BAB 2

TINJAUAN PUSTAKA

2.1 *String Matching*

String matching adalah proses pencarian semua kemunculan *query* yang selanjutnya disebut *pattern* ke dalam *string* yang lebih panjang (teks). *Pattern* dilambangkan dengan x=x[0..m-1] dan panjangnya adalah m. Teks dilambangkan dengan y=y[0..n-1] dan panjangnya adalah n. *String matching* dibagi menjadi dua, yaitu *exact matching* dan *heuristic matching* (Sarno. Dkk, 2012).

Contoh sebagai berikut.

```
String = COM211
n=6
x=x[0..5]

String = Logika
n=6
x=x[0..5]
```

2.1.1 *Exact Matching*

Exact matching digunakan untuk menemukan *pattern* yang berasal dari satu teks. Contoh pencarian *exact matching* adalah pencarian kata ”pelajar” dalam kalimat ”saya seorang pelajar” atau saya seorang siswa. Sistem akan memberikan hasil bahwa kalimat pertama mengandung kata ”pelajar” sedangkan kalimat kedua tidak.

Algoritma *exact matching* diklasifikasikan menjadi tiga bagian menurut arah pencarian sebagai berikut.

1. Arah pembacaan dari kiri ke kanan.
2. Arah pembacaan dari kanan ke kiri.
3. Arah pembacaan yang ditentukan pemrogram.

2.1.2 *Heuristic Matching*

Heuristic matching adalah teknik yang digunakan untuk menghubungkan dua data terpisah ketika *exact matching* tidak mampu mengatasi karena ada pembatasan pada data yang tersedia. *Heuristic matching* dapat dilakukan dengan perhitungan *distance* antara *pattern* dengan teks. Euzenat(2007) menuliskan contoh dengan perhitungan distance berdasarkan *String Based Tecnique* yaitu *string equality* yang menggunakan algoritma n-*gram similarity*. N-*gram similarity* sering digunakan untuk membandingkan beberapa string. Algoritma ini menghitung sejumlah n-gram bersama seperti serangkaian *n* karakter di antara *string*.

x = COM211
y = COM201

x = “CO”,”OM”,”M2”,”21”,”11”.
y = “CO”,”OM”,”M2”,”20”,”01”.

x$\frown$y = “CO”,”OM”,”M2”
x$\smile$y = “CO”,”OM”,”M2”,”21”,”20”,”11”,”01”

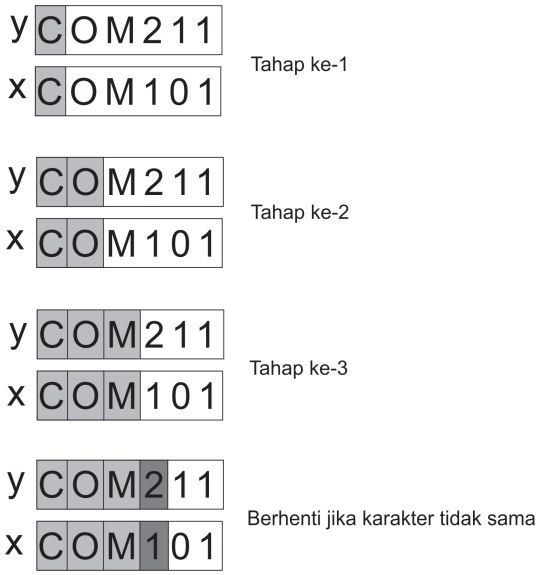
$$\text{Jaccard coefficient } (x,y) = \frac{|x \frown y|}{|x \smile y|} = \frac{3}{7} = 0{,}428$$

Gambar 2.1: *N-Gram Similarity*.

2.2 Algoritma *Brute Force*

Algoritma *brute force* adalah algoritma untuk mencocokkan *pattern* dengan semua teks antara 0 dan n-m untuk menemukan keberadaan *pattern* dalam teks. Secara rinci, langkah-langkah yang dilakukan algoritma ini saat mencocokkan *string*.

1. Algoritma *brute force* mulai mencocokkan *pattern* dari awal teks.
2. Dari kiri ke kanan, algoritma ini mencocokkan karakter per karakter *pattern* dengan karakter pada teks yang bersesuaian, sampai salah satu kondisi berikut terpenuhi.
   (a) Karakter di *pattern* dan di teks yang dibandingkan tidak cocok.
   (b) Semua karakter di *pattern* cocok. Kemudian algoritma memberitahukan penemuan posisi ini.
3. Algoritma kemudian terus menggeser *pattern* sebesar satu ke kanan, dan mengulangi langkah ke-2 sampai *pattern* berada di ujung teks.



Gambar 2.2: Algoritma *Brute Force*.

2.3 Pengertian Sistem Informasi

Sistem informasi merupakan kumpulan dari perangkat keras dan perangkat lunak komputer serta perangkat manusia yang mengolah data menggunakan perangkat keras dan lunak tersebut. Data memiliki peranan dalam sistem informasi. Data yang dimasukkan adalah sebuah sistem informasi dapat berupa formulir-formulir, prosedur dan bentuk data lainnya. Komponen-komponen sistem informasi terdiri dari sebagai berikut.

1. *Input*

   Input adalah semua data yang dimasukkan ke dalam sebuah sistem informasi.

2. Proses

   Proses merupakan kumpulan prosedur untuk memanipulasi *input* yang kemudian disimpan dalam bagian basis data dan seterusnya diolah menjadi suatu *output* yang digunakan oleh penerima.

3. *Output*

   Output merupakan semua keluaran atau hasil dari model yang sudah diolah menjadi suatu informasi yang berguna dan dapat dipakai penerima. Komponen ini berhubungan langsung dengan pemakai sistem informasi dan merupakan tujuan akhir dari pembuatan sistem informasi.

4. Basis data

   Basis data merupakan kumpulan data yang saling berhubungan satu sama lain yang disimpan dalam perangkat keras komputer dan diolah menggunakan perangkat lunak. Basis data merupakan kumpulan *file* yang memiliki keterkaitan antara satu *file* dengan *file* yang lain sehingga membentuk satu bangunan data.

5. Teknologi

   Teknologi merupakan bagian yang berfungsi untuk memasukkan *input*, men-

golah *input* dan menghasilkan keluaran. Teknologi ini meliputi *hardware*, *software*, *brainware*.

6. Kendali

   Kendali merupakan semua tindakan yang diambil untuk menjaga sistem informasi tersebut agar berjalan dengan lancar dan tidak mengalami gangguan (Kristanto, 2003).

2.4 PHP *(Hypertext Preprocessor)*

2.4.1 Pengertian PHP *(Hypertext Preprocessor)*

PHP *(Hypertext Preprocessor)* merupakan bahasa pemrograman yang berbentuk *scripting*, sistem kerja dari program ini adalah sebagai *interpreter* bukan sebagai *compiler* (Nugroho, 2004). Menurut Kadir bahwa ”*Interpreter* adalah perintah menerjemahkan *statement* program. Sedangkan bahasa *compiler* adalah semua perintah di dalam program diterjemahkan terlebih dahulu, baru kemudian semuanya dijalankan” (Kadir, 2005).

Bahasa pemrograman merupakan sebuah paket bahasa yang digunakan untuk membuat sebuah bahasa turunan. Bahasa turunan dapat berupa bahasa pemrograman, atau dapat juga berupa hasil akhir yang disebut dengan istilah aplikasi pemrograman. Menurut cara prosesnya, bahasa pemrograman dapat dikategorikan menjadi dua, yaitu sebagai berikut.

1. Bahasa *Compiler*

   Bahasa *compiler* adalah bahasa yang mengubah *script* program ke dalam *source code*, selanjutnya dari bentuk *source code* diubah menjadi bentuk *object code*, bentuk dari kode objek menghasilkan *file* yang lebih kecil dari *file* mentah sebelumnya. Selanjutnya bentuk kode objek berubah menjadi sebuah

program yang siap dijalankan tanpa adanya program bantu pembuatnya, sehingga hasil dari pemrograman yang berbentuk kompiler membuat sebuah program yang berstatus sebagai program eksekusi. Contoh dari program *compiler* adalah *Pascal, Visual C, Lazarus*.

2. Bahasa *Interpreter*

   Bahasa *interpreter*, *script* mentah tidak harus diubah ke dalam bentuk *source code*. Sehingga pada saat menjalankan bentuk program, kode dasar secara langsung dijalankan tanpa harus melalui proses pengubahan ke dalam bentuk *source code*.

   Contoh penggunaan perintah PHP sebagai berikut.

```
<?php
$teks=$_POST['isiteks'];
$teks="\n".$teks." ";
$rows= explode("\n", $teks);
array_shift($rows);
```

Dalam pemrograman PHP kelompok informasi yang ditampilkan dalam halaman *browser* memerlukan perintah untuk menampilkan informasi tersebut. sebagai contoh pada pemrograman *Pascal* menggunakan perintah *while* dan *writeln* untuk menampilkan informasi. Fungsi *cout* dalam bahasa C dan C++, sedangkan pada PHP menggunakan perintah *echo* dan *print*.

Kelebihan yang dimiliki oleh pemrograman PHP adalah dapat disisipkan ke dalam *tag-tag* HTML. Namun, dengan kelebihan yang dimiliki, PHP juga mampu berdiri sendiri tanpa berada di sela-sela program lain. Contoh program PHP yang berada pada *tag* HTML sebagai berikut.

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

2.4.2 Tipe Data

Tipe data adalah suatu bentuk data yang dideklarasikan pada saat membuat tabel atau program. Tipe data memberi pengaruh pada setiap data yang dimasukkan ke dalam sebuah tabel. Data yang dimasukkan harus sesuai dengan tipe data yang dideklarasikan (Nugroho, 2004).

Lima tipe data yang terdapat pada pemrograman PHP.

1. *Integer*

   Integer adalah tipe data yang berisikan data semua bilangan yang besarnya *range* sama dengan data pada bahasa C, yaitu antara -2,147,483,64 sampai +2,147,483,64 pada *platform* 32-bit.

   Contoh sebagai berikut.

   ```
   $angka=234;
   ```

2. *Floating Point*

   Floating Point adalah tipe data yang berguna untuk menyimpan bilangan desimal atau pecahan. Sebagai contoh (0.1),(1.3),(1.7),(1.8),(9.7),(2.4).

   Contoh Penulisannya adalah sebagai berikut.

   ```
   <?php
   $a = 1.234;
   $b = 1.2e3;
   $c = 2E-6;
   $d = 1.2E8;
   ?>
   ```

3. *Character*

   Character merupakan bentuk yang sama dengan tipe data *varchar* yaitu

mampu menangani data sampai dengan 225 karakter. Tetapi dari ke dua tipe data tersebut memiliki perbedaan yang sangat signifikan yaitu dalam hal media penyimpanan data. Sebagai contoh adalah apabila membuat kolom *varchar*(25) maka data yang dimasukkan paling banyak adalah 25 *digit* tetapi dapat juga memasukkan data kurang dari 25 *digit*. Apabila dalam penyimpanan data dalam bentuk *char*(4) maka harus dimasukkan data paling banyak 4 karakter, jadi apabila memasukkan sebanyak 2 karakter maka, data tersebut tetap dibaca 4 karakter, sehingga keadaan tersebut memboroskan ketersediaan memori komputer. (Nugroho, 2004)

Contoh penulisan tipe data *string* adalah sebagai berikut.

```
$teks=$_POST['isiteks'];
$teks="\n".$teks." ";
$rows= explode("\n", $teks);
```

4. *Array*

   Larik atau *array* adalah penampung sejumlah data bertipe sama dan menggunakan satu *identifier*. *Array* adalah penampung data yang disebut dengan variabel. Perbedaan antara *array* dan variabel adalah terletak pada kapasitas penampungannya. Variabel bertipe *int* hanya dapat menampung sebuah bilangan bulat dan variabel bertipe *char* hanya bisa menampung sebuah karakter ASCII. *Array* bertipe *int* mampu menampung sejumlah bilangan bulat, sedangkan *array* bertipe *char* mampu menampung sejumlah karakter ASCII (Ngoen, 2009).

   Tipe data *array* atau larik merupakan tipe *compound* primitif, dan terdapat juga pada bahasa-bahasa pemrograman lain. Tipe data *array* digunakan untuk menyimpan banyak data dalam satu variabel. *Array* dibagi menjadi dua yaitu *numeric array* dan *assosiative array*.

1. *Numeric Array*

   Pemanggilan data pada *numeric array* berdasarkan dengan angka, angka dimulai dari 0 sampai dengan jumlah data yang dimasukkan. Contoh *Numeric Array* sebagai berikut.

```
#$no[$baris]=$gets[0];
$kopel[$baris]=$gets[0];
$matkul[$baris]=$gets[1];
$sks[$baris]=$gets[2];
$nilai[$baris]=$gets[3];
$pengambilan[$baris]=$gets[4];
$semester[$baris]=$gets[5];
```

2. *Assosiative Array*

   Pemanggilan data *array* tersebut menggunakan *string* yang ditentukan oleh *User*, dengan kata lain pemanggilan menggunakan angka diganti menjadi *string* yang diinginkan. Contoh penggunaan *assosiative array* adalah sebagai berikut.

```
$kopel[$baris]=str_replace(" ","",$kopel[$baris]);
$nilai[$baris]=str_replace(" ","",$nilai[$baris]);
```

5. *Object*

   Tipe data *object* adalah tipe data yang dapat berupa bilangan, variabel ataupun fungsi. Dengan dibuat data *object* ini dapat membantu *programmer* untuk membuat sebuah program. Data ini dapat disertakan ke dalam program, sehingga meringkas beberapa fungsi dan dapat memperkecil ukuran *file*. Semakin kecil ukuran *file* semakin singkat waktu yang dibutuhkan untuk mengakses *file* tersebut.

Contoh penulisan data *object* adalah sebagai berikut.

```
<?php
function brute_force(&$kopel_m,&$kopel_db,&$stat){
if(strlen($kopel_m)!=strlen($kopel_db)){
return $stat;
}else{
for($i=0;$i<strlen($kopel_db);$i++){
if(substr($kopel_m,$i,1)==substr($kopel_db,$i,1)){$stat=true;
}else{
$stat=false;
break;
return $stat;
}
}
return $stat;
}
}
```

Semua variabel dalam bahasa PHP diawali dengan tanda dolar tanpa mempedulikan jenis nilai yang ditampungnya, baik karakter, *integer, string*, maupun bilangan *floating point* dan *array*. Semuanya ditulis dalam bentuk yang mirip dan secara otomatis PHP selalu mengingat tipe data yang disimpan. Secara umum variabel dalam PHP ada tiga macam, yaitu *script*, variabel yang dikirim dari HTML, dan variabel bawaan lingkungan PHP.

2.5 *Xampp*

XAMMP merupakan suatu paket*software* yang terdiri dari Apache, MySQL, dan PHP. Apache adalah sebuah aplikasi yang memungkinkan suatu komputer menjadi *web server*. MySQL adalah DBMS (*Database Management System*), yaitu suatu sistem yang berfungsi untuk mengolah data dalam *database*. Sedangkan PHP adalah bahasa pemrograman *server side coding* yang sering digunakan untuk menciptakan halaman *web*. Penggunaan paket *software* ini memudahkan penulis dalam pembangunan sistem terutama dalam pengolahan *database* (Kadir, 2005).

2.6 *MySQL*

MySQL merupakan *database* yang kuat dan stabil, digunakan sebagai media penyimpanan data. *MySQL* juga merupakan sebuah *database server* yang mampu mengelola *database*. *Database sever* yang memiliki kemampuan mengolah data dengan baik, diantaranya adalah *Oracle* dan *PostgreSQL* (Nugroho, 2004).

Dalam penggunaan *MySQL* dipadukan dengan bahasa pemrograman PHP, hal ini dikarenakan penggunaan keduanya memiliki kehandalan dalam menangani permintaan data. Kemampuan yang dimiliki oleh *MySQL* adalah mampu mendukung *Relational Database Management System* (RDMS) sehingga bisa menangani data sebuah perusahaan yang berukuran besar.

MySQL adalah salah satu jenis *database server* yang sangat terkenal. *MySQL* menggunakan bahasa *SQL* untuk mengakses basis datanya. Untuk melakukan administrasi secara lebih mudah terhadap *MySQL*, dapat menggunakan *software* tertentu, diantaranya adalah PHP*MyAdmin* dan *MySQL Yog*.

2.6.1 Kelebihan *MySQL*

Sebagai *database* yang memiliki konsep *database* yang modern, *MySQL* memiliki beberapa kelebihan menurut Nugroho (2004).

1. Portabilitas. MySQL dapat berjalan stabil pada berbagai sistem operasi seperti Windows, Linux, FreeBSD, Mac Os X Server, Solaris, Amiga, dan sebagainya.

2. *Open Source*. MySQL didistribusikan secara *open source*, di bawah lisensi GPL sehingga dapat digunakan secara cuma-cuma.

3. *Multiuser*. MySQL dapat digunakan oleh beberapa *user* dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

4. *Performance Tuning*. MySQL memiliki kecepatan yang menakjubkan dalam menangani *query* sederhana, dengan kata lain dapat memproses lebih banyak SQL per satuan waktu.

5. Jenis Kolom. MySQL memiliki tipe kolom yang sangat kompleks, seperti *signed / unsigned integer, float, double, char, text, date,* atau *timestamp*.

6. Perintah dan Fungsi. MySQL memiliki operator dan fungsi secara penuh yang mendukung perintah *Select* dan *Where* dalam perintah (*query*).

7. Keamanan. MySQL memiliki beberapa lapisan sekuritas seperti *level subnet-mask*, nama *host*, dan izin akses *user* dengan sistem perizinan yang mendetail serta sandi terenkripsi.

8. Skalabilitas dan Pembatasan. *MySQL* mampu menangani basis data dalam skala besar, dengan jumlah rekaman (*records*) lebih dari 50 juta dan 60 ribu tabel serta 5 milyar baris. Selain itu batas indeks yang dapat ditampung mencapai 32 indeks pada tiap tabelnya.

9. Konektivitas. *MySQL* dapat melakukan koneksi dengan klien menggunakan protokol TCP/IP, Unix *socket* (UNIX), atau *Named Pipes* (NT).

10. Lokalisasi. *MySQL* dapat mendeteksi pesan kesalahan pada klien dengan menggunakan lebih dari dua puluh bahasa. Meski pun demikian, bahasa Indonesia belum termasuk di dalamnya.

11. Antar Muka. *MySQL* memiliki *interface* (antar muka) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (*Application Programming Interface*).

12. Klien dan Peralatan. *MySQL* dilengkapi dengan berbagai *tools* yang dapat digunakan untuk administrasi basis data, dan pada setiap peralatan yang ada disertakan petunjuk *online*.

13. Struktur Tabel. *MySQL* memiliki struktur tabel yang lebih fleksibel dalam menangani *ALTER TABLE*, dibandingkan basis data lainnya semacam PostgreSQL ataupun Oracle.

2.7 Perintah SQL Yang Digunakan

Perintah-perintah SQL yang digunakan dalam pengembangan sistem sebagai berikut.

1. Menampilkan *database*.
   show database;

2. Membuat *database*.
   create database databasename;

3. Menghapus *database*.
   drop database databasename;

4. Melihat isi *database*.

   show tables;

5. Memilih *database*.

   use databasename;

6. Membuat tabel baru.

   create table tablename;

7. Melihat struktur tabel.

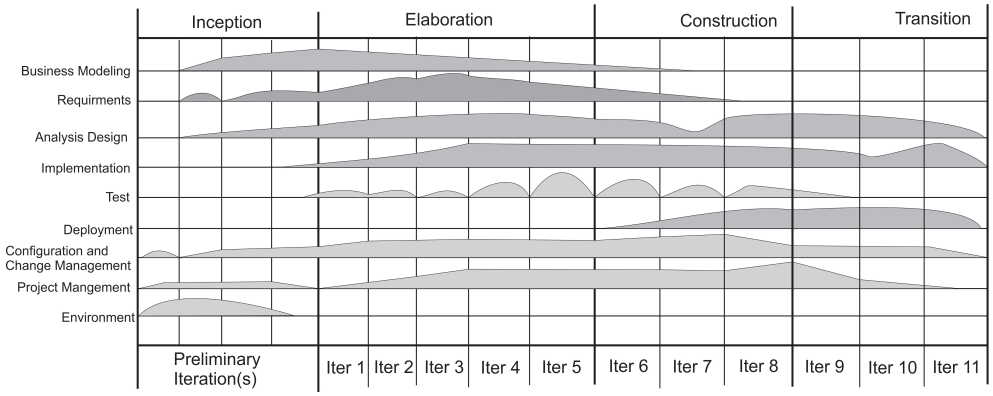
   describe tablename;

8. Memasukkan data kedalam tabel.

   insert into tablename;

2.8 *Rational Unified Process*

Rational Unified Process (RUP) merupakan proses rekayasa perangkat lunak yang menggunakan pendekatan disiplin untuk menetapkan tugas dan tanggung jawab dalam pengembangan sistem (Manalil, 2010). RUP diciptakan, dikembangkan dan dikelola oleh *Rational Software* sekarang IBM. Tujuan dari RUP adalah menghasilkan perangkat lunak berkualitas tinggi yang yang memenuhi kebutuhan pengguna dan dapat diprediksi penjadwalan dan biaya pengembangannya (Kruchten, 2003).



Gambar 2.3: Arsitektur *Rational Unified Process* (Manalil, 2010)

Aktifitas dalam pengembangan perangkat lunak menggunakan RUP terfokus pada pengembangan model dengan menggunakan *Unified Model Language* (UML), karena menggunakan konsep berorientasi objek atau object oriented. Gambar 2.3 dapat dilihat bahwa RUP memili dua dimensi yaitu:

1. Dimensi pertama, menggambarkan aspek dinamis dalam sebuah pengembangan perangkat lunak, yang dijabarkan dalam beberapa fase yang memiliki *major milestone* atau tonggak utama sebagai tanda berakhirnya fase tersebut. Setiap fase dapat terdiri dari satu atau lebih pengulangan atau iterasi. Dimensi horizontal ini terdiri atas *Inception, Elaboration, Construction dan Transition*.

2. Dimensi kedua, merupakan dimensi vertikal yang mewakili aspek-aspek statis dari sebuah pengembangan perangkat lunak terdiri dari empat elemen penting, yakni siapa yang melakukan, apa, bagaimana dan kapan. Dimensi ini terdiri atas *Business Modeling, Requirement, Analysis and Design, Implementation, Test, Deployment, Configuration* dan *Change Management, Project Management, Environment*.

2.9 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan serangkaian kegiatan yang bertujuan untuk menemukan kesalahan dalam sebuah sistem, serta bertujuan untuk memastikan apakah sistem telah bekerja sesuai dengan spesifikasi. Pengujian perangkat lunak memiliki dua tujuan utama, yaitu menemukan *bug* atau kesalahan dan memastikan sistem telah bekerja sesuai yang diinginkan (Ehmer, 2011).

Masalah yang dihadapi dalam pengujian perangkat lunak yaitu waktu yang dibutuhkan dalam mengumpulkan data, ketidakpuasan *user* terhadap *software* yang dibutuhkan, kualitas *software* yang buruk, dan sulit dalam pengelolaan *software*. Pengujian sistem dan perangkat lunak memiliki aturan yang berfungsi sebagai sasaran pengujian, diantaranya sebagai berikut.

1. Pengujian adalah proses eksekusi suatu program dengan maksud menemukan kesalahan.

2. Pada *testcase* harus memiliki probabilitas tinggi, untuk menemukan kesalahan yang belum ditemukan sebelumnya.

2.9.1 Pengujian *Blackbox*

Blackbox testing merupakan pengujian yang memperhatikan atau memfokuskan kepada faktor fungsionalitas dan spesifikasi perangkat lunak. *Blacbox testing* tidak membutuhkan pengetahuan mengenai, alur internal, struktur atau implementasi dari *software under test* (SUT). Tidak seperti *whitebox testing* yang dilakukan pada awal proses pengujian, *blackbox testing* dilakukan dibeberapa tahapan berikutnya.

Ujicoba *blackbox* memeriksa beberapa aspek sistem, tetapi memeriksa sedikit mengenai struktur logikal *internal software*. Teknik pengujian *(*blackbox) berfokus pada domain informasi dari perangkat lunak, dengan melakukan *test case* dengan

mempartisi domain *input* dari suatu program dengan cara yang memberikan cakupan pengujian yang mendalam.

Empat keuntungan yang diperoleh dari jenis pengujian *blacbox* sebagai berikut.

1. Penguji tidak harus menguasai pemrograman.
2. Kesalahan dari perangkat lunak ataupun *bug* sering kali ditemukan oleh kelompok penguji yang berasal dari pengguna.
3. Hasil dari *blackbox testing* dapat memperjelas kejanggalan yang mungkin timbul dari eksekusi dari pengguna.
4. Proses *testing* lebih cepat dibandingkan dari pengujian *whitebox*.