Modul Praktikum DAA:
Hashing#1-Hash Functions

A. Modul Praktikum

1. Truncation Function

Dengan truncation, sebuah key dimasukkan ke dalam tabel hash dengan cara memenggal atau memotong key. Contohnya, jika kita hendak menyimpan data mahasiswa, dan key yang digunakan adalah NPM mahasiswa tersebut, maka kita dapat menggunakan truncation function untuk mengambil k-digit pada NPM sebagai indeks tabel (hash value). Teknik ini cepat namun seringkali tidak bisa mendistribusikan key ke dalam tabel secara merata.

Contoh: mengambil 3 digit terakhir dari NPM

- Mahasiswa dengan NPM 2013730042 akan disimpan ke dalam indeks ke-42 di dalam tabel.
- Mahasiswa dengan NPM 2012730010 akan disimpan ke dalam indeks ke-10 di dalam tabel.
- Mahasiswa dengan NPM 2011730042 tidak dapat disimpan ke dalam indeks ke-42 karena indeks tersebut sudah diisi dengan NPM 2013730042.

Latihan: Membuat Truncation Function

NPM dan nama mahasiswa/i UNPAR akan disimpan ke dalam sebuah tabel hash. Key yang digunakan adalah NPM, diambil tiga digit terakhir sebagai alamat indeks. Tabel hash yang digunakan berukuran 150.

Langkah-langkah:

a) Buatlah sebuah kelas dengan nama Data. Kelas Data memiliki atribut berupa key (String) dan atribut value (String). Tambahkan constructor dengan parameter key dan value seperti di bawah ini.

```
1  public class Data{
2     String key;
3     String value;
4     Data(String key, String value){
5        this.key=key;
6        this.value=value;
7     }
8  }
```

b) Buatlah sebuah kelas lain bernama TruncationHash. Tambahkan constructor pada kelas tersebut seperti di bawah ini:

```
5     public TruncationHash(int size){
6        this.M=size;
7        hashTable=new Data[this.M];
8     }
9
```

c) Untuk bisa melakukan hashing, maka kelas Truncation harus memiliki method yang tugasnya adalah melakukan hashing dengan fungsi truncation. Tambahkan sebuah method baru dengan nama truncateFunction pada kelas Truncation. Fungsi hash truncation melakukan hash terhadap key berupa String, tambahkan parameter bernama key pada method truncateFunction seperti di bawah ini:

```
10    public int truncateFunction(String key){
11        int key_length=key.length();
12
13        //Fungsi hash truncate
14        String truncated_key=key.substring(key_length-3, key_length);
15
16        int hashValue=Integer.parseInt(truncated_key);
17        return hashValue;
18    }
```

Method truncateFunction akan mengembalikan nilai berupa alamat indeks (bertipe int) pada tabel hash, maka method truncateFunction harus bertipe int.

d) Untuk menyimpan data ke dalam tabel hash, diperlukan method lain bernama insert. Buatlah method insert dengan parameter input bertipe Data seperti gambar di bawah ini:

```
20    public boolean insert(Data x){
21        //Lakukan Hashing dengan fungsi hash truncation
22        int hashValue= truncateFunction(x.key);
23
24        //Cek apakah tabel pada indeks hasil truncation sudah ada isi
25        //jika sudah berisi, maka return false
26        //karena tidak dapat menginsert data pada indeks tersebut
27        if(hashTable [hashValue]!=null) return false;
28
29        //jika kosong,
30        //tambahkan data pada indeks tersebut di dalam tabel.
31        //kembalikan nilai true
32        else{
33            hashTable[hashValue]=x;
34            return true;
35        }
36    }
```

e) Operasi lain yang diperlukan adalah untuk penghapusan data. Tambahkan method bernama delete dengan parameter input berupa key dari data yang akan dihapus.

```
38      public boolean delete(String key){
39          //Lakukan hashing (truncation) untuk key yang akan didelete
40          int hashValue=truncateFunction(key);
41
42          //Cek apakah tabel pada indeks hasil truncation kosong
43          //jika tidak kosong
44          //Cek apakah key dari data
45          //yang terdapat pada indeks tersebut cocok
46          //Jika iya, hapus data pada indeks tersebut
47          //kemudian return true
48          if(hashTable[hashValue]!=null){
49              if(hashTable[hashValue].key==key){
50                  hashTable[hashValue] = null;
51                  return true;
52              }
53              //jika key tidak cocok, return false
54              else return false;
55          }
56          //jika kosong, return false
57          else return false;
58      }
```

f) Selanjutnya, tambahkan method untuk mencari data berdasarkan key yang diinputkan.

```
60      public Data search(String key){
61          //Lakukan hashing (truncation) untuk key yang akan dicari
62          int hashValue=truncateFunction(key);
63
64          //Cek apakah tabel pada indeks hasil truncation kosong
65          //jika tidak kosong
66          //Cek apakah key dari data
67          //yang terdapat pada indeks tersebut cocok
68          //Jika iya, return data pada indeks tersebut
69          if (hashTable[hashValue]!=null){
70              if(hashTable[hashValue].key==key){
71                  return hashTable[hashValue];
72              }
73              //jika key tidak cocok, return null
74              else return null;
75          }
76          //jika kosong, return null
77          else return null;
78      }
79  }
```

g) Untuk menguji coba fungsi truncation, insert, delete dan search data, buatlah sebuah kelas lain bernama truncationTester. Pada contoh berikut ini, akan diinsertkan dua buah data dari dua NPM yang berbeda.

```
1  public class truncationTester{
2    public static void main(String[] args){
3       Data x= new Data("2013730075", "Tono Sungkono Mardiono");
4       Data y= new Data("2012730075", "Rani Maryani Hardiyani");
5       TruncationHash tes_truncate=new TruncationHash(150);
6       if(tes_truncate.insert(x)){
7          System.out.println("Sukses!");
8       };
9       if(!tes_truncate.insert(y)){
10         System.out.println("Gagal!");
11      };
12
13      Data temp=tes_truncate.search("2013730075");
14
15      if(temp!=null){
16          System.out.println("Ditemukan NPM "+ temp.key+ " dengan nama "+ temp.value);
17      }
18   }
19 }
```

2. Multiplicative Function

- Jika terdapat key-key berupa floating point pada rentang $0 \leq key < 1$, dan tabel hash berukuran M, maka cocok digunakan fungsi hash multiplicative: $h(k) = \lfloor k \times M \rfloor$.
  Misalnya terdapat key k=0.12567 dan akan dilakukan hashing terhadap key tersebut untuk tabel berukuran 10, maka nilai hash untuk k adalah $h(k) = \lfloor 0.12567 \times 10 \rfloor = \lfloor 1.2567 \rfloor = 1$.
- Jika key-key yang akan di-hash bukan merupakan floating point, maka fungsi hash multiplicative yang digunakan adalah $h(k) = \lfloor (kA \ mod \ 1) \times M \rfloor$. Dengan A adalah floating point dengan $0 \leq A < 1$. Nilai A yang direkomendasikan adalah A ≈ (√5 − 1)/2 = 0.6180339887…
  Contoh hashing dengan fungsi multplicative untuk A=0.137 dan M=1000:
  $h(61) = \lfloor 1000(61 \times 0.137 \ mod \ 1) \rfloor = \lfloor 1000(0.357) \rfloor = 357$
  $h(62) = \lfloor 1000(62 \times 0.137 \ mod \ 1) \rfloor = \lfloor 1000(0.494) \rfloor = 494$
  $h(63) = \lfloor 1000(63 \times 0.137 \ mod \ 1) \rfloor = \lfloor 1000(0.631) \rfloor = 631$
  $h(64) = \lfloor 1000(64 \times 0.137 \ mod \ 1) \rfloor = \lfloor 1000(0.768) \rfloor = 768$
  $h(65) = \lfloor 1000(65 \times 0.137 \ mod \ 1) \rfloor = \lfloor 1000(0.905) \rfloor = 905$