**PERTEMUAN KE 14**

**HASHING**

**1. Tujuan :**

Mahasiswa dapat mengimplementasikan penggunaan fungsi Hashing menggunakan Java Colection

**2. Teori :**

*Hashing* adalah transformasi aritmatik sebuah string dari karakter menjadi nilai yang merepresentasikan string aslinya. Menurut bahasanya, *hash* berarti memenggal dan kemudian menggabungkan. *Hashing* digunakan sebagai metode untuk menyimpan data dalam sebuah array agar penyimpanan data, pencarian data, penambahan data, dan penghapusan data dapat dilakukan dengan cepat. Ide dasarnya adalah menghitung posisi *record* yang dicari dalam array, bukan membandingkan *record* dengan isi pada array. Fungsi yang mengembalikan nilai atau kunci disebut fungsi *hash* (*hash* function) dan array yang digunakan disebut tabel *hash* (*hash* table). *Hash table* menggunakan struktur data *array* asosiatif yang mengasosiasikan *record* dengan sebuah *field* kunci unik berupa bilangan (*hash*) yang merupakan representasi dari *record* tersebut.

Ada beberapa metode untuk menemukan lokasi baru yang masih kosong. Dalam proses menemukan lokasi baru ini harus menggunakan pola tertentu agar *record* yang disimpan tetap bisa dicari dengan mudah saat dibutuhkan kemudian. Metode-metode yang sering digunakan adalah:

**1. *Linear probing***

Dengan menambahkan suatu interval pada hasil yang diperoleh dari fungsi *hash* sampai ditemukan lokasi yang belum terisi. Interval yang biasa digunakan adalah

**2. *Quadratic probing / squared probing***

Hampir sama dengan linear probing, hanya saja pada quadratic probing, hasil yang diperoleh dari fungsi *hash* ditambahkan dengan kuadrat dari interval yang digunakan.

**3. *Double hashing***

Pada metode *double hashing,* jika lokasi yang diperoleh dengan fungsi *hash* sudah terisi, maka dilakukan proses *hash* lagi sampai ditemukan lokasi yang belum terisi.

Java memiliki dua kelas yang mengimplementasikan interface Map, yaitu : TreeMap dan HashMap.

Secara umum, lebih baik menggunakan HashMap kecuali kita butuh struktur data dalam urutan tertentu yang hanya bisa dilakukan dengan TreeMap. Atau dengan kata lain, jika kita hanya menggunakan perintah put dan get, gunakan HashMap.

**HashMap** adalah kelas implementasi dari Map, Map itu sendiri adalah interface yang mempunyai fungsi untuk memetakan nilai dengan key unik. HashMap sangat bermanfaat sebagai memory record management, dimana tiap record dapat disimpan disebuah Map.

**TreeMap**  adalah salah satu implementasi dari class interface yang mengurutkan collection berdasarkan **key** dari elemen berupa pasangan **<key, value>**.

**Hash table** adalah salah satu (bukan satu-satunya) implementasi dari associative array, artinya hash table ini adalah concrete data type (sudah konkrit bukan lagi abstract/konseptual).

import java.util.\*;

public class HashtableDemo

{

public static void main(String args[])

{

Hashtable<Integer, String> ht= new Hashtable<Integer, String>();

ht.put(10, "Chaitanya");

ht.put(1, "Ajeet");

ht.put(11, "Test");

ht.put(9, "Demo");

ht.put(3, "Anuj");

Set set = ht.entrySet();

Iterator i = set.iterator();

while(i.hasNext()) {

Map.Entry me = (Map.Entry)i.next();

System.out.print(me.getKey() + ": ");

System.out.println(me.getValue());

}

}

}

import java.util.\*;

public class LinkedHashMapDemo

{

public static void main(String args[])

{

LinkedHashMap<Integer, String> lhm= new LinkedHashMap<Integer, String>();

lhm.put(10, "Chaitanya");

lhm.put(1, "Ajeet");

lhm.put(11, "Test");

lhm.put(9, "Demo");

lhm.put(3, "Anuj");

Set set = lhm.entrySet();

Iterator i = set.iterator();

while(i.hasNext()) {

Map.Entry me = (Map.Entry)i.next();

System.out.print(me.getKey() + ": ");

System.out.println(me.getValue());

}

}

}

import java.util.\*;

public class TreeMapDemo

{

public static void main(String args[])

{

TreeMap<Integer, String> tm= new TreeMap<Integer, String>();

tm.put(10, "Chaitanya");

tm.put(1, "Ajeet");

tm.put(11, "Test");

tm.put(9, "Demo");

tm.put(3, "Anuj");

Set set = tm.entrySet();

Iterator i = set.iterator();

while(i.hasNext()) {

Map.Entry me = (Map.Entry)i.next();

System.out.print(me.getKey() + ": ");

System.out.println(me.getValue());

}

}

}

**. Praktek :**

1. Implementasikan program di atas

**4. Latihan :**

Salinlah program berikut ini dan analisa hasil output dari program tersebut :

public class HashGram {

    public Long rollingHash(String sub) {

        long hash\_value = 0;

        int ascii;

        int prev\_hash = 0;

        int basis = 3;

        int c\_awal = 0;

        int length = sub.length() - 1;

        System.out.println("k = "+sub.length());

        System.out.println("b = "+basis);

        if (prev\_hash == 0) {

            for (int i = 0; i <= length; i++) {

                ascii = sub.charAt(i);

                hash\_value += (long) (ascii \* Math.pow(basis, length - i));

       System.out.println("c ke-"+(i+1)+" adalah huruf '"+sub.charAt(i)+"' bernilai ASCII = "+ascii);

            }

        } else {

          hash\_value = prev\_hash - (long) (c\_awal \* Math.pow(basis, length));

            hash\_value \*= basis;

            hash\_value += sub.charAt(length);

        }

        c\_awal = sub.charAt(0);

        prev\_hash = (int) hash\_value;

        System.out.println("Nilai hash pada kata = '"+sub+"' = "+hash\_value);

        return hash\_value;

    }

    public static void main(String[] args) {

        HashGram hg = new HashGram();

        String text = "STMIK Akakom";

        hg.rollingHash(text);

    }

}

**5. Tugas**

Diberikan soal oleh dosen pengampu