BAB 1

PENDAHULUAN

1.1 Latar Belakang

SharIF Judge merupakan perangkat lunak berbasis web yang dapat digunakan untuk menilai kode program dalam bahasa C, C++, Java, dan Python. SharIF Judge yang dibahas pada dokumen ini adalah *fork* dari Sharif Judge yang dibuat oleh Mohammad Javad Naderi. Versi *fork* ini sudah dikembangkan sesuai kebutuhan jurusan Teknik Informatika UNPAR dalam proses penilaian di beberapa mata kuliah.

PHP Standards Recommendations (PSR) adalah kumpulan standar penulisan PHP yang dibuat oleh PHP Framework Interop Group. Pada saat skripsi ini dibuat, terdapat 14 bab yang sudah diterima (Accepted) untuk digunakan, 4 bab masih didiskusikan (Draft), 3 bab ditinggalkan (Abandoned), dan 2 bab sudah usang (Deprecated). Bab-bab standar yang sudah diterima dan digunakan antara lain:

1. Basic Coding Standard
2. Logger Interface
3. Autoloading Standard
4. Caching Interface
5. HTTP Message Interface
6. Container Interface
7. Extended Coding Style Guide
8. Hypermedia Links
9. Event Dispatcher
10. HTTP Handlers
11. Simple Cache
12. HTTP Factories
13. HTTP Client
14. Clock

Pada skripsi ini, keseluruhan PHP pada SharIF Judge akan dilihat dan dievaluasi seberapa jauh standar PSR yang sudah dipenuhi. Selanjutnya akan dibuat rekomendasi berdasarkan hasil evaluasi. Walaupun demikian, masih akan ditentukan strategi untuk melakukannya, misalnya menggunakan tools atau alat tertentu untuk membantu, bab-bab apa saja yang relevan untuk dievaluasi sesuai yang digunakan pada SharIF Judge, dan seberapa banyak bab yang harus diperiksa secara manual. Salah satu alat yang digunakan adalah PHP linter. PHP linter membantu proses pemeriksaan sesuai salah satu standar, yaitu Extended Coding Style Guide atau aturan penulisan PHP yang sudah diperbarui.

Perangkat lunak akhir yang akan dibuat memiliki fitur yang sama persis dengan SharIf Judge yang sudah ada. Perbedaannya terdapat pada struktur PHP yang sudah dievaluasi sehingga memenuhi PSR. Berikut adalah fitur-fitur dari dokumentasi SharIF Judge:

- Dapat diakses untuk empat role : *admin*, *head instructor*, *instructor*, dan *student*
- Dapat mendeteksi plagiarisme pada kode
- Pengaturan khusus untuk keterlambatan pengumpulan

- Menunjukkan antrean pengumpulan
- Hasil penilaian dapat diekspor dalam dokumen Excel
- Dapat melakukan penilaian ulang
- Terdapat Scoreboard dan Notifications
- Tersedia log untuk 24 jam

1.2 Rumusan Masalah

Rumusan masalah yang akan dibahas pada skripsi ini sebagai berikut:

- Seberapa jauh PSR yang sudah terpenuhi ada SharIF Judge?
- Bagaimana mengevaluasi kode PHP pada SharIF Judge sesuai PSR?
- Bagaimana memberikan rekomendasi perbaikan pada kode PHP SharIF Judge agar meningkatkan jumlah PSR yang terpenuhi?

1.3 Tujuan

Tujuan yang ingin dicapai dalam penulisan skripsi ini sebagai berikut:

- Mengetahui seberapa jauh PSR yang sudah terpenuhi pada SharIF Judge.
- Mengevaluasi kode PHP pada SharIF Judge sesuai PSR.
- Memberikan rekomendasi perbaikan pada kode PHP SharIF Judge agar meningkatkan jumlah PSR yang terpenuhi.

1.4 Batasan Masalah

1.5 Metodologi

Metode penelitian yang akan digunakan dalam skripsi ini adalah:

1. Memperlajari SharIF Judge saat ini
2. Melakukan studi literatur mengenai PSR dan PHP linter
3. Mengevaluasi PHP dari SharIF Judge sesuai dengan PSR
4. Menguji SharIF Judge yang sudah dievaluasi
5. Memberikan rekomendasi sesuai hasil evaluasi
6. Menulis dokumen skripsi

1.6 Sistematika Pembahasan

Untuk penulisan skripsi ini akan dibagi dalam lima bagian sebagai berikut:

Bab 1 Pendahuluan
Bab 2 Landasan Teori
Bab 3 Analisis
Bab 4 Perancangan
Bab 5 Implementasi
Bab 6 Kesimpulan

BAB 2

LANDASAN TEORI

2.1 SharIF Judge

SharIF Judge (dengan huruf kapital "IF") merupakan perangkat lunak berbasis web yang digunakan untuk menilai kode program dalam bahasa C, C++, Java, dan Python. SharIF Judge yang dibahas dalam dokumen ini adalah versi fork dari Sharif Judge (dengan huruf kecil "if") yang dibuat oleh Mohammad Javad Naderi. Versi fork ini dikembangkan sesuai dengan kebutuhan jurusan Teknik Informatika UNPAR dalam proses penilaian di beberapa mata kuliah. SharIF judge dibuat dengan PHP pada framework CodeIgniter dan BASH untuk backend.

2.1.1 Fitur

Berikut adalah fitur-fitur dari SharIF Judge.

1. Terdapat beberapa role pengguna, antara lain admin, head instructor, instructor, dan student.
2. Sandboxing (belum tersedia untuk Python)
3. Deteksi kecurangan (mendeteksi kemiripan kode) menggunakan Moss
4. Pengaturan untuk keterlambatan pengumpulan
5. Antrian pengiriman
6. Mengunduh hasil dalam bentuk file excel
7. Mengunduh kode yang dikirim dalam bentuk file zip
8. Metode "Output Comporison" dan "Tester Code" untuk memeriksa kebenaran dari hasil output
9. Menambahkan beberapa pengguna sekaligus
10. Deskripsi masalah (PDF/Markdown/HTML)
11. Penilaian ulang (rejudge)
12. Papan Nilai (Scoreboard) dan Notifikasi (Notifications)

2.1.2 Instalasi

Untuk menjalankan SharIF Judge, dibutuhkan sebuah server Linux dengan syarat sebagai berikut:

- Webserver menjalankan PHP versi 5.3 atau lebih baru
- Pengguna dapat menjalankan PHP melalui command line.
- Menggunakan database MySql atau PostgreSql.
- PHP harus diberikan akses untuk untuk menjalankan perintah menggunakan fungsi shell$_exec.Toolsyangdig$
- Perl lebih baik diinstal untuk alasan ketepatan waktu, batas memori dan memaksimalkan batas ukuran pada output kode yang dikirim.

2.2 PHP Standards Recommendations

PHP Standards Recommendations (PSR) adalah kumpulan rekomendasi yang dibuat oleh PHP Framework Interop Group (PHP-FIG) untuk membantu para pengembang PHP dalam menciptakan kode yang lebih mudah dibaca, dipahami, dan dipelihara. Setiap PSR memiliki status, antara lain

Accepted, Draft, Abandoned, dan Deprecated. Saat dokumen ini dibuat, terdapat 23 bab PSR dengan rincian sebagai berikut.

- Accepted
    1. (1) Basic Coding Standard
    2. (3) Logger Interface
    3. (4) Autoloading Standard
    4. (6) Caching Interface
    5. (7) HTTP Message Interface
    6. (11) Container Interface
    7. (12) Extended Coding Style Guide
    8. (13) Hypermedia Links
    9. (14) Event Dispatcher
    10. (15) HTTP Handlers
    11. (16) Simple Cache
    12. (17) HTTP Factories
    13. (18) HTTP Client
    14. (20) Clock
- Draft
    1. (5) PHPDoc Standard
    2. (19) PHPDoc Tags
    3. (21) Internationalization
    4. (22) Application Tracing
- Abandoned
    1. (8) Huggable Interface
    2. (9) Security Advisories
    3. (10) Security Reporting Process
- Deprecated
    1. (0) Autoloading Standard
    2. (2) Coding Style Guide

2.3 PHP Linter

Lint awalnya merujuk pada tool yang digunakan untuk menganalisis suatu kode program dengan tujuan menemukan kesalahan pada bahasa C. Kemudian istilah ini menjadi sebutan untuk mendeskripsikan hal-hal yang berkaitan dengan pengecekan kode program. PHP linter adalah tool yang digunakan untuk menganalisis kode PHP sesuai dengan standar tertentu. PHP linter yang digunakan adalah yang dibuat oleh Brueggern. Linter ini berdasar pada standar PSR ke-2 dan ke-12, yaitu Coding Style Guide yang sudah usang (deprecated) dan Extended Coding Style Guide sebagai penggantinya (accepted).

2.3.1 Syarat Instalasi

Sebelum menginstal linter, perlu dilakukan penginstalan Composer terlebih dahulu. Composer adalah alat untuk mengelola dependency pada PHP.

2.3.2 Instalasi

Berikut adalah langkah instalasi PHP linter.

1. Pada bagian root project, buka file composer.json.
2. Pada bagian "Repositories", tambahkan kode berikut.

Kode 2.1: kode kode

```
{
    "repositories": [
    {
        "type": "vcs",
        "url": "git@github.com:brueggern/php-linter.git"
    }
    ]
}
```

3. Install composer package. `composer require brueggern/php-linter`
4. Tambahkan script berikut untuk menjalankan linting/fixing. "app" dapat diganti dengan nama file atau folder yang akan di-lint.

Kode 2.2: kode kode

```
{
    "scripts": {
        "lint": "php-linter␣app",
        "lint:fix": "php-linter␣--fix␣app"
    }
}
```

2.3.3 Penggunaan

Untuk menjalankan linter perintah yang digunakan adalah: `composer run lint`

Untuk memperbaiki error secara otomatis, perintah yang digunakan adalah: `composer run lint:fix`

2.3.4 Tabel

Berikut adalah contoh pembuatan tabel. Penempatan tabel dan gambar secara umum diatur secara otomatis oleh LaTeX, perhatikan contoh di file bab2.tex untuk melihat bagaimana cara memaksa tabel ditempatkan sesuai keinginan kita.

Perhatikan bawa berbeda dengan penempatan judul gambar gambar, keterangan tabel harus diletakkan di atas tabel!! Lihat Tabel 2.1 berikut ini:

Tabel 2.1: Tabel contoh

	v_{start}	$\mathcal{S}_1$	v_{end}
τ_1	1	12	20
τ_2	1		20
τ_3	1	9	20
τ_4	1		20

Tabel 2.2 dan Tabel 2.3 berikut ini adalah tabel dengan sel yang berwarna dan ada dua tabel yang bersebelahan.

Tabel 2.2: Tabel bewarna(1)

	v_{start}	$\mathcal{S}_2$	$\mathcal{S}_1$	v_{end}
τ_1	1	5	12	20
τ_2	1	8		20
τ_3	1	2/8/17	9	20
τ_4	1			20

Tabel 2.3: Tabel bewarna(2)

	v_{start}	$\mathcal{S}_1$	$\mathcal{S}_2$	v_{end}
τ_1	1	12	5	20
τ_2	1		8	20
τ_3	1	9	2/8/17	20
τ_4	1			20

2.3.5 Kutipan

Berikut contoh kutipan dari berbagai sumber, untuk keterangan lebih lengkap, silahkan membaca file referensi.bib yang disediakan juga di template ini. Contoh kutipan:

- Buku: [1]
- Bab dalam buku: [2]
- Artikel dari Jurnal: [3]
- Artikel dari prosiding seminar/konferensi: [4]
- Skripsi/Thesis/Disertasi: [5] [6] [7]
- Technical/Scientific Report: [8]
- RFC (Request For Comments): [9]
- Technical Documentation/Technical Manual: [10] [11] [12]
- Paten: [13]
- Tidak dipublikasikan: [14] [15]
- Laman web: [16]
- Lain-lain: [17]

2.3.6 Gambar

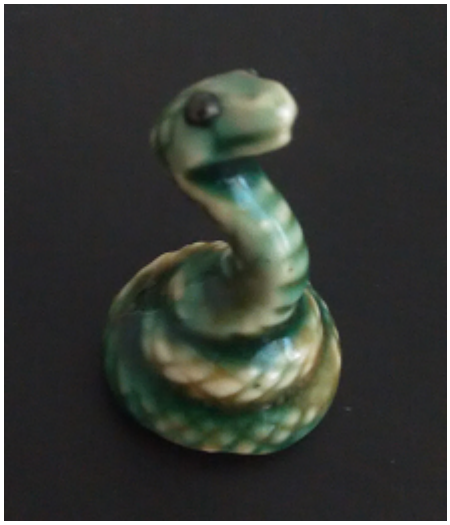
Pada hampir semua editor, penempatan gambar di dalam dokumen LaTeX tidak dapat dilakukan melalui proses *drag and drop*. Perhatikan contoh pada file bab2.tex untuk melihat bagaimana cara menempatkan gambar. Beberapa hal yang harus diperhatikan pada saat menempatkan gambar:

- Setiap gambar **harus** diacu di dalam teks (gunakan *field* LABEL)
- *Field* CAPTION digunakan untuk teks pengantar pada gambar. Terdapat dua bagian yaitu yang ada di antara tanda [dan] dan yang ada di antara tanda { dan }. Yang pertama akan muncul di Daftar Gambar, sedangkan yang kedua akan muncul di teks pengantar gambar. Untuk skripsi ini, samakan isi keduanya.
- Jenis file yang dapat digunakan sebagai gambar cukup banyak, tetapi yang paling populer adalah tipe PNG (lihat Gambar 2.1), tipe JPG (Gambar 2.2) dan tipe PDF (Gambar 2.3)
- Besarnya gambar dapat diatur dengan *field* SCALE.
- Penempatan gambar diatur menggunakan *placement specifier* (di antara tanda [dan] setelah deklarasi gambar. Yang umum digunakan adalah **H** untuk menempatkan gambar **sesuai** penempatannya di file .tex atau **h** yang berarti "kira-kira" di sini.
  Jika tidak menggunakan *placement specifier*, LaTeX akan menempatkan gambar secara otomatis untuk menghindari bagian kosong pada dokumen anda. Walaupun cara ini sangat mudah, hindarkan terjadinya penempatan dua gambar secara berurutan.
  - Gambar 2.1 ditempatkan di bagian atas halaman, walaupun penempatannya dilakukan setelah penulisan 3 paragraf setelah penjelasan ini.
  - Gambar 2.2 dengan skala 0.5 ditempatkan di antara dua buah paragraf. Perhatikan penulisannya di dalam file bab2.tex!
  - Gambar 2.3 ditempatkan menggunakan *specifier* **h**.

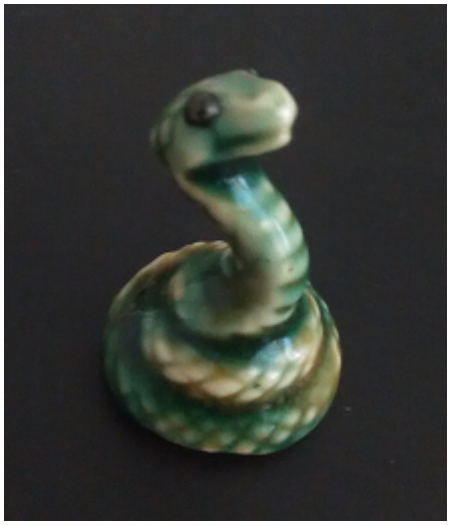
Gambar 2.2: Ular kecil

2.3.7 Kode Program

Kode program dalam bahasa tertentu seringkali harus ditulis di dalam bab, bukan hanya dilampirkan di bagian Lampiran. Kode 2.3 menampilkan penggunaan karakter-karakter yang umum digunakan



Gambar 2.1: Gambar *Serpentes* dalam format png



Gambar 2.3: *Serpentes* jantan

dalam sebuah program yang ditulis dengan bahasa C.

Kode 2.3: Kode untuk menampilkan karakter-karakter aneh

```
// 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
```

2.3.8 Notasi

Simbol-simbol (matematika) yang sering digunakan sepanjang penulisan skripsi, dapat dimasukkan ke dalam "Daftar Notasi". Daftar ini ada di halaman depan sebelum Bab 1. Cara memasukkan sebuah simbol ke dalam Daftar Notasi adalah menggunakan perintah `\nomenclature`. Contoh:

```
\nomenclature[]{$A$}{luas kandang ular}
```

Argumen opsional digunakan untuk mengurutkan notasi. Silahkan lihat sendiri dokumentasi package `nomencl`

DAFTAR REFERENSI

[1] de Berg, M., Cheong, O., van Kreveld, M. J., dan Overmars, M. (2008) *Computational Geometry: Algorithms and Applications*, 3rd edition. Springer-Verlag, Berlin.

[2] van Kreveld, M. J. (2004) Geographic information systems. Bagian dari Goodman, J. E. dan O'Rourke, J. (ed.), *Handbook of Discrete and Computational Geometry*. Chapman & Hall/CRC, Boca Raton.

[3] Buchin, K., Buchin, M., van Kreveld, M. J., Löffler, M., Silveira, R. I., Wenk, C., dan Wiratma, L. (2013) Median trajectories. *Algorithmica*, **66**, 595–614.

[4] van Kreveld, M. J. dan Wiratma, L. (2011) Median trajectories using well-visited regions and shortest paths. *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Chicago, USA, 1-4 November, pp. 241–250. ACM, New York.

[5] Lionov (2002) Animasi algoritma sweepline untuk membangun diagram voronoi. Skripsi. Universitas Katolik Parahyangan, Indonesia.

[6] Wiratma, L. (2010) Following the majority: a new algorithm for computing a median trajectory. Thesis. Utrecht University, The Netherlands.

[7] Wiratma, L. (2022) Coming Not Too Soon, Later, Delay, Someday, Hopefully. Disertasi. Utrecht University, The Netherlands.

[8] van kreveld, M., van Lankveld, T., dan Veltkamp, R. (2013) Watertight scenes from urban lidar and planar surfaces. Technical Report UU-CS-2013-007. Utrecht University, The Netherlands.

[9] Rekhter, Y. dan Li, T. (1994) A border gateway protocol 4 (bgp-4). RFC 1654. RFC Editor, http://www.rfc-editor.org.

[10] ITU-T Z.500 (1997) *Framework on formal methods in conformance testing*. International Telecommunications Union. Geneva, Switzerland.

[11] Version 9.0.0 (2016) *The Unicode Standard*. The Unicode Consortium. Mountain View, USA.

[12] Version 7.0 Nougat (2016) *Android API Reference Manual*. Google dan Open Handset Alliance. Mountain View, USA.

[13] Webb, R., Daruca, O., dan Alfadian, P. (2012) *Method of optimizing a text massage communication between a server and a secure element*. Paten no. EP2479956 (A1). European Patent Organisation. Munich, Germany.

[14] Wiratma, L. (2009) Median trajectory. Report for GMT Experimentation Project at Utrecht University.

[15] Lionov (2011) Polymorphism pada C++. Catatan kuliah AKS341 Pemrograman Sistem di Universitas Katolik Parahyangan, Bandung. http://tinyurl.com/lionov. 30 September 2016.

[16] Erickson, J. (2003) CG models of computation? http://www.computational-geometry.org/mailing-lists/compgeom-announce/2003-December/000852.html. 30 September 2016.

[17] AGUNG (2012) Menjajal tango 12. Majalah HAI no 02, Januari 2012.

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
```

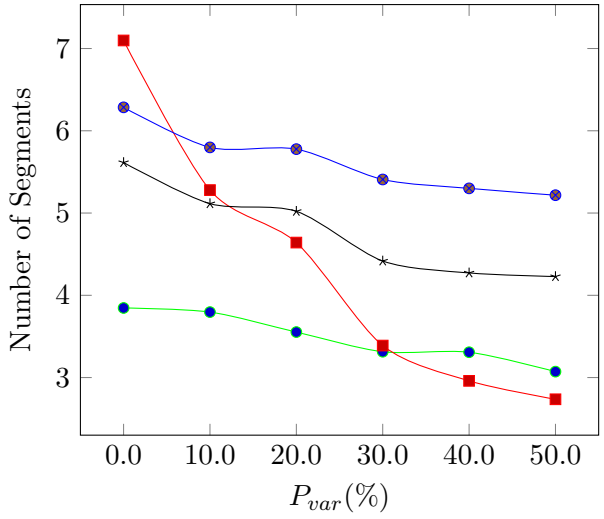
Kode A.2: MyCode.java

```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);
        }
    }

}
```

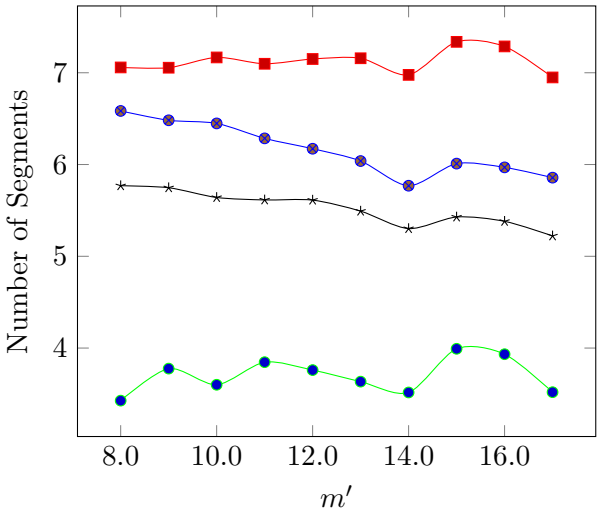
LAMPIRAN B

HASIL EKSPERIMEN

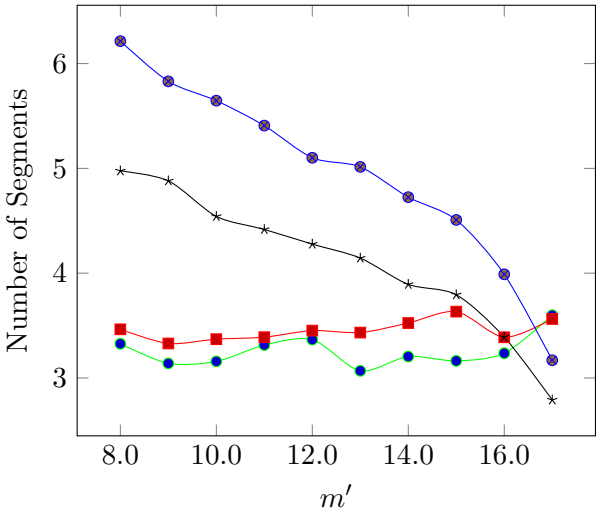
Hasil eksperimen berikut dibuat dengan menggunakan TIKZPICTURE (bukan hasil excel yg diubah ke file bitmap). Sangat berguna jika ingin menampilkan tabel (yang kuantitasnya sangat banyak) yang datanya dihasilkan dari program komputer.



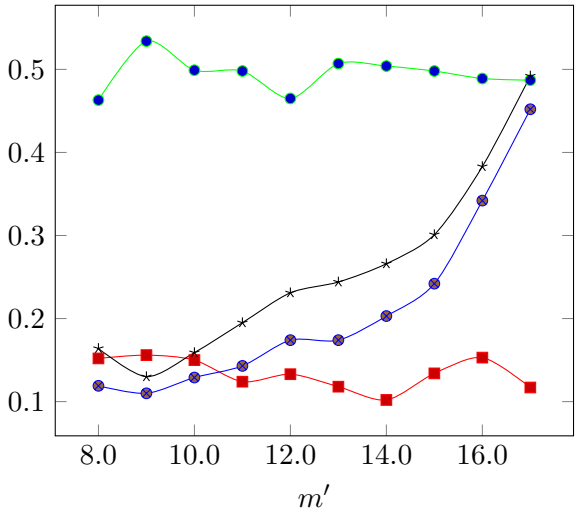
Gambar B.1: Hasil 1



Gambar B.2: Hasil 2



Gambar B.3: Hasil 3



Gambar B.4: Hasil 4