Struktur Data
Pendidikan Teknik Informatika
Universitas Negeri Padang
TIM DOSEN ©2022

JOBSHEET (JS-13)

Hashing and Collition

Fakultas	Proram Studi	Kode MK	Waktu
Teknik	Pendidikan Teknik Informatika	TIK1.61.2317	4 x 50 Menit

TUJUAN PRAKTIKUM

1. Mahasiswa mampu mengaplikasikan konsep hashing and collition dalam menyelesaikan kasus.

HARDWARE & SOFTWARE

1. Personal Computer
2. IDE: Dev C++

TEORI SINGKAT

A. Hashing

Hashing adalah teknik pemetaan sekumpulan besar data arbitrer ke indeks tabel menggunakan fungsi hash . Ini adalah metode untuk mewakili kamus untuk kumpulan data besar. Hal ini memungkinkan pencarian, pembaruan dan operasi pengambilan terjadi dalam waktu yang konstan.

Setelah menyimpan sejumlah besar data, kita perlu melakukan berbagai operasi pada data tersebut. Pencarian tidak dapat dihindari untuk kumpulan data. Pencarian linier dan pencarian biner melakukan pencarian/pencarian dengan kompleksitas waktu masing-masing. Ketika ukuran kumpulan data meningkat, kompleksitas ini juga menjadi sangat tinggi yang tidak dapat diterima. Fungsi hash digunakan untuk memetakan setiap elemen dari kumpulan data ke indeks dalam tabel.

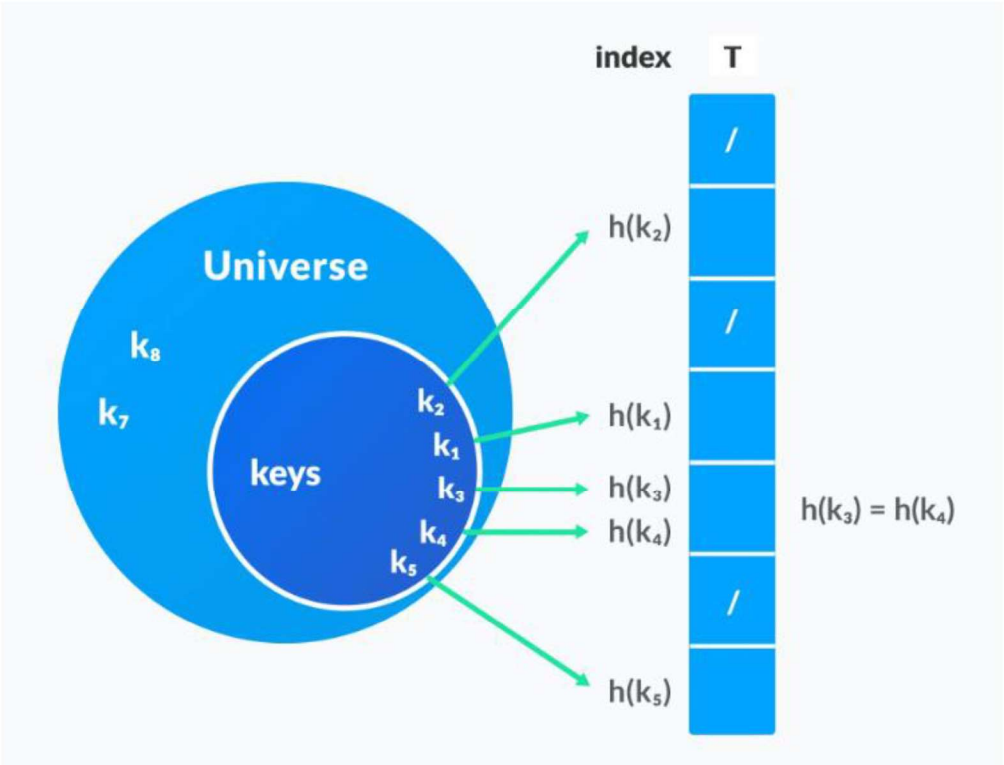
B. Hash Table

Struktur data hash table menyimpan elemen dalam pasangan nilai kunci di mana :

- **Key** : bilangan bulat unik yang digunakan untuk mengindeks nilai
- **Value** : data yang terkait dengan kunci.

Key and Value in Hash table

Dalam hash table, indeks baru diproses menggunakan kunci. Dan, elemen yang sesuai dengan kunci itu disimpan dalam file index. Proses ini disebut **hashing**. Membiarkan k menjadi kunci dan h(x) menjadi fungsi hash. Di sini, h(k) akan memberi kita indeks baru untuk menyimpan elemen yang terhubung dengan k.



Hash table Representation

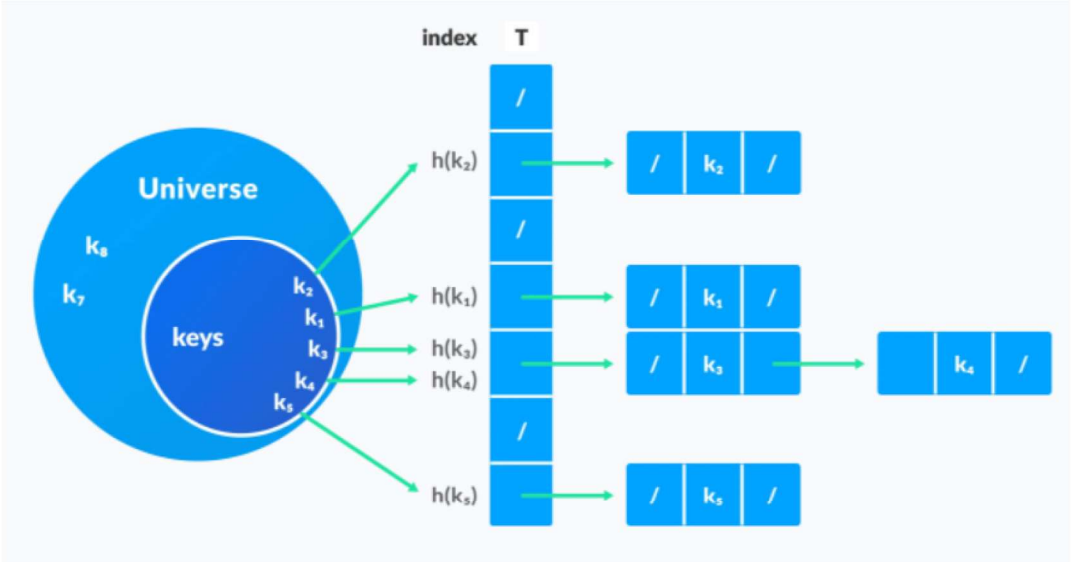
C. Hash Collition

Ketika fungsi hash menghasilkan indeks yang sama untuk beberapa kunci, akan ada konflik (nilai apa yang akan disimpan dalam indeks itu). Ini disebut **hash collition**. Kita dapat menyelesaikan hash collition menggunakan salah satu teknik berikut :

- Collision resolution by chaining
- Open adressing: Penyelidikan Linier/Kuadrat dan Hashing Ganda

1. Collision resolution by chaining

Dalam chaining, jika fungsi hash menghasilkan indeks yang sama untuk beberapa elemen, elemen ini disimpan dalam indeks yang sama dengan menggunakan daftar tertaut ganda. Jika j adalah slot untuk beberapa elemen, itu berisi penunjuk ke kepala daftar elemen. Jika tidak ada elemen, j mengandung NIL.



Collision Resolution using chaining

2. Open Adressing

Tidak seperti chaining, open adressing tidak menyimpan banyak elemen ke dalam slot yang sama. Di sini, setiap slot diisi dengan satu tombol atau kiri NIL.

Berbagai teknik yang digunakan dalam open adressing adalah:

a. Linear Probing

Dalam linier probing, collition diselesaikan dengan memeriksa slot berikutnya.

```
h(k, i) = (h'(k) + i) mod m
```

di mana :

- i = {0, 1, ….}
- h'(k) adalah fungsi hash baru

Jika collition terjadi pada h(k, 0), maka h(k, 1)diperiksa. Dengan cara ini, nilai dari i bertambah secara linier. Masalah dengan linier probing adalah bahwa sekelompok slot yang berdekatan terisi. Saat memasukkan elemen baru, seluruh cluster harus dilalui. Ini menambah waktu yang dibutuhkan untuk melakukan operasi pada tabel hash.

b. Quadratic Probing

Ini bekerja mirip dengan linier probing tetapi jarak antara slot ditingkatkan (lebih besar dari satu) dengan menggunakan hubungan berikut.

$$h(k, i) = (h'(k) + c_1 i + c_2 i^2) \bmod m$$

di mana :

- c_1 dan c_2 adalah konstanta bantu positif,
- i = {0, 1, ….}

c. Double Hashing

Jika collition terjadi setelah menerapkan fungsi hash h(k), maka fungsi hash lain dihitung untuk menemukan slot berikutnya :

```
h(k, i) = (h1(k) + ih2(k)) mod m
```

D. Hash Functions

Sebuah fungsi hash yang baik mungkin tidak mencegah collition sepenuhnya namun dapat mengurangi jumlah collition. Di sini, kita akan melihat berbagai metode untuk menemukan fungsi hash yang baik :

1. Metode Pembagian

Jika k adalah kunci dan m merupakan ukuran tabel hash, fungsi hash h() dihitung sebagai:

```
h(k) = k mod m
```

Misalnya, Jika ukuran tabel hash adalah 10 dan k = 112 kemudian h(k) = 112 mod 10 = 2. Nilai dari m tidak boleh merupakan pangkat dari 2. Ini karena pangkat 2 dalam format biner adalah 10, 100, 1000, …. Ketika kita menemukan k mod m, kita akan selalu mendapatkan p-bit orde rendah.

```
if m = 22, k = 17, then h(k) = 17 mod 22 = 10001 mod 100 = 01
if m = 23, k = 17, then h(k) = 17 mod 22 = 10001 mod 100 = 001
if m = 24, k = 17, then h(k) = 17 mod 22 = 10001 mod 100 = 0001
if m = 2p, then h(k) = p lower bits of m
```

2. Metode Perkalian

```
h(k) = ⌊m(kA mod 1)⌋
```

di mana :

- kA mod 1 memberikan bagian pecahan kA,
- ⌊ ⌋ memberikan nilai dasar
- A adalah selalu konstan. Nilai A terletak antara 0 dan 1. Tapi, pilihan yang optimal akan ≈ (√5-1)/2 disarankan oleh Knuth.

3. Universal Hashing

Dalam universal hashing, fungsi hash dipilih secara acak terlepas dari kunci.

LATIHAN

1. Hash Table

```
/* Nama File   : hash table
Pembuat        : tuliskan nama dan NIM anda
Tgl pembuatan : tuliskan tanggal hari ini*/

// Implementing hash table in C

#include <stdio.h>
#include <stdlib.h>

struct set
{
  int key;
```

```
  int data;
};
struct set *array;
int capacity = 10;
int size = 0;

int hashFunction(int key)
{
  return (key % capacity);
}
int checkPrime(int n)
{
  int i;
  if (n == 1 || n == 0)
  {
  return 0;
  }
  for (i = 2; i < n / 2; i++)
  {
  if (n % i == 0)
  {
    return 0;
  }
  }
  return 1;
}
int getPrime(int n)
{
  if (n % 2 == 0)
  {
  n++;
  }
  while (!checkPrime(n))
  {
  n += 2;
  }
```

```
  return n;
}
void init_array()
{
  capacity = getPrime(capacity);
  array = (struct set *)malloc(capacity * sizeof(struct
set));
  for (int i = 0; i < capacity; i++)
  {
  array[i].key = 0;
  array[i].data = 0;
  }
}

void insert(int key, int data)
{
  int index = hashFunction(key);
  if (array[index].data == 0)
  {
  array[index].key = key;
  array[index].data = data;
  size++;
  printf("\n Key (%d) has been inserted \n", key);
  }
  else if (array[index].key == key)
  {
  array[index].data = data;
  }
  else
  {
  printf("\n Collision occured  \n");
  }
}

void remove_element(int key)
{
```

```
  int index = hashFunction(key);
  if (array[index].data == 0)
  {
  printf("\n This key does not exist \n");
  }
  else
  {
  array[index].key = 0;
  array[index].data = 0;
  size--;
  printf("\n Key (%d) has been removed \n", key);
  }
}
void display()
{
  int i;
  for (i = 0; i < capacity; i++)
  {
  if (array[i].data == 0)
  {
    printf("\n array[%d]: / ", i);
  }
  else
  {
    printf("\n key: %d array[%d]: %d \t", array[i].key, i,
array[i].data);
  }
  }
}

int size_of_hashtable()
{
  return size;
}

int main()
```

```
{
  int choice, key, data, n;
  int c = 0;
  init_array();

  do
  {
  printf("1.Insert item in the Hash Table"
     "\n2.Remove item from the Hash Table"
     "\n3.Check the size of Hash Table"
     "\n4.Display a Hash Table"
     "\n\n Please enter your choice: ");

  scanf("%d", &choice);
  switch (choice)
  {
  case 1:

    printf("Enter key -:\t");
    scanf("%d", &key);
    printf("Enter data -:\t");
    scanf("%d", &data);
    insert(key, data);

    break;

  case 2:

    printf("Enter the key to delete-:");
    scanf("%d", &key);
    remove_element(key);

    break;

  case 3:
```

```
            n = size_of_hashtable();
            printf("Size of Hash Table is-:%d\n", n);

            break;

        case 4:

            display();

            break;

        default:

            printf("Invalid Input\n");
        }

        printf("\nDo you want to continue (press 1 for yes): ");
        scanf("%d", &c);

        } while (c == 1);
    }
```

DAFTAR PUSTAKA

Kelley, Al and Pohl, Ira. 2003. *C by Dissection: The Essentials of C Programming*. Addison-Wcaslcy.

Khannedy, Eko Kurniawan. 2007. *Diktat Pemrograman C*.Bandung : Unikom.

Liem, Inggriani. 2003. *Catatan Singkat Bahasa C*. Jurusan Teknik Informatika. Institut Teknologi Bandung.

Reema Thareja. 2014. *Data Structures Using C*. India : Oxford University Press.

Solichin, Achmad. 2003. *Pemrograman Bahasa C dengan Turbo C*. Kuliah Berseri ilmukomputer.com