BAB V
UJI COBA DAN EVALUASI

Pada bab ini akan dijelaskan hasil uji coba dan evaluasi program yang telah selesai diimplementasi.

5.1 Lingkungan Uji Coba

Lingkungan uji coba yang akan digunakan adalah,

1. Perangkat Keras
   Processor Intel® Core™ i3-2310M CPU @ 2.10 GHz
   RAM 4 GB
   Sistem Operasi 64-bit
2. Perangkat Lunak
   Sistem Operasi Windows 8.0
   Integrated Development Environment Code::Blocks 13.12

5.2 Skenario Uji Coba

Ada dua skenario uji coba yang akan dilakukan pada subbab ini. Dua skenario tersebut adalah uji coba kebenaran program dan uji coba kinerja program.

5.2.1 Uji Coba Kebenaran

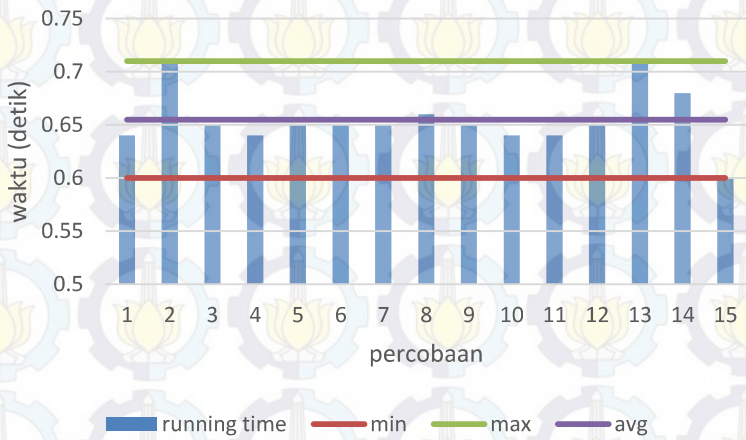
Uji coba kebenaran merupakan uji coba yang dilakukan untuk menguji kebenaran implementasi metode yang digunakan pada program. Uji coba dilakukan dengan mengirim kode sumber pada situs SPOJ dengan kode permasalahan CTSTRING yang diangkat sebagai studi kasus Tugas Akhir ini. Setelah kode sumber dikirim, SPOJ akan memberikan umpan balik terhadap hasil berjalannya program. Hasil yang didapatkan dari SPOJ adalah *accepted*, waktu yang diperlukan untuk penyelesaian permasalahan adalah 0.60 detik dan memori yang dibutuhkan adalah 3MB seperti yang ditunjukkan pada **Gambar 5.2.1**. Hal tersebut menunjukkan bahwa implementasi yang dilakukan berhasil menginterpretasikan *regular*

expression dan mengubahnya ke bentuk DFA untuk melakukan pengecekan *string* yang dapat diterima.

Gambar 5.2.1 Umpan balik dari situs SPOJ

Kemudian dilakukan uji coba dengan mengirim kode sumber sebanyak 15 kali pada permasalahan yang sama untuk melihat variasi hasil yang dihasilkan oleh program. Hasil uji coba tersebut dapat dilihat pada **Gambar 5.2.2**, **Gambar 6.2.1** dan **Tabel 6.2.1**.

Dari hasil uji coba yang dilakukan waktu yang dibutuhkan bervariasi antara 0.6 s.d. 0.71 dengan rata-rata 0.65 detik namun memori yang dibutuhkan program tetap 3 MB.



Gambar 5.2.2 Grafik hasil uji coba pada SPOJ sebanyak 15 kali

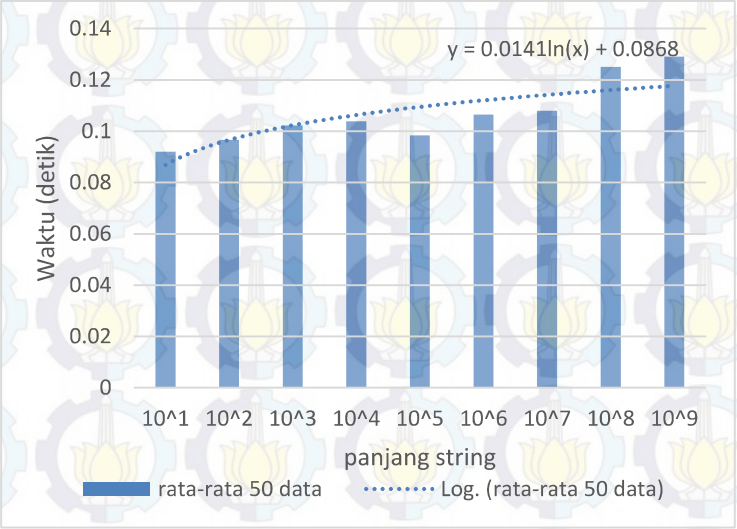
5.2.2 Uji Coba Kinerja

Uji coba kinerja adalah skenario uji coba yang dilakukan dengan parameter tertentu untuk melihat bagaimana performa dari program yang dibuat. Pada uji coba ini parameter yang digunakan adalah panjang *regular expression* yang harus

diproses dan L di mana program harus menghitung berapa banyak *string* yang bisa diterima *regular expression* apabila panjang *string* adalah L.

5.2.2.1 Pengaruh Panjang L Terhadap Waktu

Uji coba yang pertama kali dilakukan adalah dengan memberikan program beberapa panjang L yang berbeda yakni bilangan $10^1, 10^2, 10^3, 10^4 \ldots 10^9$. Panjang *regular expression* yang digunakan adalah 100 karakter. Pada uji coba yang dilakukan, setiap bilangan L yang berbeda akan diuji dengan *regular* expression yang sama sebanyak 50 kali kemudian di rata-rata untuk memperoleh waktu rata-rata untuk memproses *regular expression* dengan panjang 100 dan *string* dengan panjang L.



Gambar 5.2.3 Grafik rata-rata hasil uji coba untuk L yang berbeda

Hasil uji coba tersebut dapat dilihat di **Gambar 5.2.3**. Uji coba ini dimaksudkan untuk melakukan tes performa ketika *regular expression* dicocokan dengan sebuah *string*. Performa

yang diharapkan dalam uji coba ini adalah logaritmik. Untuk *string* yang semakin panjang maka waktu yang dibutuhkan akan semakin lama.

Pada **Gambar 5.2.3** dapat dilihat bahwa dengan perbedaan L yang diberikan, semakin besar L maka semakin besar pula waktu yang diperlukan untuk mengeksekusi program, dan perkembangan waktu yang dihasilkan memenuhi waktu logaritmik seperti yang ditujukkan pada gambar. hal dikarenakan algoritma perkalian matriks yang digunakan menggunakan algoritma dengan kompleksitas $O(\log(L)\, xN^2)$ dimana *L* adalah besar pangkat dan *N* adalah dimensi matriks.

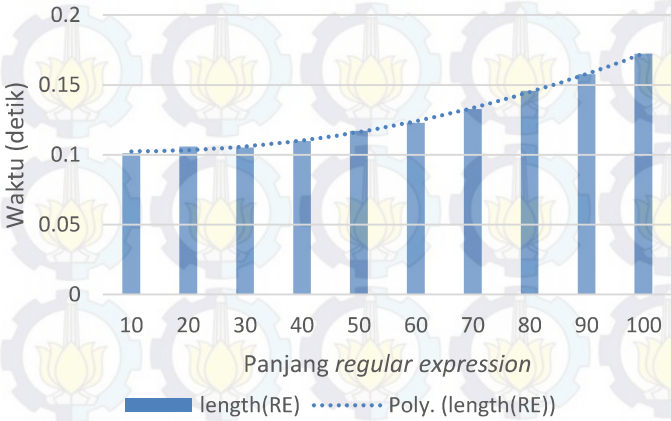
Dari hasil uji coba yang dilakukan dapat dilihat bahwa besar atau kecilnya *L* memiliki pengaruh terhadap waktu yang diperlukan program untuk melakukan evaluasi permasalahan dengan kompleksitas logaritmik.

5.2.2.2Pengaruh Panjang Regular Expression Terhadap Waktu

Uji coba yang dilakukan selanjutnya adalah dengan menggunakan *regular expression* acak dengan operator yang digunakan hanya operator *concatenate* yang dibangkitkan dengan sebuah program pembangkit uji kinerja. Operator *concatenate* digunakan karena memberikan hasil DFA yang lebih konsisten untuk setiap input dengan panjang *regular expression* yang berbeda, di mana semakin panjang *regular expression* maka *state* DFA yang dihasilkan juga akan semakin banyak pula. Parameter yang digunakan pada uji coba adalah panjang *regular expression*. Panjang yang diujikan adalah 10, 20, 30, 40, 50, 60, 70, 80, 90, dan 100 karakter. Bilangan L yang digunakan pada uji coba ini adalah 10^9.

Data hasil uji coba tersebut dapat dilihat pada **Gambar 5.2.4**. Uji coba ini dimaksudkan untuk melakukan tes performa konversi *regular expression* menjadi NFA dan kemudian DFA. Performa algoritma yang diharapkan adalah polinomial berderajat 2, Pada **Gambar 5.2.4** terlihat jika *regular expression* yang perlu dievaluasi semakin panjang makan semakin besar

pula waktu yang diperlukan untuk melakukan konversi terhadap *regular expression* tersebut dan waktu berjalan program memenuhi kurva polinomial berderajat 2, hal ini dikarenakan saat pembentukan NFA operasi yang diperlukan adalah sebanyak $O(2xN)$ atau bisa dibilang linear untuk setiap operator *concatenate* yang dilakukan, sedangkan untuk konstruksi NFA ke DFA diperlukan operasi sebanyak $O(M^2)$ dimana M adalah jumlah *state* NFA.



Gambar 5.2.4 Grafik uji coba dengan panjang RE yang berbeda

Dari hasil tersebut dapat dilihat bahwa semakin panjang *regular expression* yang diberikan, maka semakin besar pula waktu yang dibutuhkan program untuk melakukan konversi *regular expression* tersebut.