SKRIPSI

PEREKAMAN KEHADIRAN DARING OTOMATIS

Reinalta Sugianto

NPM: 2017730035

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN SAINS
UNIVERSITAS KATOLIK PARAHYANGAN
2022

DAFTAR ISI

DAFTAR GAMBAR

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Perkuliahan di UNPAR biasanya membutuhkan perekaman kehadiran untuk mengetahui kehadiran mahasiswa dan dosen, bagi mahasiswa UNPAR perekaman kehadiran biasanya dilakukan dengan melakukan tanda tangan pada daftar kehadiran atau dicatat langsung oleh dosen yang memanggil mahasiswanya, sedangkan bagi dosen UNPAR perekaman kehadiran dilakukan dengan menggunakan *fingerprint*. Perekaman kehadiran diperkirakan membutuhkan waktu sekitar kurang dari 5 detik.

Pada tahun 2020 terjadi pandemi Covid-19 di seluruh negara. Pandemi Covid-19 masuk ke Indonesia pada awal bulan Maret tahun 2020. Covid-19 adalah penyakit yang disebabkan oleh virus *severe acute respiratory syndrome coronavirus 2* (SARS-CoV-2) [1]. Penularan virus Covid-19 terjadi saat seseorang menyentuh barang yang sudah terkontaminasi oleh droplet orang yang terkena virus Covid-19 atau terkena droplet orang lain saat berinteraksi langsung dengan orang yang terkena virus Covid-19. Akibat pandemi Covid-19 yang dapat menular ini, maka hampir seluruh kegiatan di Indonesia dilakukan secara daring untuk mengurangi interaksi orang secara langsung yang dapat meningkatkan angka penularan virus tersebut.

Pembelajaran secara daring diberlakukan oleh UNPAR di akhir bulan Maret untuk seluruh kegiatan perkuliahan demi mencegah penularan virus Covid-19. Akibat diberlakukannya pembelajaran secara daring, maka perekaman kehadiran di UNPAR dilakukan dengan menggunakan aplikasi atau situs web milik UNPAR. Cara perekaman kehadiran secara daring di UNPAR ini mumbutuhkan waktu lebih agar dapat tercatat perekaman kehadirannya, karena butuh waktu untuk membuka situs web serta perlu memasukan *email* dan *password* hingga akhirnya melakukan perekaman kehadiran.

Selenium adalah *open-source framework* pengujian otomatisasi untuk aplikasi web[1]. WebDriver menggunakan API otomatisasi *browser* yang disediakan oleh vendor *browser* untuk mengontrol *browser* dan melakukan pengujian. API WebDriver ini seolah-olah membuat pengguna secara langsung mengoperasi *browser*, padahal dijalankan secara otomatis langsung oleh *API* WebDriver tersebut. Selenium WebDriver adalah sebuah *tools* yang berguna untuk melakukan otomatisasi terhadap web pada *browser*. Selenium WebDriver ini tersedia untuk bahasa pemrograman Ruby, Java, Python, C#, dan JavaScript. Pembuatan Perekaman kehadiran daring otomatis ini akan menggunakan Selenium WebDriver dengan bahasa pemrograman Python.

Pada skripsi ini, akan dibuat sebuah perangkat lunak yang dapat melakukan perekaman kehadiran otomatis dengan sistem menerima rangsangan satu “klik” sehingga dapat melakukan

[1] *Pandemi Covid-19 di Indonesia* https://id.wikipedia.org/wiki/Pandemi_Covid-19_di_Indonesia

hal-hal berikut :

1. Membuka peramban.
2. Membuka situs web perekaman kehadiran.
3. Mengisi dan *login* dengan *username* serta *password* yang ddiambil dari file konfigurasi.
4. Melakukan rekam kehadiran.

perangkat lunak ini bertujuan agar mahasiswa dan dosen dapat melakukan perekaman kehadiran secara online dengan lebih mudah serta mengurangi waktu yang dibutuhkan untuk berinteraksi dengan aplikasi atau situs web dan bukan untuk mempercepat waktu agar kehadiran terekam, sehingga membuat waktu perekaman kehadiran secara daring dapat menyamai waktu perekaman kehadiran secara luring.

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas di skripsi ini adalah sebagai berikut :

- Bagaimana cara membangun program Perekaman Kehadiran Daring Otomatis?
- Bagaimana cara mengurangi waktu interaksi dengan aplikasi atau situs web untuk merekam kehadiran?

1.3 Tujuan

Tujuan yang ingin dicapai dari penulisan skripsi ini sebagai berikut :

- Membangun program menggunakan Selenium WebDriver.
- Membuat program yang mampu menerima rangsangan satu tombol untuk melakukan beberapa hal menggunakan Selenium.

1.4 Batasan Masalah

Beberapa batasan yang dibuat terkait dengan pengerjaan skripsi ini adalah sebagai berikut :

1. Program ini bukan untuk mempercepat kehadiran terekam, hanya untuk mengurangi waktu untuk berinteraksi dengan aplikasi.

1.5 Metodologi

Metodologi yang dilakukan pada skripsi ini adalah sebagai berikut :

1. Melakukan studi mengenai Selenium WebDriver.
2. Mempelajari bahasa pemrograman python.
3. Mempelajari cara menggunakan Selenium.
4. Menganalisis web Student Portal UNPAR.
5. Membangun program perekaman kehadiran daring otomatis.
6. Melakukan pengujian dan eksperimen.
7. Menulis dokumen skripsi.

1.6 Sistematika Pembahasan

Sistematika penulisan setiap bab skripsi ini adalah sebagai berikut :

1. Bab 1 Pendahuluan
   Bab ini berisi latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi, dan sistematika pembahasan yang digunakan untuk menyusun skripsi ini.
2. Bab 2 Dasar Teori
   Bab ini berisi teori-teori yang digunakan dalam pembuatan skripsi ini. Teori yang digunakan yaitu Selenium, Portal Akademik Mahasiwa, dan Sistem Perekam Kehadiran.
3. Bab 3 Analisis Masalah
   Bab ini berisi analisis yang digunakan pada skripsi ini, analisa kebutuhan program Perekaman Kehadiran Online dan analisis Portal Akademik Mahasiswa.
4. Bab 4 Perancangan
   Bab ini berisi perancangan program,
5. Bab 5 Implementasi dan Pengujian
   Bab ini berisi implementasi dan pengujian program, meliputi lingkungan implementasi, hasil implementasi, pengujian fungsional, dan pengujian eksperimental.
6. Bab 6 Kesimpulan dan Saran
   Bab ini berisi kesimpulan dari hasil pembangunan program beserta saran untuk pengembangan selanjutnya.

BAB 2

LANDASAN TEORI

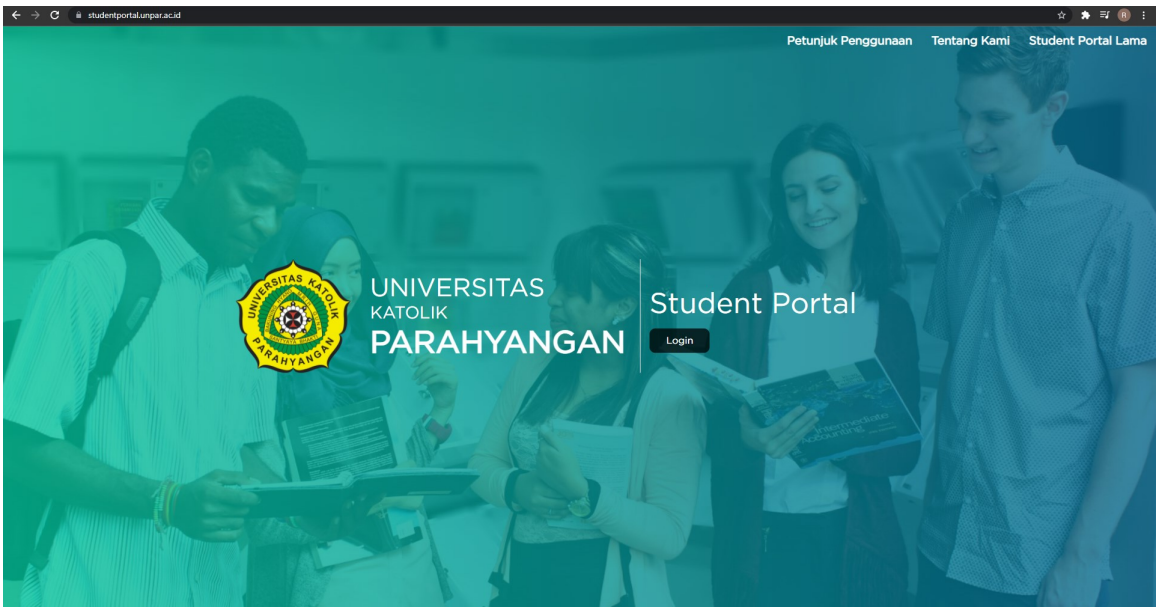
Pada bab ini dijelaskan dasar teori mengenai Selenium, Portal Akademik Mahasiswa, dan Sistem Perekam Kehadiran.

2.1 Portal Akademik Mahasiswa 2018

Portal Akademik Mahasiswa (selanjutnya disingkat dengan PAM) adalah sebuah *web* yang di peruntukan bagi mahasiswa dalam rangka mendapatkan informasi kegiatan akademik mulai dari registrasi, melihat jadwal kuliah dan ujian, info nilai sampai pendaftaran sidang[2]. Portal Akademik Mahasiswa dapat diakses melalui https://www.studentportal.unpar.ac.id/. Pada masa pandemi Covid-19 ini Portal Akademik Mahasiswa UNPAR sudah dapat melakukan perekaman kehadiran secara online melalui *web* Portal Akademik Mahasiswa. Mahasiswa harus *login* dengan *email* dan *passowrd* agar bisa melakukan perekaman kehadiran online.

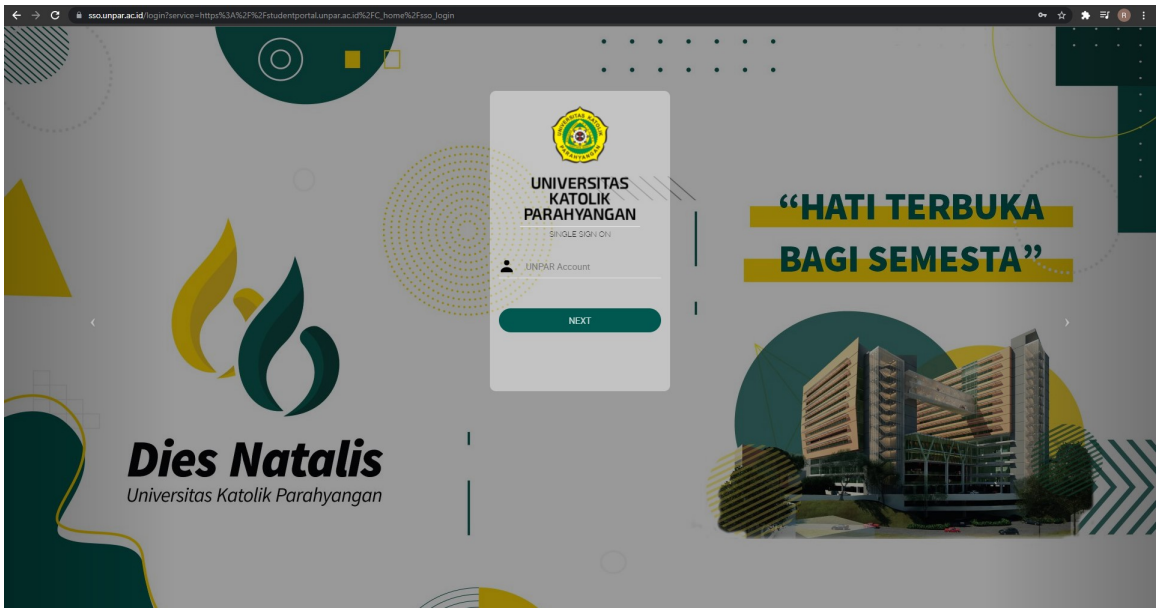
Panduan untuk melakukan perekaman kehadiran online di Portal Akademik Mahasiswa 2018 sebagai berikut:

1. Masuk ke *web* https://www.studentportal.unpar.ac.id/ (Gambar 2.1). Lalu klik tombol *Login*.



Gambar 2.1: Tampilan halaman awal Portal Akademik Mahasiswa

2. *Login* dengan memasukan *email* mahasiswa unpar dan *password* 2.2).

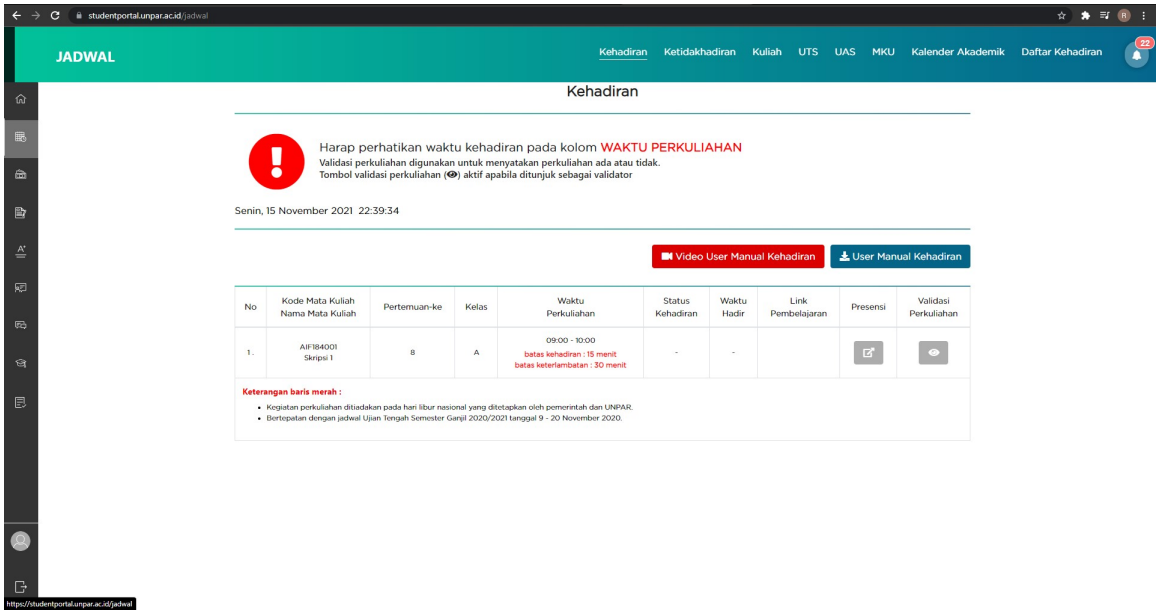


Gambar 2.2: Tampilan halaman untuk melakukan *Login*

3. Setelah *login* berhasil dilakukan, maka akan muncul halaman utama (Gambar 2.3). Lalu klik pada heksagon berlabel 'Jadwal & Kehadiran' untuk melakukan perekaman kehadiran online.

Gambar 2.3: Tampilan halaman utama Portal Akademik Mahasiswa

4. Browser akan menampilkan halaman utama pada fitur 'Jadwal & Kehadiran' (Gambar 2.4). Mahasiswa dapat melihat jadwal kuliah untuk besok hari. Lalu terdapat tombol merah pada kolom presensi untuk melakukan perekaman kehadiran.



Gambar 2.4: Tampilan halaman untuk melakukan perekam kehadiran online

2.2 Selenium

Selenium adalah *open-source framework* pengujian otomatisasi untuk aplikasi web[1]. WebDriver menggunakan API otomatisasi *browser* yang disediakan oleh vendor *browser* untuk mengontrol *browser* dan melakukan pengujian. API WebDriver ini seolah-olah membuat pengguna secara langsung mengoperasi *browser*, padahal dijalankan secara otomatis langsung oleh *API* WebDriver tersebut. Selenium WebDriver adalah sebuah *tools* yang berguna untuk melakukan otomatisasi terhadap web pada *browser*. Selenium WebDriver ini tersedia untuk bahasa pemrograman Ruby, Java, Python, C#, dan JavaScript.

2.2.1 Navigating

Hal pertama untuk menggunakan WebDriver adalah menavigasi ke *link*. Cara normal untuk melakukan navigasi adalah dengan *method get*().

2.2.2 Locating Elements

Pada selenium ada berbagai cara untuk menemukan elemen di halaman. Selenium menyediakan berbagai metode menemukan elemen yang dapat pilih untuk menyelesaikan kasus tertentu, berikut berbagai metode untuk menemukan elemen:

1. find_element_by_id: untuk mencari elemen berdasarkan *id* atribut.
2. find_element_by_name: untuk mencari elemen berdasarkan nama atribut.
3. find_element_by_xpath: untuk mencari elemen berdasarkan *xpath*. *XPath* adalah bahasa yang digunakan untuk menemukan node dalam dokumen XML.

4. find_element_by_link_text: digunakan untuk mengetahui *link* teks yang digunakan dalam *anchor* tag. Cara ini, elemen pertama dengan *link* teks lengkap yang cocok dengan nilai yang diberikan akan dikembalikan.
5. find_element_by_partial_link_text: digunakan untuk mengetahui *link* teks yang digunakan dalam *anchor* tag. Cara ini, elemen pertama dengan *link* teks sebagian yang cocok dengan nilai yang diberikan akan dikembalikan.
6. find_element_by_tag_name: untuk mencari elemen berdasarkan nama tag.
7. find_element_by_class_name: untuk mencari elemen berdasarkan nama kelas
8. find_element_by_css_selector: untuk mencari elemen dengan menggunakan sintaks *CSS selector*

Untuk menemukan beberapa elemen secara langsung, dengan cara berikut:

1. find_elements_by_id
2. find_elements_by_name
3. find_elements_by_xpat
4. find_elements_by_link_text
5. find_elements_by_partial_link_text
6. find_elements_by_tag_name
7. find_elements_by_class_name
8. find_elements_by_css_selector

2.2.3 Waits

Selenium WebDriver menyediakan dua jenis *waits* adalah implisit dan eksplisit. *Waits* eksplisit ini membuat WebDriver menunggu kondisi tertentu terjadi sebelum melanjutkan eksekusi. *Waits* implisit membuat WebDriver melakukan polling DOM untuk jangka waktu tertentu saat mencoba menemukan elemen.

2.2.4 WebDriver API

WebDriver API ini mencakup semua *interface* Selenium WebDriver. Selenium WebDriver dapat digunakan dengan melakukan import webdriver. API pada selenium ini untuk menunjukan lokasi kelas yang absolut, misalkan untuk Firefox, Chrome, Opera, Ie, dan lain-lainnya.

2.3 Sistem Perekam Kehadiran

DAFTAR REFERENSI

[1] 123 (2001) *Selenium*. Software Freedom Conservancy All. qweqwe.

[2] 2018, T. P. P. A. M. P. (2018) Portal akademik mahasiswa. https://studentportal.unpar.ac.id/assets/BUKU_PANDUAN_PENGGUNAAN_FRS_GABUNGAN.pdf. Online; diakses 15-November-2021.

LAMPIRAN A

KODE PROGRAM

Kode 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
```

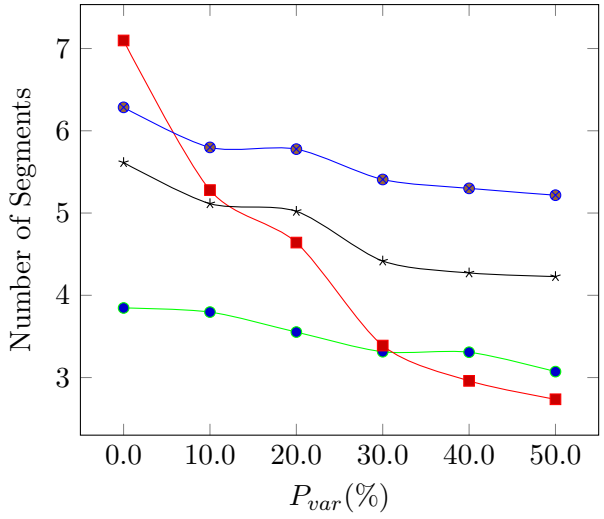
Kode A.2: MyCode.java

```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);
        }
    }

}
```

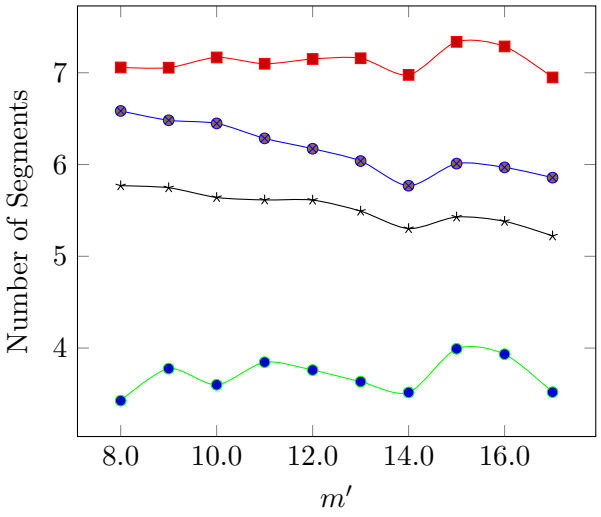
LAMPIRAN B

HASIL EKSPERIMEN

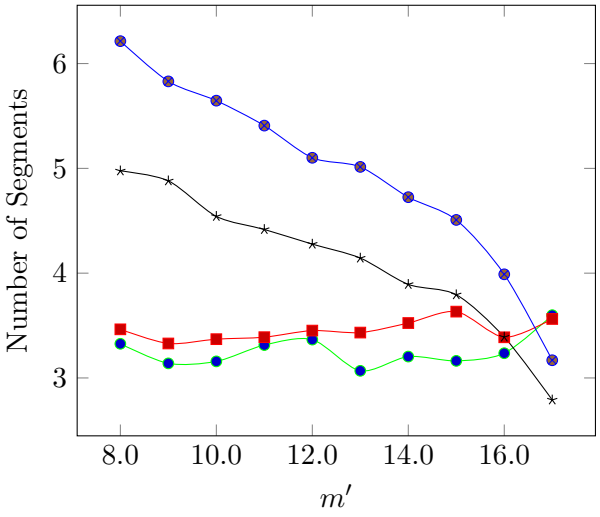
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



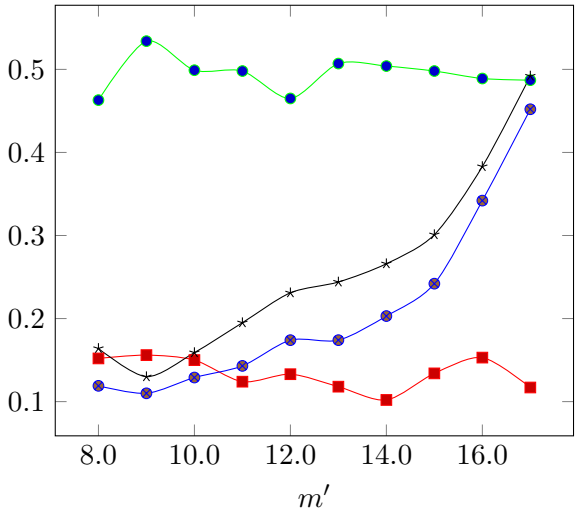
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4