

TABEL HASH (*HASH TABLE*)

Matakuliah Algoritma dan Struktur Data

Annisa Puspa Kirana, S.Kom, M.Kom

OVERVIEW

Pengenalan

- Hashing
- Tabel Hash

Memilih fungsi tabel hash

- Sisa pembagian
- Pemotongan
- Pelipatan

Menangani tabrakan dalam tabel hash

- Pengalamatan terbuka
- Doublr Hashing
- Pembentukan rantai
- Pengalamatan buket

PENGENALAN HASHING

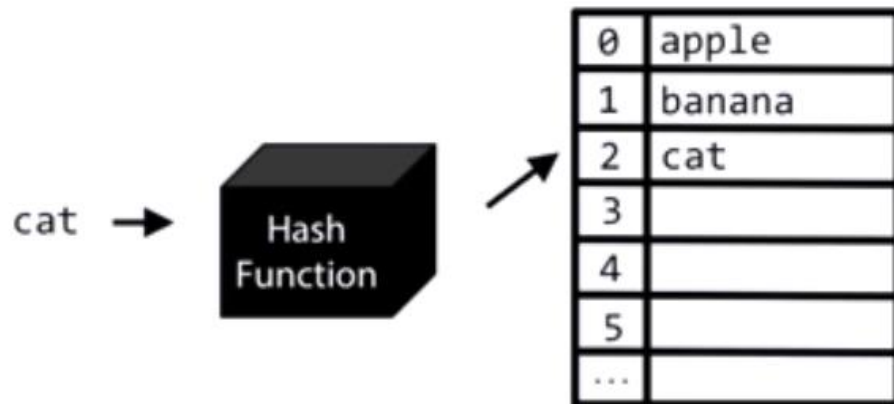
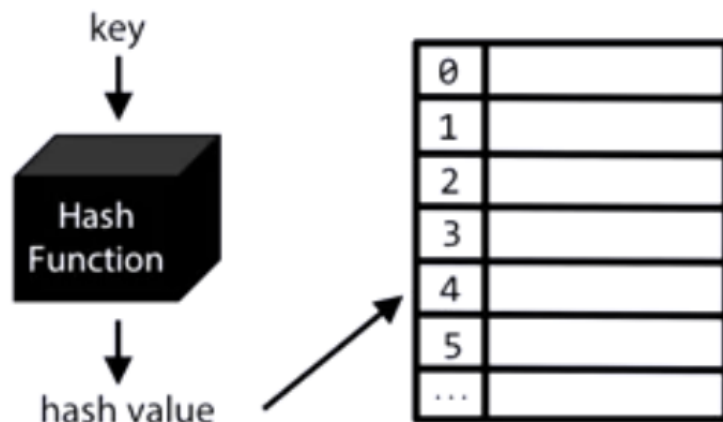
HASHING Metode untuk menyimpan data dalam sebuah array agar penyimpanan data, pencarian data, penambahan data, dan penghapusan data dapat dilakukan dengan cepat.

Metode untuk melakukan penambahan, penghapusan dan pencarian dengan constant average time.

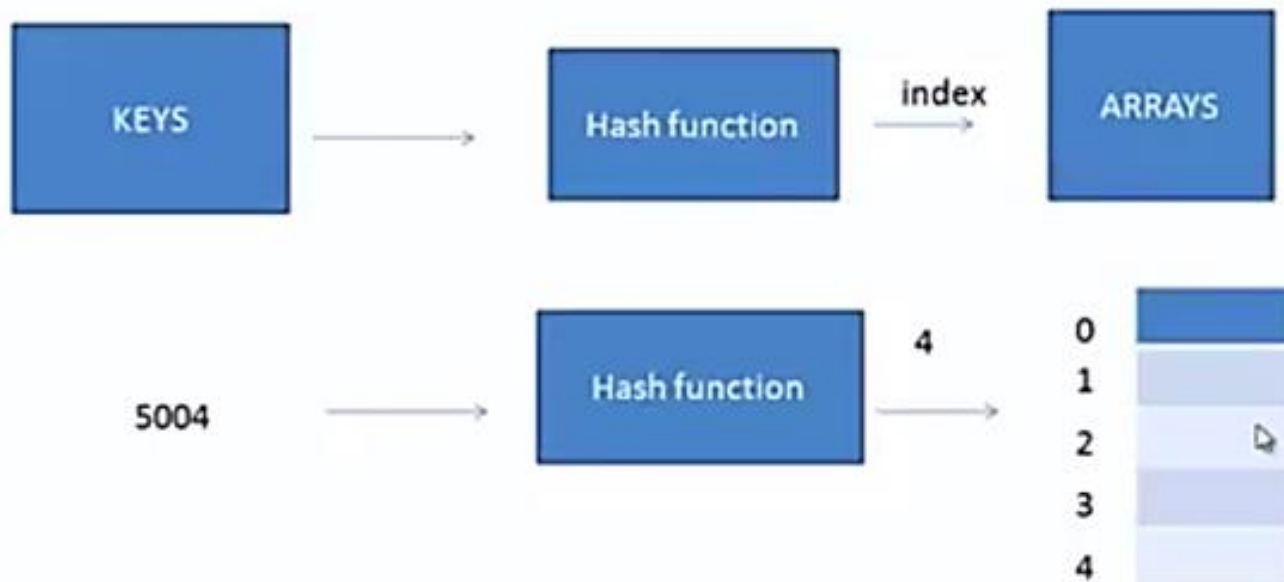
PENGENALAN TABEL HASH

Hash Table → sebuah struktur data yang terdiri atas sebuah tabel dan fungsi yang bertujuan untuk memetakan nilai kunci yang unik untuk setiap record (baris) menjadi angka (hash) lokasi record tersebut dalam sebuah tabel.

Menyimpan data pada memori ke dalam baris-baris dan kolom-kolom sehingga membentuk table yang diakses dengan cepat



PENGENALAN TABEL HASH CONT...



Hash function computes the Keys(Input value) , to generate Index value.

TUJUAN

6

- Mendapatkan posisi (lokasi, alamat) record secara langsung (immediate, direct) pada waktu dicari.
- Mempercepat pencarian kembali dari banyak data yang disimpan.
- Dapat memotong banyak biaya pencarian direktori. (memasukkan berkas, menghapus data juga lebih mudah dan cepat)
- Mempercepat table *look-up*, atau untuk membandingkan data (misalnya mencari data tertentu dalam sebuah basis data, mendeteksi data yang terduplikasi dalam sebuah file berukuran besar, dan sebagainya).
- Proses menyimpan dan mencari data lebih cepat

WHY HASH? ⁷

Untuk mengindex sekumpulan array untuk memudahkan proses pencarian.

NIP dalam sebuah Perusahaan menggunakan 5 digit,

Nilai Berkisar antara 00000 - 99999. Bila menggunakan *array* diperlukan *array* yang mampu menampung 100.000 elemen.

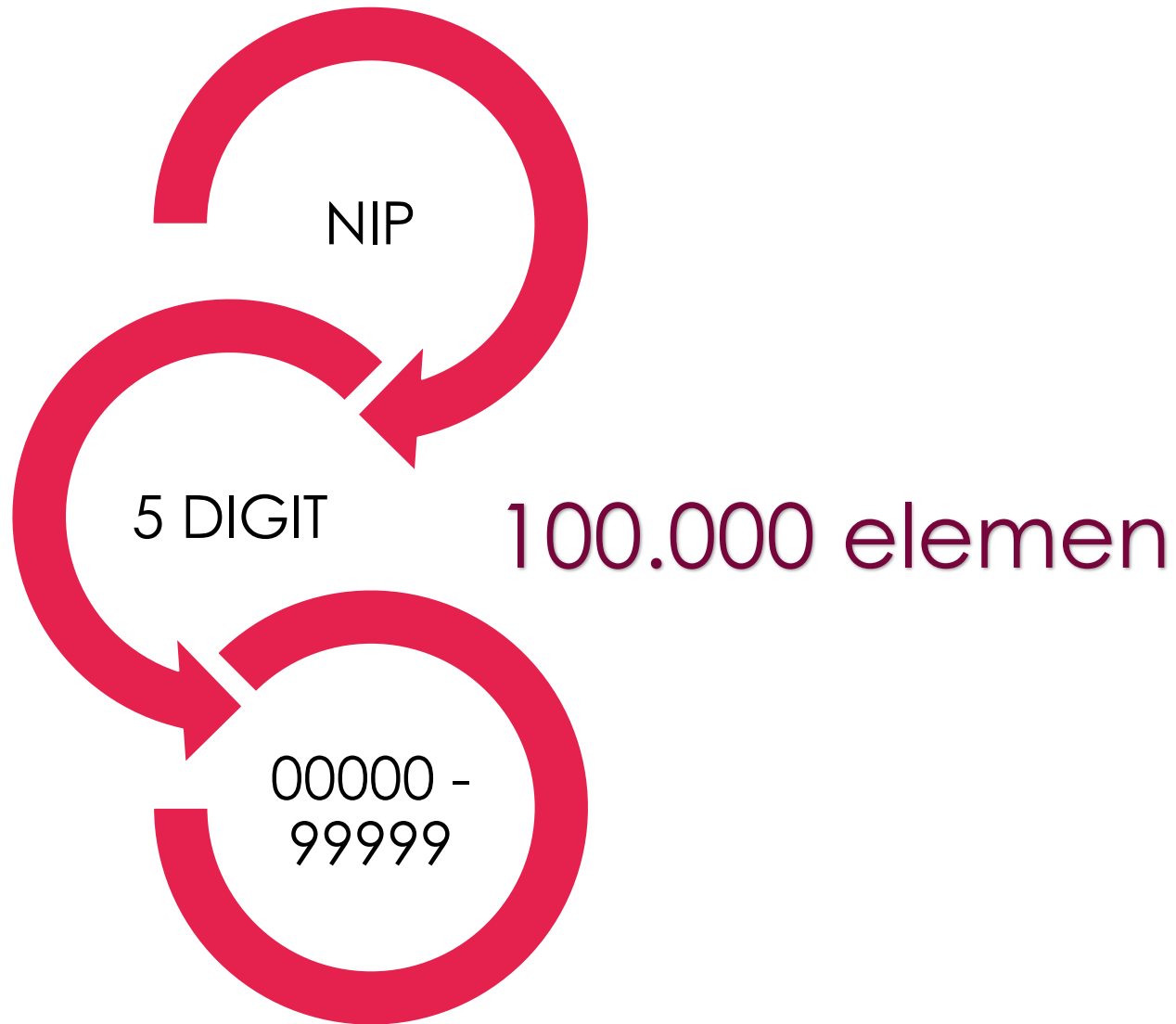
Fakta yang ada Perusahaan hanya memiliki 100 pegawai. 100.000 elemen akan membuat banyak ruang tidak terpakai / pemborosan memori.

Berapa jumlah *array* yang sebaiknya digunakan agar pencarian data menurut kunci bisa dilakukan dengan cepat? Diperlukan *array* yang berukuran kecil tetapi bisa menampung semua data.

Bagaimana cara memetakan antara NIP dengan lokasi *Array* ? Dengan fungsi Hash (*hash function*).

WHY HASH? CONT...

8



Total
Pegawai
100 orang

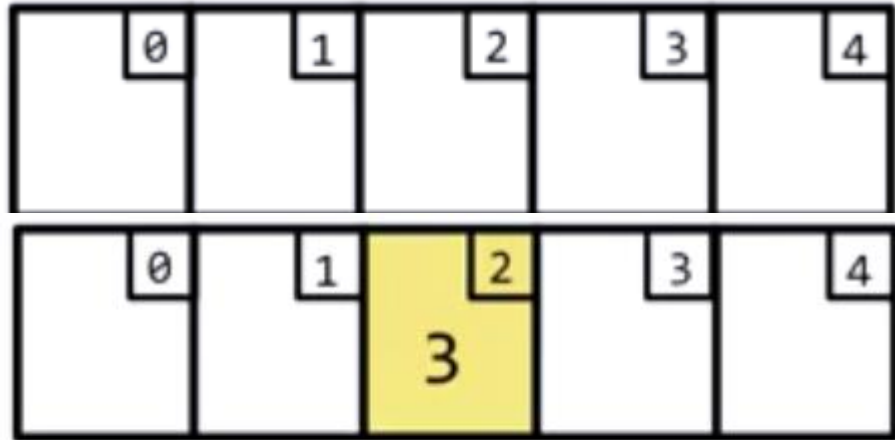
HASH

WHY HASH? CONT...

9

Array

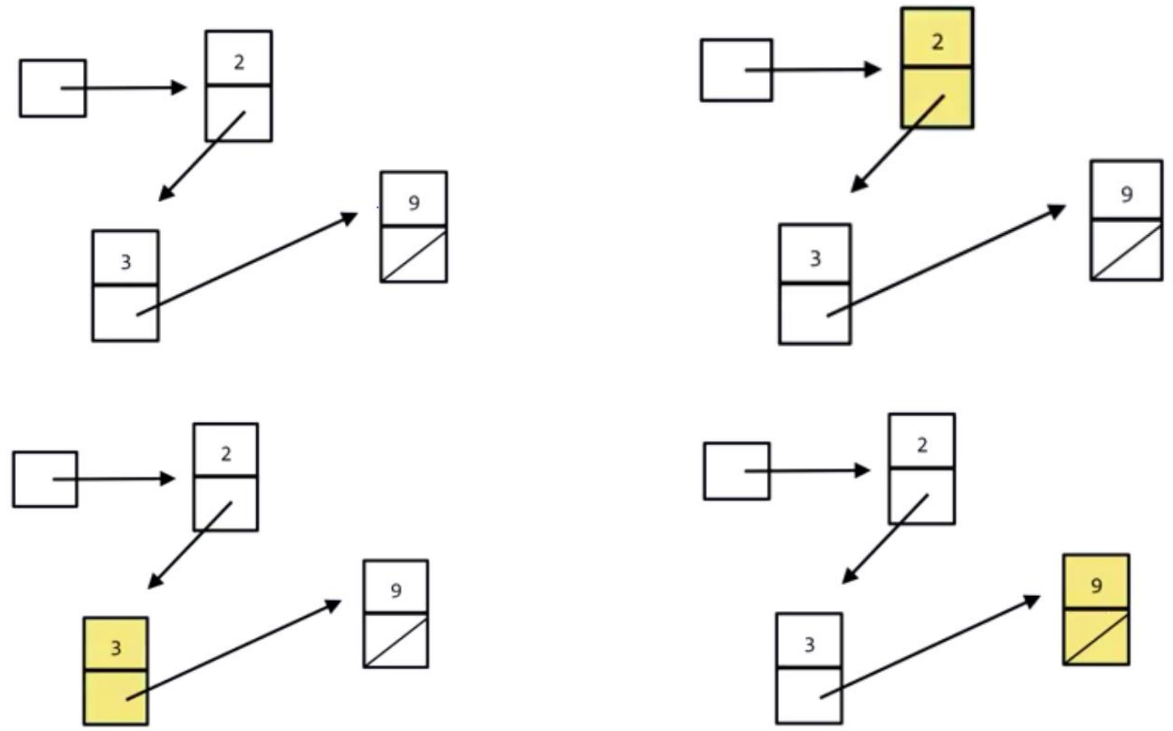
- Ukuran tetap (fix)
- Banyak memori yang tidak terpakai
- Boros ruang memori



`array[2] = 3;`

Linked list

- Ukuran dinamis
- Hemat memori
- Waktu yang dipakai untuk proses search lama
- Boros waktu



KELEBIHAN TABEL HASH¹⁰



Waktu aksesnya yang cukup cepat, jika record yang dicari langsung berada pada angka hash lokasi penyimpanannya.



Hashing relatif lebih cepat



Kecepatan dalam insertions, deletions, maupun searching relatif sama



Cocok untuk merepresentasikan data dengan frekuensi insert, delete dan search yang tinggi

KEKURANGAN TABEL HASH¹¹



Sering sekali ditemukan hash table yang record-recordnya mempunyai angka hash yang sama (bertabrakan).



Sulit (tidak efficient) untuk mencetak seluruh elemen pada hash table



Tidak efficient untuk mencari elemen minimum or maximum

IMPLEMENTASI HASH



Mencari pola rantai DNA



Pencarian dan pengarsipan data pada sistem informasi geografis



Di bidang jaringan computer digunakan untuk memproses jaringan misalnya pada jaringan ad hoc bergerak

- Fungsi hash harus memiliki sifat berikut:
 - Mudah dihitung.
 - Dua key yang berbeda akan dipetakan pada dua sel yang berbeda pada array.
 - Dapat dicapai dengan menggunakan direct-address table dimana semesta dari key relatif kecil.
 - Membagi key secara rata pada seluruh sel.
- Sebuah fungsi hash sederhana adalah menggunakan fungsi mod (sisa bagi)

MEMILIH FUNGSI HASH¹⁴

Kriteria Memilih Fungsi Hash

- Komputasi harus mudah dan cepat
- Harus menghasilkan nilai tersebar disepanjang jangkauan indeks *array*.
- Harus dapat cepat dihitung.
- Harus meminimalkan juga collisions yang terjadi.

METODE FUNGSI HASH¹⁵



FUNGSI HASH :¹⁶

SISA PEMBAGIAN (MODULAR ARITHMETIC)

Sisa dari pembagian dua buah bilangan

Melakukan konversi data ke bentuk bilangan bulat, dibagi dengan ukuran hash table, dan mengambil hasil sisa baginya sebagai indeks.

Konsep dari sisa pembagian adalah membagi nilai kunci Misalnya NIP pada data pegawai dengan suatu nilai dan sisa pembagian yang digunakan sebagai alamat hash.

Secara matematika fungsi hash ditulis menjadi :

$$H(k) = k \bmod m, m > n$$

dengan :

k = kunci

m = suatu bilangan pembagi

n = jumlah data

SISA PEMBAGIAN (MODULAR ARITHMETIC) CONT...



Mengingat $k \bmod m$ menghasilkan bilangan antara 0 sampai $m-1$ maka apabila lokasi memori (indeks array) berawal dengan 1, hasil pembagian perlu ditambah dengan angka 1.



Dengan demikian fungsi hash berupa

- $H(k) = (k \bmod m) + 1$

CONTOH MODULAR ARITHMETIC

$\text{Key}(\text{Input}) \% (\text{No of slots in an Array}) = \text{Index}$

Consider an Example

0	
1	
2	
3	7003
4	

If key is 7003

$7003 \% 5 = 3.$

Non Integer Keys?

Hash Function are applied

- To convert keys from string to Integer.
- Then Integer is computed to get Index.

a) Non Integer Key \longrightarrow Integer.

b) Integer \longrightarrow Index value

CONTOH MODULAR ARITHMETIC

CONT...

000 (nul)	016 ► (dle)	032 sp	048 0	064 @	080 P	096 `	112 p
001 ☉ (soh)	017 ◀ (dc1)	033 !	049 1	065 A	081 Q	097 a	113 q
002 ☉ (stx)	018 † (dc2)	034 "	050 2	066 B	082 R	098 b	114 r
003 ♥ (etx)	019 !! (dc3)	035 #	051 3	067 C	083 S	099 c	115 s
004 ♦ (eot)	020 ¶ (dc4)	036 \$	052 4	068 D	084 T	100 d	116 t
005 ♣ (enq)	021 § (nak)	037 %	053 5	069 E	085 U	101 e	117 u
006 ♠ (ack)	022 − (syn)	038 &	054 6	070 F	086 V	102 f	118 v
007 • (bel)	023 ‡ (etb)	039 '	055 7	071 G	087 W	103 g	119 w
008 ▣ (bs)	024 ↑ (can)	040 (056 8	072 H	088 X	104 h	120 x
009 (tab)	025 ↓ (em)	041)	057 9	073 I	089 Y	105 i	121 y
010 (lf)	026 (eof)	042 *	058 :	074 J	090 Z	106 j	122 z
011 ♂ (vt)	027 ← (esc)	043 +	059 ;	075 K	091 [107 k	123 {
012 ♯ (np)	028 L (fs)	044 ,	060 <	076 L	092 \	108 l	124
013 (cr)	029 ↔ (gs)	045 −	061 =	077 M	093]	109 m	125 }
014 ♀ (so)	030 ▲ (rs)	046 .	062 >	078 N	094 ^	110 n	126 ~
015 ✱ (si)	031 ▼ (us)	047 /	063 ?	079 O	095 _	111 o	127 △

HAI

CONTOH MODULAR ARITHMETIC CONT...

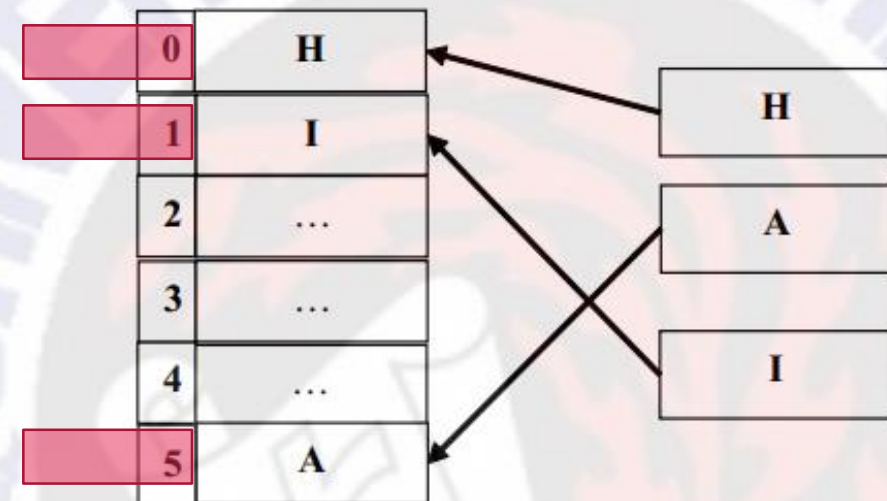
- Misalnya dibuat Tabel hash yang memetakan huruf A-Z
- Dibuat kunci yang memetakan ASCII yang diinputkan
- Dibuat sistem modulus dari 6

• Misal: input → **HAI**

- $H \rightarrow 72\%6 = 0$
- $A \rightarrow 65\%6 = 5$
- $I \rightarrow 73\%6 = 1$

1. Ambil nilai ASCII dari H (yaitu 72) lalu dibagi dengan 6, sisa hasil baginya (0) menjadi kunci untuk nilai H.
2. Ambil nilai ASCII dari A (yaitu 65) lalu dibagi dengan 6, sisa hasil baginya (5) menjadi kunci untuk nilai A.
3. Ambil nilai ASCII dari I (yaitu 73) lalu dibagi dengan 6, sisa hasil baginya (1) menjadi kunci untuk nilai I.

Gambar 1 memperlihatkan hasil pemetaan data masukan tersebut ke dalam tabel hash.



Gambar 1 Pemetaan Data ke Dalam Tabel Hash

DISKUSI KELOMPOK I

Hash (x) = x mod 10

Key : 45, 72, 39, 48, 56, 77, 91, 63, 84, 90

0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

Hash (x) = x mod 6

Key : 45, 72, 39, 48, 56, 77, 91, 63, 84, 90

0	
1	
2	
3	
4	
5	

JAWABAN

Hash (x) = x mod 10

Key : 45, 72, 39, 48, 56, 77, 91, 63, 84, 90

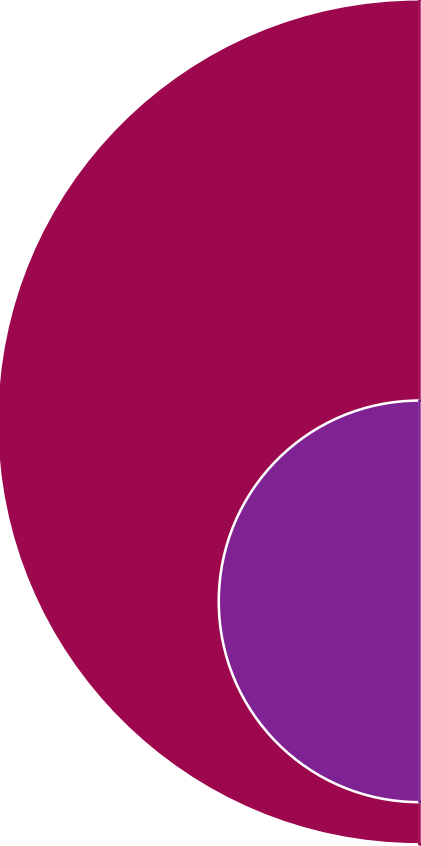
0	90
1	91
2	72
3	63
4	84
5	45
6	56
7	77
8	48
9	39

Hash (x) = x mod 6

Key : 45, 72, 40, 49, 14,

0	72
1	49
2	14
3	45
4	40
5	

FUNGSI HASH : PEMOTONGAN (TRUNCATION)



Dilakukan dengan mengabaikan bagian - bagian tertentu dalam kunci dan menggunakan yang tersisa sebagai indeks untuk mengakses data dalam tabel hash.

Sebagian dari key dapat dibuang/diabaikan, bagian key sisanya digabungkan untuk membentuk index.

CONTOH PEMOTONGAN (TRUNCATION)

- Key : 222345654, 301657434, 123882345, 125456789

Truncation **Delete 6 digit terakhir**

222345654 = 222 ; simpan 222345654 dilokasi 222

301657434 = 301; simpan 301657434 dilokasi 301

123882345 = 123; simpan 123882345 dilokasi 123

125456789 = 125; simpan 123456789 dilokasi 125

CONTOH PEMOTONGAN (TRUNCATION) CONT

contoh:

Phone no:

731-3018

539-2309

428-1397

index

338

329

217

FUNGSI HASH²⁷: PELIPATAN (*FOLDING*)



Kunci dibagi-bagi menjadi beberapa bagian misalnya per dua digit kemudian dijumlahkan .



Hasilnya dipotong sehingga masuk jangkauan indeks dalam tabel hash



Data dipecah menjadi beberapa bagian, kemudian tiap bagian tersebut digabungkan lagi dalam bentuk lain.

CONTOH PELIPATAN (*FOLDING*)

- Contoh: kunci 123456, 234351, 222456, 321654, dilipat menjadi 2 bagian, setiap 3 digit.
- Maka:
 $123+654 = 777$; simpan 123456 dilokasi 777
 $234+153 = 387$; simpan 234351 dilokasi 387
 $222+654 = 876$; simpan 222456 dilokasi 876
 $321+456 = 777$; simpan 321654 dilokasi 777
- Dari perhitungan terjadi kolisi untuk nomor 123456 dan 321654

CONTOH PELIPATAN (*FOLDING*) CONT...

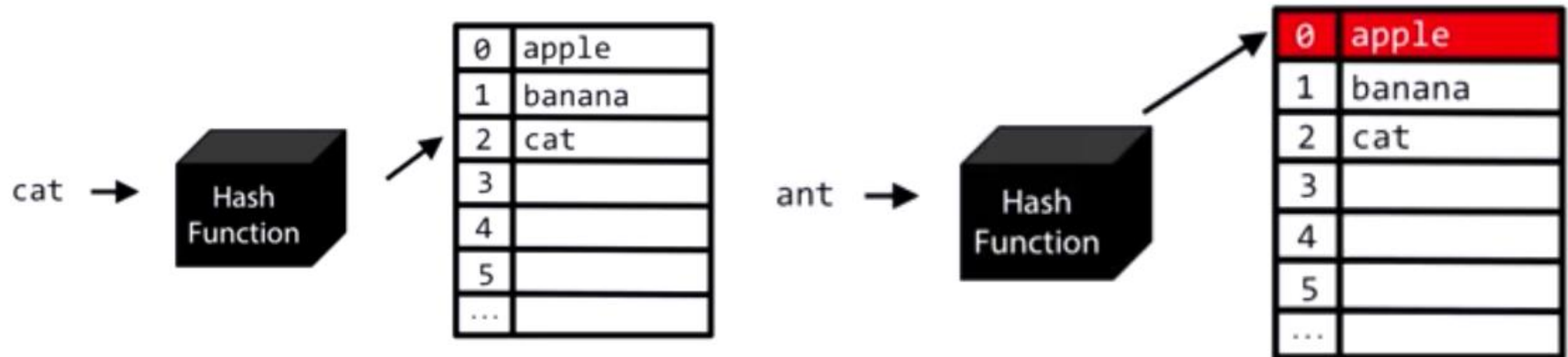
contoh.

<i>Phone no:</i>	<i>3-group</i>	<i>index</i>
7313018	73+13+018	104
5392309	53+92+309	454
4281397	42+81+397	520

FUNGSI HASH : PERKALIAN (*MULTIPLICATION*)

- Contoh: kunci 12345789 , kalikan 3 digit pertama dengan 3 digit terakhir dari kunci.
- Maka:
 $123 \times 789 = 97047$; simpan 12345789 dilokasi 97047

TABRAKAN (COLLISION)



Collision Occurs, when hash function maps 2 or more keys to same Index.

MENANGANI TABRAKAN¹² (COLLISION RESOLUTION)



Collision Resolution: Penyelesaian bila terjadi *collision* (*tabrakan*)

Situasi yang membuat beberapa kunci memiliki alamat hash yang sama atau disebut dengan tabrakan hash (*hash collision*).

Dikatakan terjadi *collision* jika dua buah *keys* dipetakan pada sebuah sel.

Collision bisa terjadi saat melakukan *insertion*.

Dibutuhkan prosedur tambahan untuk mengatasi terjadinya *collision*.

METODE MENANGANI TABRAKAN

Pengalamatan
Terbuka (*Open
Addressing*)
– ***Closed Hashing***

Pembentukan Rantai
(*Chaining*)
– ***Open Hashing***

Pengalamatan Buket
(***Bucket Addressing***)

OPEN ADDRESSING - CLOSED HASHING

34

Pada pengalamatan terbuka semua elemen disimpan dalam tabel hash itu sendiri.

Ide: mencari alternatif sel lain pada tabel

Pada proses insertion, coba sel lain sesuai urutan dengan menggunakan fungsi pencari urutan seperti berikut:

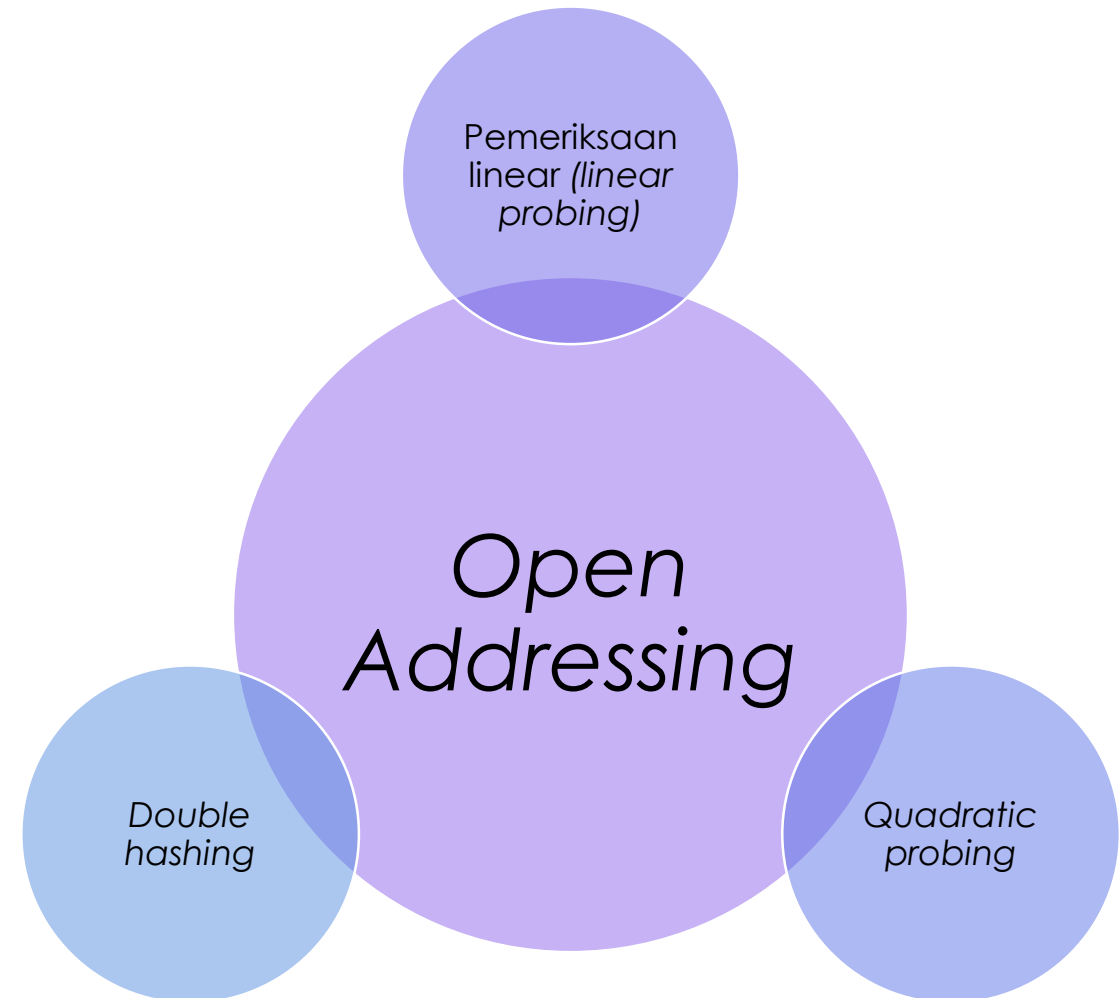
$$h_i(x) = (\text{hash}(x) + \mathbf{f(i)}) \bmod H\text{-size} \quad \mathbf{f(0) = 0}$$

Fungsi f digunakan sebagai pengatur *strategy collision resolution*.
Bagaimana bentuk fungsi \mathbf{f} ?

OPEN ADDRESSING - CLOSED HASHING CONT...

35

Beberapa strategi/alternatif untuk menentukan bentuk fungsi f , yaitu:



CONTOH CLOSED HASHING

Key yang bertabrakan ditaruh di slot yang tak terpakai

Misalnya

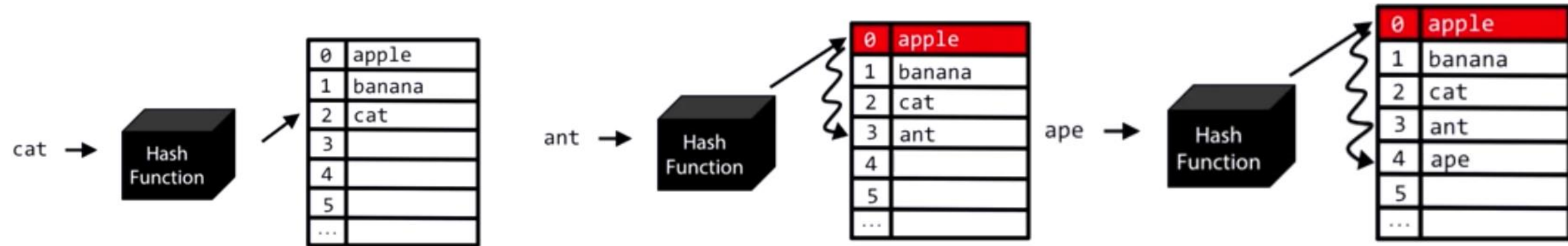
- $k \bmod m = k \bmod 100$

Jika kita ingin meng input kan angka 8002, seharusnya ditaruh di index 2

Dengan menggunakan open addressing angka 8002 ditaruh di index 4

0	5000
1	9001
2	2002
3	7003
4	8002

CONTOH CLOSED HASHING ³⁷ CONT...



LINEAR PROBING³⁸

Fungsi linear relatif paling sederhana dan mudah diimplementasikan.

Gunakan fungsi linear
 $f(i) = i$

Tabrakan hash ditangani dengan mencari lokasi terdekat yang masih kosong atau yang dikenal dengan pemeriksaan linear.

Bila terjadi collision, cari posisi pertama pada tabel yang terdekat dengan posisi yang seharusnya.

LINEAR PROBING CONT...

- Dapat menimbulkan masalah:

primary clustering

□ Elemen-elemen yang menurut perhitungan hash diletakkan pada lokasi sel berbeda, diarahkan pada sel pengganti yang sama

- Adapun fungsi hash bisa menjadi :

$$h(k,i) = (h'(k) + i) \bmod m, m=0..n-1$$

- Contoh urutan pencarian sbb :

$$h+1, h+2, h+3, \dots, h+i$$

- Linear Probing hanya disarankan untuk ukuran hash table yang ukurannya lebih besar dua kali dari jumlah data.

CONTOH LINEAR PROBING⁴⁰

- Tabel hash dengan fungsi
- Hash (x) = $x \bmod 10$
- Data yang ingin diinput : 89, 18, 49, 58, 9

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

- $H(89) = 89 \% 10 = 9$
- $H(18) = 18 \% 10 = 8$
- $H(49) = 49 \% 10 = 9 \rightarrow \text{collision}$
 - $H_1(49) = (9 + 1) \% 10 = 0$
- $H(58) = 58 \% 10 = 8 \rightarrow \text{collision}$
 - $H_1(58) = (8 + 1) \% 10 = 9$
 - $H_2(58) = (8 + 2) \% 10 = 0$
 - $H_3(58) = (8 + 3) \% 10 = 1 \quad (\checkmark)$

- $H(9) = 9 \% 10 = 9 \rightarrow \text{collision}$
 - $H_1(9) = (9 + 1) \% 10 = 0$
 - $H_2(9) = (9 + 2) \% 10 = 1$
 - $H_3(9) = (9 + 3) \% 10 = 2 \quad (\checkmark)$

CONTOH LINEAR PROBING CONT...

	After insert 89	After insert 18	After insert 49	After insert 58	After insert 9
0			→ 49	49	49
1				→ 58	58
2					→ 9
3					
4					
5					
6					
7					
8		→ 18	18	18	18
9	→ 89	89	89	89	89

- As long as table is big enough, a free cell can always be found, but the time to do so can get quite large.
- Worse, even if the table is relatively empty, blocks of occupied cells start forming.
- This effect is known as *primary clustering*.

DISKUSI KELOMPOK 2

- Misal terdapat data sebagai berikut



Rekaman	A	B	C	K	P	Q	R	Y	Z
$H(k)$	5	6	7	5	0	1	2	9	0

- Data diatas dimasukkan ke dalam urutan yang sama.
- Silahkan isi key pada hash table disamping menggunakan linear probing

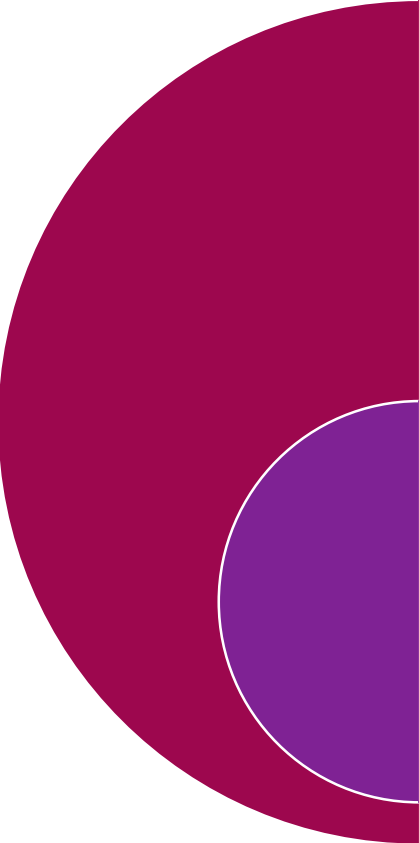
0	
1	
2	
3	
4	
5	
6	
7	
8	
9	

PEMBAHASAN DISKUSI KELOMPOK 2

- Terjadi collision key K
 - K seharusnya berada di index 5 ditempatkan di index 8
 - Melalui 4 probe
- Terjadi collision key Z
 - Z seharusnya berada di index 0 ditempatkan di index 3
 - Melalui 4 probe

0	P Z	
1	Q	
2	R	
3	Z	
4		
5	A K	
6	B	
7	C	
8	K	
9	Y	

PROBE BERHASIL DAN PROBE GAGAL



Probe berhasil adalah dengan menghitung banyaknya alamat dari alamat yang seharusnya informasi tersebut berada sampai alamat dimana informasi tersebut dicatat.

Probe gagal adalah dengan menjumlahkan probe yang diperlukan untuk mencari alamat kosong yang terdekat oleh setiap alamat yang ada.

PEMBAHASAN DISKUSI KELOMPOK 2

CONT...

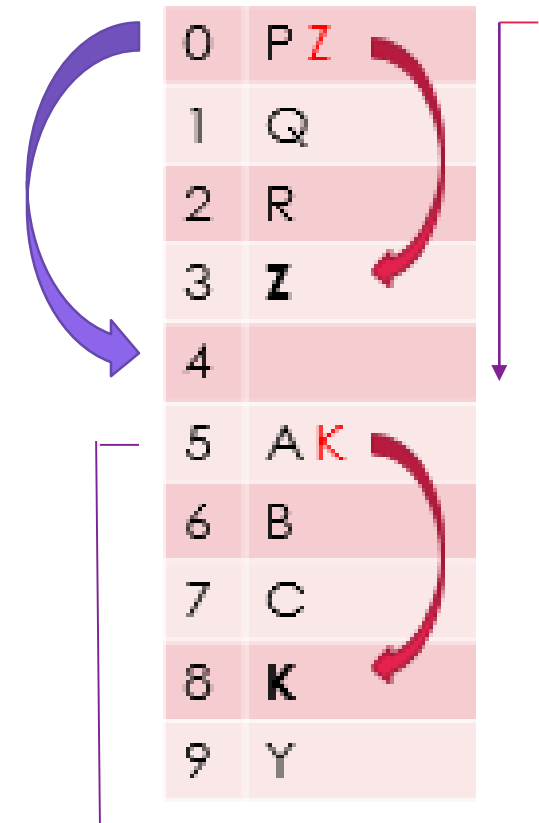
- Probe rata-rata pencarian **berhasil**

Rekaman	A	B	C	K	P	Q	R	Y	Z
---------	---	---	---	---	---	---	---	---	---

- $B = (1 + 1 + 1 + 4 + 1 + 1 + 1 + 1 + 4) / 9$
- $B = 15 / 9 = 1.667$

- Probe rata-rata pencarian **gagal**

- P Q R Z - A B C K Y
- $B = (5 + 4 + 3 + 2 + 1 + 10 + 9 + 8 + 7 + 6) / 10$
- $B = 55 / 10 = 5.5$



DISKUSI KELOMPOK 3

- ▶ Fungsi hash yang dipakai adalah :
 $f(\text{key}) = \text{key} \bmod 10$
- ▶ Ruang alamat yang tersedia : 10 alamat
- ▶ Metode *Collision Resolution* yang dipakai adalah *Open Addressing* dengan *Linear Probing* jarak 3
- ▶ Urutan kunci yang masuk adalah : 20, 31, 33, 40, 10, 12, 30, dan 15

PEMBAHASAN DISKUSI KELOMPOK 3

Key	f	Proses	Addr
20	0	0	0
31	1	1	1
33	3	3	3
40	0	0, 3, 6	6
10	0	0, 3, 6, 9	9
12	2	2	2
30	0	0, 3, 6, 9, 2, 5	5
15	5	5, 8	8

Probe total = 19

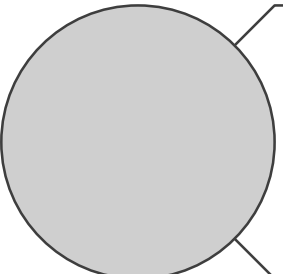
Probe rata-rata = $19/8$

0	20
1	31
2	12
3	33
4	
5	30
6	40
7	
8	15
9	10

KELEMAHAN LINEAR PROBING



Data cenderung untuk mengumpul pada satu tempat



Hal ini bisa dipahami karena jika ada suatu data yang akan disisipkan pada suatu alamat dan alamat yang dimaksud sudah dipakai, maka data baru tersebut akan ditempatkan pada lokasi berikutnya yang letaknya berurutan.



Primary clustering

OPEN ADDRESSING: PEMERIKSAAN KUADRATIK (QUADRATIC PROBING)

Menghindari *primary clustering* dengan menggunakan fungsi:
 $f(i) = i^2$

Menimbulkan banyak permasalahan bila hash table telah terisi lebih dari setengah.

Perlu dipilih ukuran hash table yang bukan bilangan kuadrat.

Dengan ukuran hash table yang merupakan bilangan prima dan hash table yang terisi kurang dari setengah, strategy quadratic probe dapat selalu menemukan lokasi untuk setiap elemen baru.

OPEN ADDRESSING: PEMERIKSAAN KUADRATIK (QUADRATIC PROBING) CONT...

Pemeriksaan Kuadratik yang dilakukan dengan urutan sbb :

- $h(k,i) = (h'(k) + i^2) \bmod m, i=0..n-1$

Contoh urutan pencarian sbb :

- $h, h+1, h+4, h+9, \dots, h+i^2$

Dapat melakukan *increment* bila terjadi *collision*

Quadratic Probing -- Example

- Example:
 - Table Size is 11 (0..10)
 - Hash Function: **$h(x) = x \bmod 11$**
 - Insert keys: 20, 30, 2, 13, 25, 24, 10, 9
 - $20 \bmod 11 = 9$
 - $30 \bmod 11 = 8$
 - $2 \bmod 11 = 2$
 - $13 \bmod 11 = 2 \rightarrow 2+1^2=3$
 - $25 \bmod 11 = 3 \rightarrow 3+1^2=4$
 - $24 \bmod 11 = 2 \rightarrow 2+1^2, 2+2^2=6$
 - $10 \bmod 11 = 10$
 - $9 \bmod 11 = 9 \rightarrow 9+1^2, 9+2^2 \bmod 11, 9+3^2 \bmod 11 = 7$

0	
1	
2	2
3	13
4	25
5	
6	24
7	9
8	30
9	20
10	10

OPEN ADDRESSING: DOUBLE HASHING⁵³

Double hashing, pemeriksaan dilakukan dengan urutan sbb :

- $h(k,i)=(h_1(k)+ih_2(k))\bmod m$

Dengan h_1 dan h_2 adalah fungsi hash contoh pemeriksaan dengan *double hashing*:

- $h_1, h_1+h_2, h_1+2h_2, h_1+3h_2, \dots, h_1+i \times h_2$

Baik pada pemeriksaan kuadratik maupun *double hashing*, h menyatakan alamat hash yang diperoleh melalui fungsi hash dan i dimulai dari 0.

OPEN ADDRESSING. DOUBLE HASHING CONT...

Fungsi untuk *collision resolution* disusun dengan fungsi hash seperti :

$$f(i) = i * \text{hash2}(x)$$

Setiap saat faktor $\text{hash2}(x)$ ditambahkan pada *probe*.

Harus hati-hati dalam memilih fungsi hash kedua untuk menjamin agar tidak menghasilkan nilai 0 dan mem-probe ke seluruh sel.

Salah satu syaratnya ukuran hash table haruslah bilangan prima.

CONTOH DOUBLE HASHING

Consider a hash table storing integer keys that handles collision with double hashing

- $N = 13$
- $h(k) = k \bmod 13$
- $d(k) = 7 - k \bmod 7$

Insert keys 18, 41, 22, 44, 59, 32, 73, in this order

k	$h(k)$	$d(k)$	Probes	
18	5	3	5	
41	2	1	2	
22	9	6	9	
44	5	5	5	10
59	7	4	7	
32	6	3	6	
73	8	4	8	

0	1	2	3	4	5	6	7	8	9	10	11	12



	41			18	32	59	73	22	44			
0	1	2	3	4	5	6	7	8	9	10	11	12

TEKNIK PEMBENTUKAN RANTAI⁵⁶ (CHAINING-OPEN HASHING)

- Permasalahan *Collision* diselesaikan dengan menambahkan seluruh elemen yang memilih nilai hash sama pada sebuah set.
- Menyediakan sebuah linked list untuk setiap elemen yang memiliki nilai hash sama.
- Tiap sel pada hash table berisi pointer ke sebuah linked list yang berisikan data/elemen.

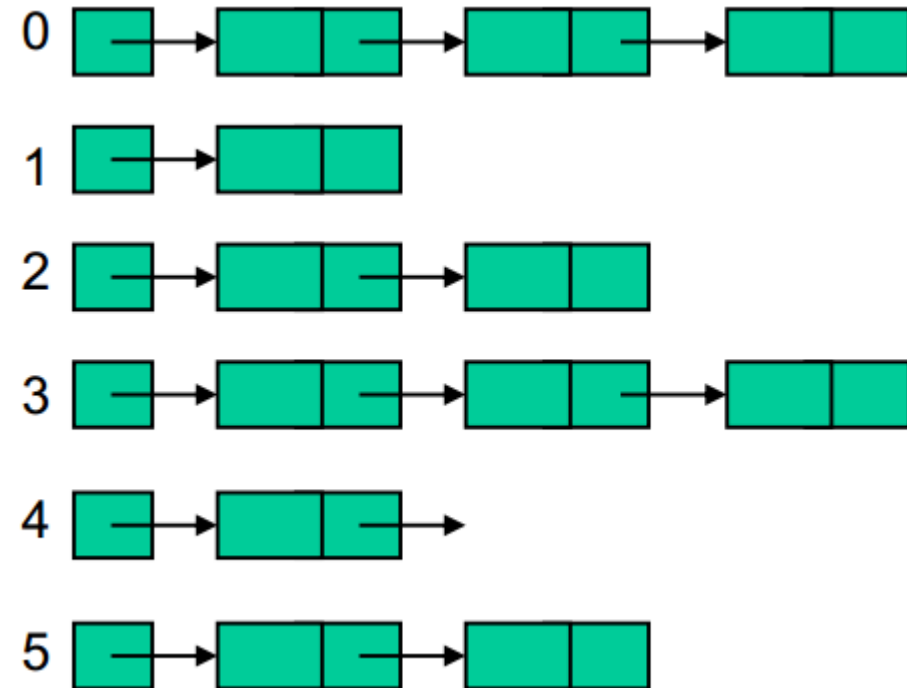
TEKNIK PEMBENTUKAN RANTAI (CHAINING-OPEN HASHING) CONT...

- Menambahkan sebuah elemen ke dalam tabel. Dilakukan dengan menambahkan elemen pada akhir atau awal linkedlist yang sesuai dengan nilai hash.
- Bergantung apakah perlu ada pengujian nilai duplikasi atau tidak.
- Dipengaruhi berapa sering elemen terakhir akan diakses.
- Untuk pencarian, gunakan fungsi hash untuk menentukan linked list mana yang memiliki elemen yang dicari, kemudian lakukan pembacaan terhadap linked list tersebut.

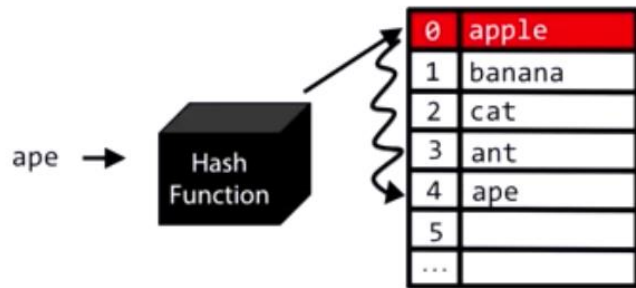
TEKNIK PEMBENTUKAN RANTAI⁵⁸ (CHAINING-OPEN HASHING) CONT...

- Penghapusan dilakukan pada linked list setelah pencarian elemen dilakukan.
- Dapat saja digunakan struktur data lain selain linked list untuk menyimpan elemen yang memiliki fungsi hash yang sama tersebut.
- Kelebihan utama dari metode ini adalah dapat menyimpan data yang tak terbatas. (dynamic expansion).
- Kekurangan utama adalah penggunaan memory pada tiap sel.

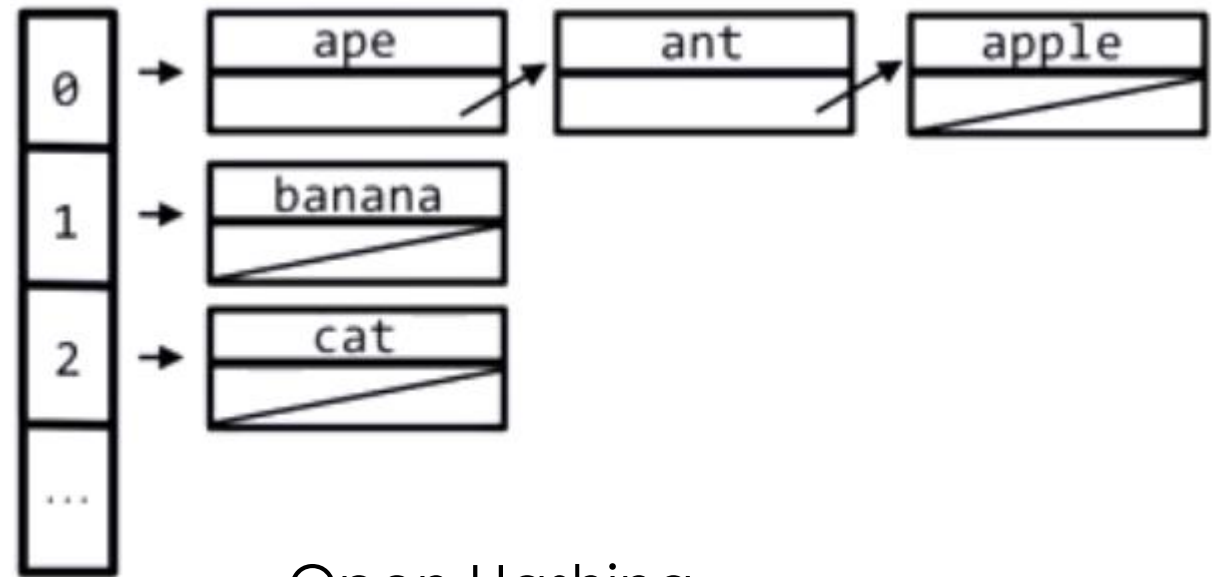
OPEN HASHING



OPEN VS CLOSED HASHING⁶⁰



- Closed Hashing



- Open Hashing

CONTOH OPEN HASHING⁶¹

It stores colliding keys as linked list.



Advantages

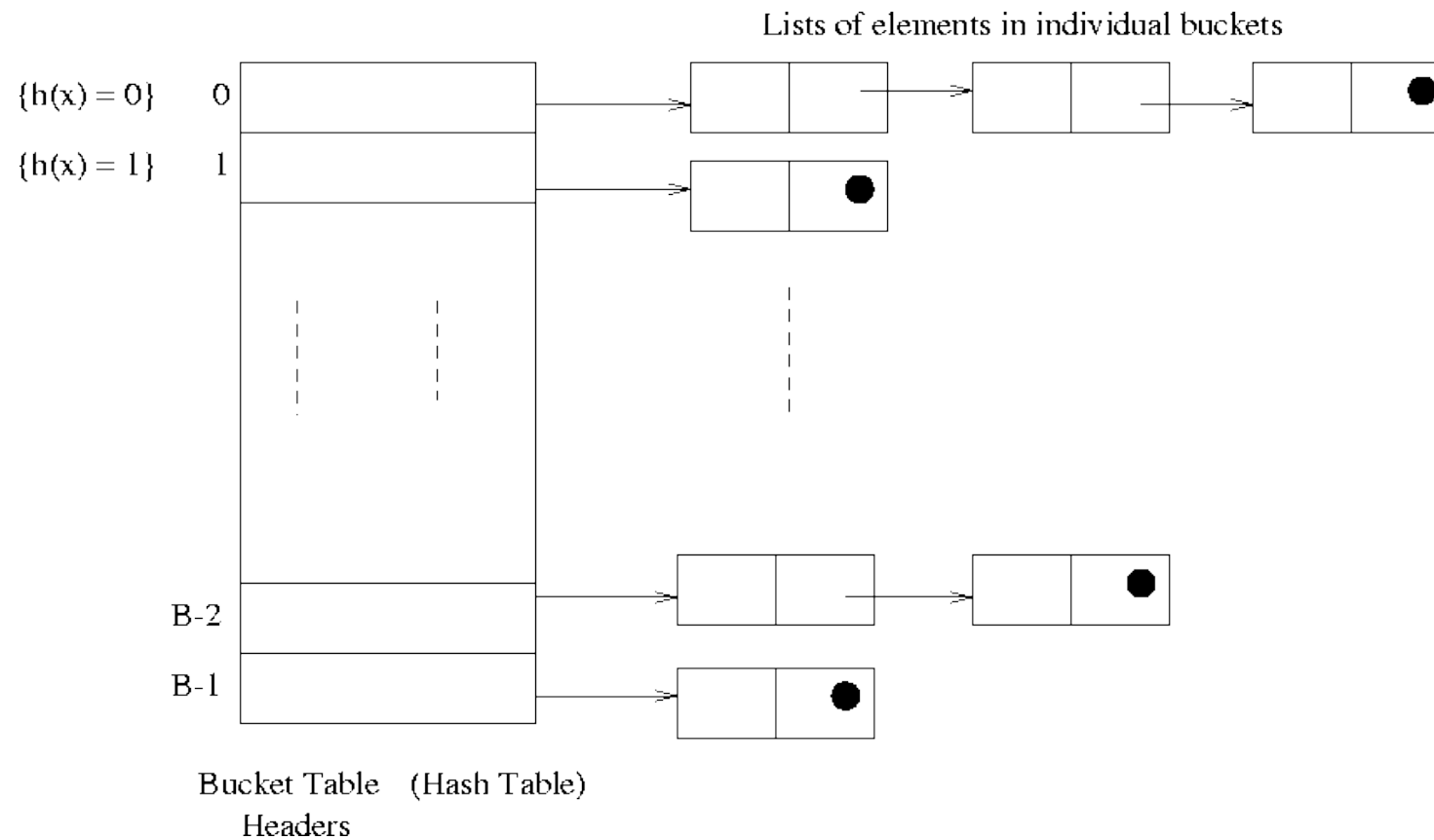
Faster to Access, series of Elements.

BUCKET ADDRESSING⁶²

Teknik pengalamatan buket mirip dengan pembentukan rantai , namun tabrakan tidak ditangani dengan link list, melainkan dengan *array*.

Buket sendiri diartikan sebagai sebuah blok ruang memori yang cukup untuk menampung sejumlah data yang memiliki alamat hash yang sama

BUCKET ADDRESSING



CONTOH BUCKET ADDRESSING

Green	30
Hall	30
Jenk	32
King	33
Land	33
Mark	33
Nutt	33

BUCKET ADDRESS	BUCKET CONTENTS		
30	Green ..	Hall ..	
31			
32	Jenks ..		
33	King ..	Land ..	Marks ..

overflow

RANGKUMAN

- Hash tables: array
- Hash function: Fungsi yang memetakan keys menjadi bilangan $[0 \rightarrow \text{ukuran dari hash table})$
- Collision resolution
 - Open hashing
 - Separate chaining
 - Closed hashing (Open addressing)
 - Linear probing
 - Quadratic probing
 - Double hashing
 - Primary Clustering, Secondary Clustering

- Advantage
 - Cocok untuk merepresentasikan data dengan frekuensi insert, delete dan search yang tinggi.
- Disadvantage
 - ☐ Sulit (tidak efficient) untuk mencetak seluruh elemen pada hash table
 - ☐ Tidak efficient untuk mencari elemen minimum or maximum
 - ☐ Tidak bisa di expand (untuk closed hash/open addressing)
 - ☐ Ada pemborosan memory/space

REFERENSI

Barnes & Noble, Hash Tables, *Sparknotes*, <http://www.sparknotes.com/cs/searching/hashtables/section1.html>.

- Kadir, Abdul. 2013. Teori dan Aplikasi Struktur Data menggunakan C++. Yogyakarta : Penerbit ANDI
- http://www.cs.auckland.ac.nz/software/AlgAnim/hash_tables.html