BAB 1

ANALISIS DAN PERANCANGAN

Pada bab ini, penulis akan menjelaskan apa saja yang dilakukan dalam pengembangan *Agglomerative Clustering* untuk Spark. Pengembangan dilakukan untuk mencapai tujuan yaitu mendapatkan pola dari dataset yang diolah. Pola yang ingin didapatkan meliputi perhitungan rata-rata, nilai maksimum, nilai minimum dan nilai standar deviasi dari setiap attribut yang ada pada data. Selain itu, perlu didapatkan juga jumlah anggota pada setiap *cluster* yang dihasilkan dari algoritma *Agglomerative Hierarchical Clustering*.

1.1 Analisis Perangkat Lunak

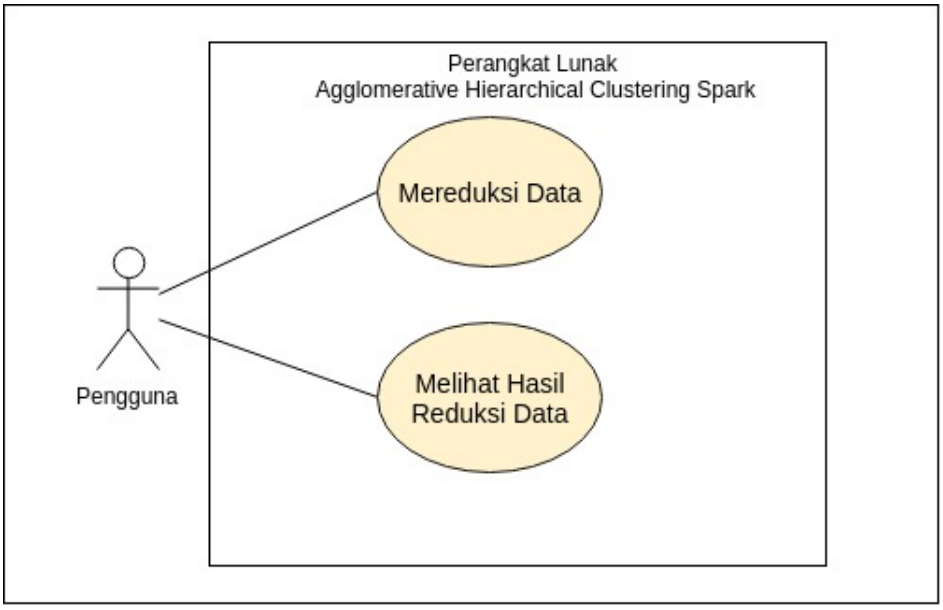
asdasd

1.2 Perancangan Perangkat Lunak

Pada bagian ini, akan dijelaskan perancangan perangkat lunak. Perancangan termasuk diagram *use case*, skenario, diagram kelas, dan rancangan *user interface*.

1.2.1 Diagram Use Case

Diagram *use case* merupakan sebuah pemodelan untuk perilaku dari perangkat lunak yang akan dibuat. Diagram *use case* digunakan untuk mengetahui fungsi apa saja yang ada dalam perangkat lunak. Fungsi-fungsi dari perangkat lunak akan dioperasikan oleh satu pengguna. Cara kerja dan perilaku dari perangkat lunak akan dijelaskan dalam bentuk diagram *use case*. Diagram *use case* dapat dilihat pada Gambar 1.1.



Gambar 1.1: Gambar diagram *use case* perangkat lunak *Agglomerative Hierarchical Clustering*

Bedasarkan gambar diagram *use case* diatas, berikut adalah skenario yang ada:

1. Nama *use case*: Mereduksi data

   - Aktor: Pengguna
   - Pre-kondisi: data yang akan diolah dimasukan kepada HDFS
   - Pra -kondisi: hasil reduksi disimpan pada HDFS
   - Deskripsi: Fitur untuk menjalankan program untuk mereduksi data
   - Langkah-langkah:
     (a) Pengguna mengisi JAR *path*, *input path*, dan *output path*
     (b) Pengguna mengisi jumlah *executor* dan besar *executor memory*
     (c) Pengguna mengisi jumlah partisi, batas maksimum objek, tipe metode, dan *cut-off distance*
     (d) Pengguna menekan tombol *submit*
     (e) Sitem melakukan pengolahan data dengan algoritma *Agglomerative Hierarchical Clustering* pada *cluster* Hadoop
     (f) Sistem membuka tab baru untuk melihat tahap dan progres program
     (g) Sistem menyimpan hasil reduksi pada HDFS

2. Nama *use case*: Melihat data

   - Aktor: Pengguna
   - Pre-kondisi: data yang akan ditampilkan sudah disimpan pada HDFS
   - Pra -kondisi: menampilkan data yang ada pada HDFS
   - Deskripsi: fitur untuk melihat data hasil reduksi
   - Langkah-langkah:
     (a) Pengguna mengisi *path* dimana data disimpan pada HDFS
     (b) Sistem menampilkan data-data pada direktori tersebut

antarmuka

DAFTAR REFERENSI

LAMPIRAN A

KODE PROGRAM

Listing A.1: MyCode.c

```c

// This does not make algorithmic sense,
// but it shows off significant programming characters.

#include<stdio.h>

void myFunction( int input, float* output ) {
    switch ( array[i] ) {
        case 1: // This is silly code
            if ( a >= 0 || b <= 3 && c != x )
                *output += 0.005 + 20050;
            char = 'g';
            b = 2^n + ~right_size - leftSize * MAX_SIZE;
            c = (--aaa + &daa) / (bbb++ - ccc % 2 );
            strcpy(a,"hello $@?");
    }
    count = ~mask | 0x00FF00AA;
}

// Fonts for Displaying Program Code in LATEX
// Adrian P. Robson, nepsweb.co.uk
// 8 October 2012
// http://nepsweb.co.uk/docs/progfonts.pdf
```

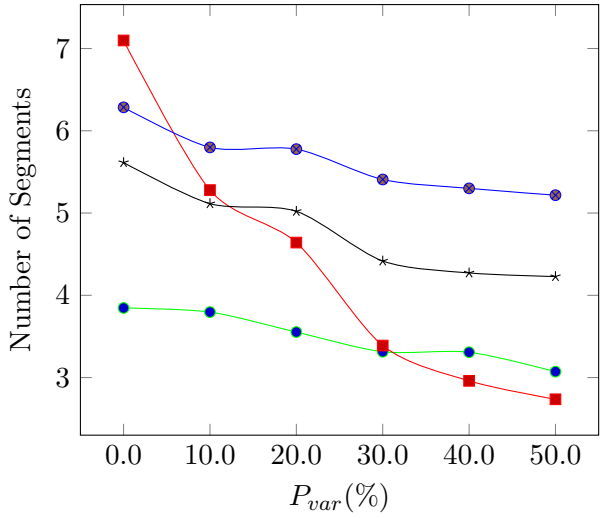
Listing A.2: MyCode.java

```java
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashSet;

//class for set of vertices close to furthest edge
public class MyFurSet {
    protected int id;                                   //id of the set
    protected MyEdge FurthestEdge;                      //the furthest edge
    protected HashSet<MyVertex> set;                    //set of vertices close to furthest edge
    protected ArrayList<ArrayList<Integer>> ordered;    //list of all vertices in the set for each trajectory
    protected ArrayList<Integer> closeID;               //store the ID of all vertices
    protected ArrayList<Double> closeDist;              //store the distance of all vertices
    protected int totaltrj;                             //total trajectories in the set

    /*
     * Constructor
     * @param id : id of the set
     * @param totaltrj : total number of trajectories in the set
     * @param FurthestEdge : the furthest edge
     */
    public MyFurSet(int id,int totaltrj,MyEdge FurthestEdge) {
        this.id = id;
        this.totaltrj = totaltrj;
        this.FurthestEdge = FurthestEdge;
        set = new HashSet<MyVertex>();
        ordered = new ArrayList<ArrayList<Integer>>();
        for (int i=0;i<totaltrj;i++) ordered.add(new ArrayList<Integer>());
        closeID = new ArrayList<Integer>(totaltrj);
        closeDist = new ArrayList<Double>(totaltrj);
        for (int i = 0;i <totaltrj;i++) {
            closeID.add(-1);
            closeDist.add(Double.MAX_VALUE);
        }
    }

}
```

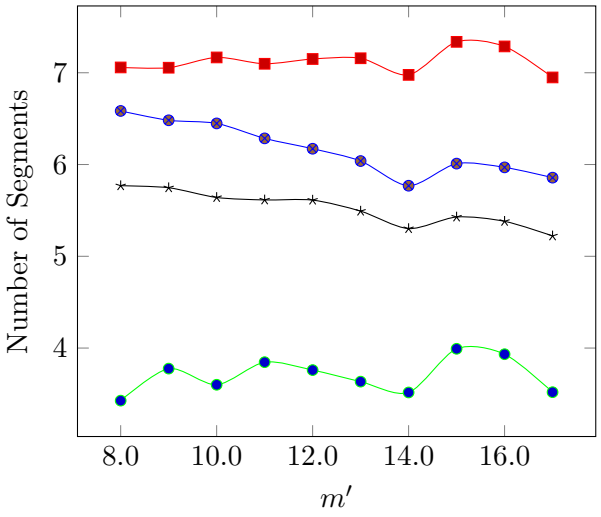
LAMPIRAN B

HASIL EKSPERIMEN

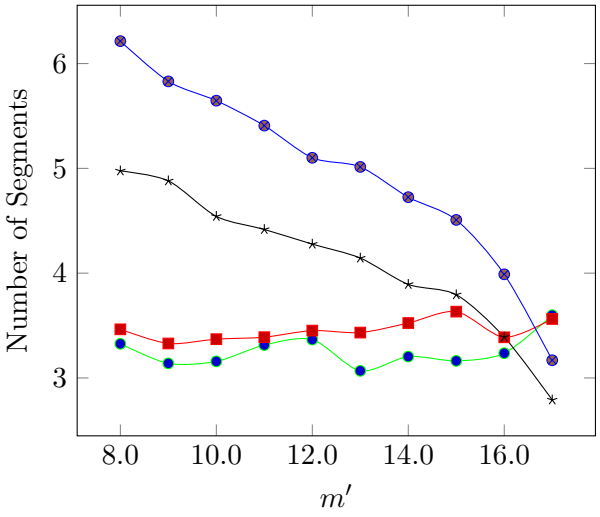
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



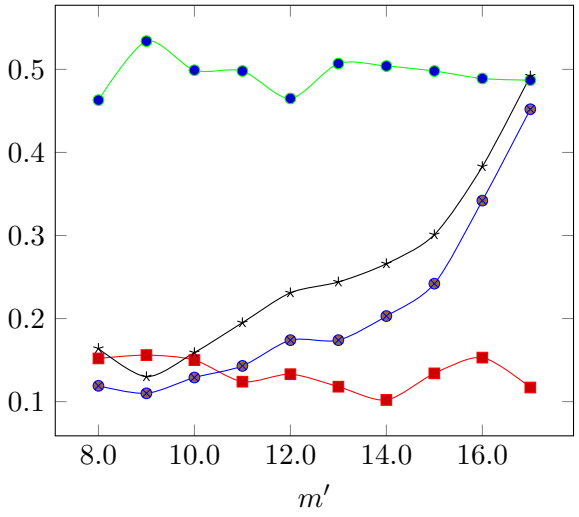
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4