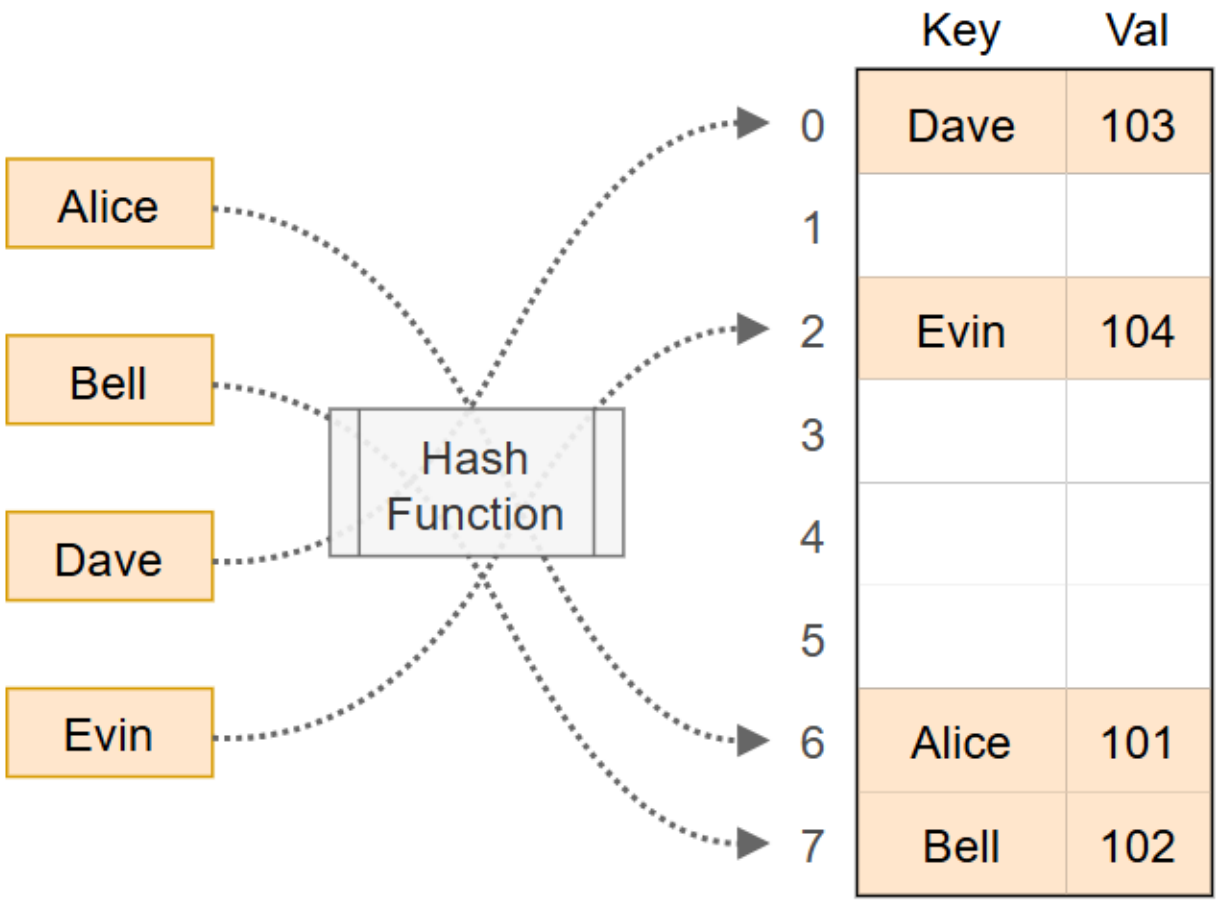
MODULE7—HASHING

1.เข้าใจวิธีการเก็บข้อมูลด้วยตารางและการเข้าถึงข้อมูล
2.วิธีการสร้างฟังก์ชันแฮช
3.การแก้ไขในกรณีเกิดการชนกันของข้อมูล ด้วยวิธี
-Separate Chaining
-Open Addressing

ข้อมูล : รหัส ชื่อ


Hashing function
=Id%365
Table
Key
6430300241
6430300259
6430300291
6430300364
241
259
291
364
0
1
2
...
241 Naphat SriFah B+
259 Nutchanit Suphabphan B
291 Thanarak Sangwanlek A
364 Thankarn Kateprathum A



Hash Table Representation

https://simplerize.com/data-structures/hash-table-introduction

7.0 Hashing

1. เป็นเทคนิคการจัดเก็บและเข้าถึงข้อมูลที่รวดเร็ว เหมาะสำหรับโครงสร้างข้อมูลแบบ Dictionary หรือ Associative Array (table["Dave"])
2. เป็นโครงสร้างข้อมูล ในรูปแบบตารางที่มี Operation ประกอบด้วย insert, delete, find
3. หลักการคือ ใช้ Hash Function แปลง Key ให้เป็น Index ที่ใช้ในการจัดเก็บหรือค้นหาข้อมูลใน ตารางแฮช
4. ให้เวลาคงที่ O(1) จะทำได้รวดเร็วมาก
5. ไม่เหมาะกับงานที่มีการเรียงข้อมูล หาค่าสูงสุด ต่ำสุด

7.1 General Idea

ส่วนประกอบที่สำคัญ

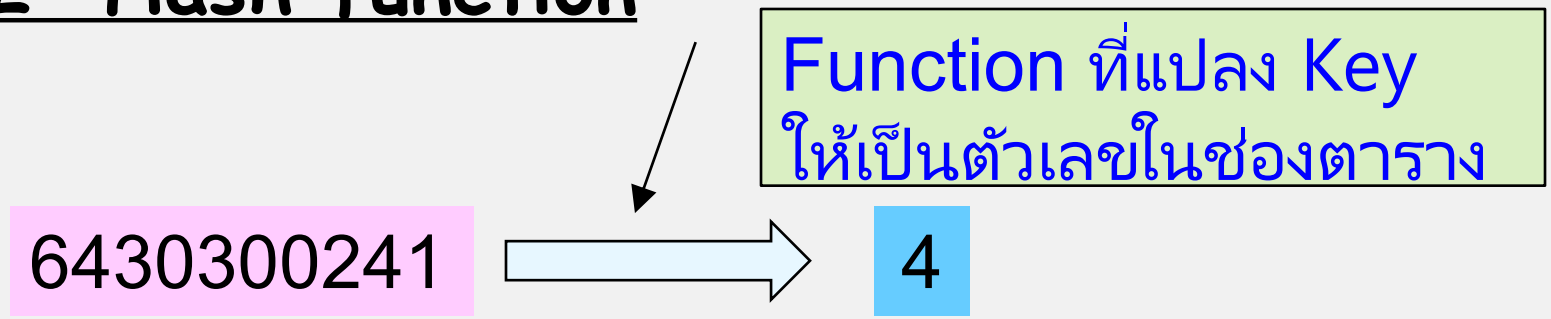
1) Key มี key เช่น Id และข้อมูลบุคคล

2) Hashing function key มาคำนวณด้วย function เพื่อให้ได้ที่อยู่ของข้อมูลว่าอยู่ในตารางช่องใด

3) Table size เก็บข้อมูลในตาราง(Array) ขนาด 0-(N-1) ช่อง (กรณีสร้างตารางอย่างง่าย)

4) ปัญหา Collision key 2 ตัวที่ไม่เหมือนกัน ผ่าน function แล้วได้ค่าเดียวกัน

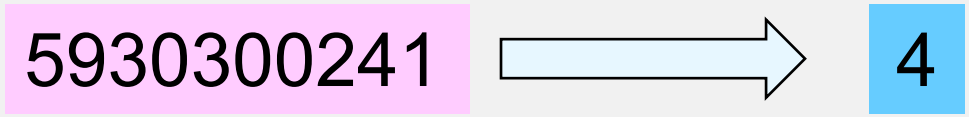
7.2 Hash function



- วิธีการเอา key มาผ่านกระบวนการหนึ่ง (เช่นนำมาคำนวณด้วย ฟังก์ชันบางตัว) จนได้ตำแหน่งที่เป็นหมายเลขช่องของตาราง จากนั้นจะนำข้อมูลไปเก็บไว้ในตำแหน่งนั้น
- คำนวณง่ายไม่เสียเวลา
- คีย์สองตัวต้องเข้าฟังก์ชั่นแล้วได้ค่าดัชนี(index) ของอาร์เรย์ที่แตกต่างกัน
- แฮชฟังก์ชั่นกระจายคีย์ไปได้ทั่วตาราง

7.2 Hash function

7.2.1) Key เป็นตัวเลข : ใช้ standard function



Standard function : Key % TableSize

Example key : integer

Standard function = Key Mod tableSize

ลงตารางช่องเดียวกัน

1. 6430300141 Chanowat Thanasakkul 6430300141%22 = 17
2. 6430300176 Thanaphat Wilawan 8
3. 6430300192 Thitiya HonglertSakul 2
4. 6430300311 Tepitak Butrsaithong 11
5. 6430300443 Teerapol Pharuthum 11
6. 6430300559 Bunnyapol Kajonpaisarn 17
7. 6430300575 Praphan Chawanaranon 11

.....

**** จำนวนนิสิตลงทะเบียน 22 คน M(คือ tableSize)=22

ปัญหา:

- Table size = 10 หรือลงท้ายด้วย 0
- Key 80, 120 , 70, 90

การแก้ไข:

- พิจารณา Key ว่ามีลักษณะเป็นอย่างไร
- Table Size is Prime
- 22 -> 23 หรือ 31

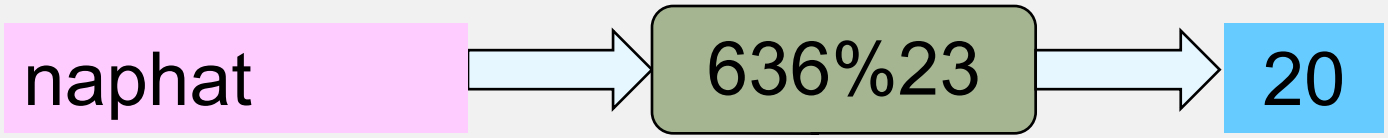
Standard function = Key Mod tableSizePrime

1.	6430300141	Chanowat Thanasakkul	17 11
2.	6430300176	Thanaphat Wilawan	8 0
3.	6430300192	Thitiya HonglertSakul	2 16
4.	6430300311	Tepitak Butrsaithong	11 20
5.	6430300443	Teerapol Pharuthum	11 14
6.	6430300559	Bunnyapol Kajonpaisarn	17 15
7.	6430300575	Praphan Chawanaranon	11 8

.....

**** จำนวนนิสิตลงทะเบียน 22 คน M=23

7.2.2) Key : string



n	110	
a	97	
p	113	= 636
h	104	
a	97	
t	116	

7.2.2 key : String

Ascii Value

a = 97

b = 98

c = 99

d = 100

e = 101

...

z = 122

Code	Char	Code	Char
96	`	112	p
97	a	113	q
98	b	114	r
99	c	115	s
100	d	116	t
101	e	117	u
102	f	118	v
103	g	119	w
104	h	120	x
105	i	121	y
106	j	122	z
107	k	123	{
108	l	124	|
109	m	125	}
110	n	126	~
111	o	127	[backspace]

7.2.2 key : String

1) Function : (นำAscii Value มาบวกกัน) mod TabSize

ตัวอย่าง

Key : 3 characters long
Tabsize : 10,007
Function : (นำ Ascii Value มาบวกกัน) Mod TabSize
ช่วงของคีย์ที่เป็นไปได้

=aaa =97+97+97 = 291
=zzz =122+122+122 = 366
= 75 ค่า

```
#include <stdio.h>
#include <iostream>
#include <string.h>
using namespace std;
int hash(string key,int tabsize)
{   int hashval=0;
    for(int i=0;i<key.length();i++)
        hashval+=key[i];
     return hashval%tabsize;
}
```

```
int main()
{  char key[4]="abc";
    int htab=5;
   cout << hash(key,htab);
}
```

ปัญหา

1. ant,tan,nat = 97+110+116
   หรือ abc,acb,bac,...
2. ถ้าตารางขนาดใหญ่ แต่ key ขนาดไม่ยาว key จะไม่กระจายไปด้านหลังตาราง

คำถาม ถ้า key เป็นตัวอักษรภาษาอังกฤษ จำนวน 6 character ผลลัพธ์จะตกอยู่ในตาราง ในช่วงใดถึงช่วงใด

จำนวนตัวอักษร
ภาษาอังกฤษ + space

2) Fu ...ะจายค่าของคีย์ให้กว้างขึ้น

นำค่าตัวอักษร ...r+space = 27 มาทำให้ช่วงกว้างขึ้น

ตัวอย่าง

Function = (k1 + 27*k2 + 27^2*k3) mod TabSize

aaa = 97 + 97 + 97 = 291 mod TabSize

zzz = 122+122+122 = 366 mod TabSize

aaa = 97 + 27*97 + 27^2*97 = 73,429 mod TabSize

zzz = 122+27*122+27^2*122=92,354 mod TabSize

จะได้ ฟังก์ชั่นที่มีการกระจาย key ที่ดี

Function = (k1 + 27*k2 + 27^2*k3) mod TabSize

abc = 97 + 27*98 + 27^2*99
= 97+2646+72171
= 74,914 mod 10007 = 4865

cba = 99 + 27*98 + 27^2*97
= 99+2646+70713
=73458mod 10007 = 3409

```
int hash(string key, int tabsize)
{   return (key[0] + 27*key[1] + 27*27*key[2])%tabsize;
}
```

3. Function พหุนาม(Polynomial Function) ของ 37
โดยใช้ Horner's Rules

$$\sum_{i=0}^{KeySize-1} Key[KeySize - i - 1] * 37^i$$

$h_k = k_0+37k_1+37^2k_2+...$

```
int hash(string key, int tabsize)
{       int hashvalue=0;
        for(int i=0;i<key.length();i++)
                hashvalue=hashvalue+pow(37,i)*key[i];
        hashvalue%=tabsize;
        if(hashvalue<0)
                hashvalue+=tabsize;
        return hashvalue;
}
```

ปัญหาเรื่อง key ยาวอาจจะเลือกค่าเฉพาะเลขคู่หรือเลขคี่

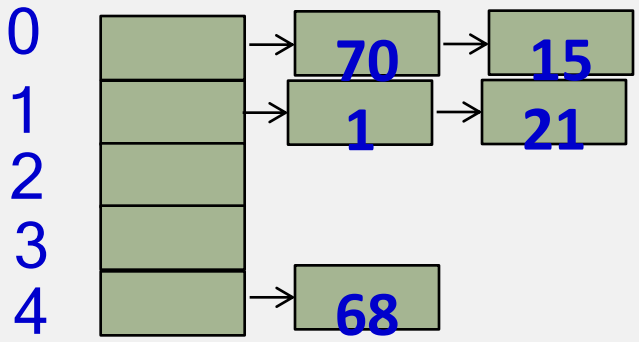
7.3 การแก้ปัญหาการชนกันของข้อมูล

- collision resolution ถ้าค่าแฮชที่ได้จากฟังก์ชันแฮชที่เกิดจากค่า key ตัวใหม่ที่จะนำข้อมูลไปบรรจุลงในตารางแฮชไปตรงกับตำแหน่งที่มีค่าอื่นยึดครองอยู่แล้ว เรียกว่าเกิดการชน (collision)
- การแก้ไขเพื่อให้ค่าใหม่นั้นมีตำแหน่งที่จะเก็บข้อมูลบรรจุลงได้มีวิธีการหลายอย่างในการแก้ปัญหาการชน แต่ในที่นี้จะ กล่าวถึงวิธีการพื้นฐาน 2 วิธี คือ:
  1) separate chaining (open hashing)
  2) open addressing (closed hashing)

7.3 การแก้ปัญหาการชนกันของข้อมูล

7.3.1 Separate Chaining

วิธีการนี้จะใช้ Link list เข้ามาช่วยในการแก้ปัญหาในกรณีที่ข้อมูลมีการชนกัน



70, 1, 21, 15, 68

h(k) = k mod 5

กำหนด n จำนวนข้อมูลในตาราง

m ขนาดของตาราง

λ เป็นค่าความยาวโดยเฉลี่ยของ List

การค้นแล้วไม่พบ $O(1+\lambda)$

ค้นแล้วพบ $O(1+\lambda/2)$

7.3.2 Open Addressing

1) Linear Probing กรณีนี้ F(i) linear

h(x) = key mod m

h'(x)=(h(x)+F(i)) mod m

Key : 89,18,49,58,69

F(i) = i, m = 10

89 =

18 =

49 =

58 =

69 = ... -> ?

index	key
0	49
1	58
2	
3	
4	
5	
6	
7	
8	18
9	89

2) Quadratic Probing

h(x) = key mod m

h'(x)=(h(x)+F(i^2)) mod m

Key : 89,18,49,58,69

f(i) = i^2, m = 10

89 =

18 =

49 =

58 =

69 =

index	key
0	49
1	
2	58
3	59
4	
5	
6	
7	
8	18
9	89

3) Double hashing

h1(x) = key mod m

h2(x) = 7 – (x mod R)

h'(x)=(h1(x)+i*h2(x))mod m

Key : 89,18,49,58,69

m = 10

89 = 9

18 = 8

49 = 9 , (9 + 1*(7- 49%7)) mod 10 -> 6

58 = 8 , (8 + 1*(7-58%7) mod 10 -> 3

69 = 9 , ?

6

index	key
0	
1	
2	
3	58
4	
5	
6	49
7	
8	18
9	89

index	key
0	49
1	58
2	69
3	
4	
5	
6	
7	
8	18
9	89

index	key
0	49
1	
2	58
3	59
4	
5	
6	
7	
8	18
9	89

index	key
0	69
1	
2	
3	58
4	
5	
6	49
7	
8	18
9	89