# METODE PENELITIAN

Metode penelitian terdiri dari empat tahap utama yaitu studi literasi, perancangan sistem, analisis hasil, dan evaluasi hasil. *Figure* 3‑1 merupakan alur tahap-tahap dari metode penelitian.

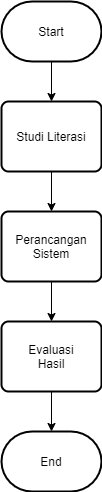


Figure 3‑1 Flowchart Metode Penelitian

## Studi Literatur

Studi literatur merupakan bagian awal dalam pengerjaan Tugas Akhir. Kegiatan yang terdapat pada studi literatur antara lainya mencari, membaca, serta memahami pustaka yang berkaitan dengan topik pendeteksi *code clones*. Sumber dari literasi yang dapat dijadikan acuan dapat berasal dari jurnal, paper, *thesis*, buku, dan penelitian yang terkait dengan topik pendeteksi *code clones*.

## Perancangan Sistem

Sistem yang akan dirancang menerima inputan berupa source code dari satu atau lebih program. Sistem yang akan dirancang memiliki lima tahap utama yaitu *input source codes*, *token parsing*, *string matching*, *format output* dan *evaluate result*. merupakan alur dari sistem pendeteksi *code clones*.

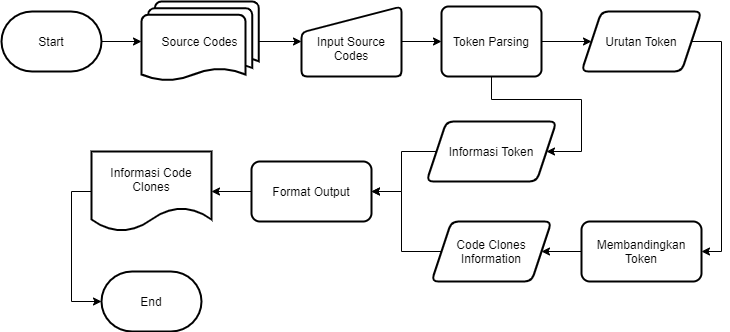


Figure 3‑2 Flowchart Sistem Pendeteksi Code Clones

### Input *Source Code*

Pada tahap ini dilakukan pembacaan file *source code* yang menjadi input pada sistem ini. Input dapat berupa aplikasi dari *source code Python*. Sistem dapat menerima input dari beberapa *source codes* maupun satu *source code*. Jika sistem menerima banyak inputan *source codes* maka sistem akan menyisipkan ke satu buah file sehingga terbaca seperti satu buah *source code*. Figure 3‑3 merupakan contoh source codes yang dapat diterima sistem.

Figure 3‑3 Python source code

### *Token Parsing*

Figure 3‑4 merupakan tiga tahap utama dalam mengubah inputan *source codes* menjadi sebuah *token sequence* dan informasi baris fungsi/prosedur dari token tersebut. Tahapan ini dibagai menjadi tiga bagian, yaitu *remove white space*, *shorten names*, dan *tokenize*. Hasil proses ini adalah hasil transformasi *token sequence* informasi seluruh fungsi dari *token sequence*.



Figure 3‑4 Tahap Token Parsing

#### *Remove White Space*

Tahap ini berfokus terhadap penghapusan indentasi di setiap baris *source codes* dan spasi tambahan yang tidak diperlukan. Tahap ini memiliki tujuan untuk menghapus data yang dapat merusak hasil dari *string matching* sehingga data mudah untuk diproses pada tahap *string matching*.



Figure 3‑5 Hasil python source code setelah diproses tahap remove white space

#### *Shorten Names*

Proses pada tahap ini untuk menghapus partikel redundan yang berada pada program seperti baris komentar dan prefix/suffix variabel. Tahap ini memiliki tujuan untuk meminimalisir data yang akan ditransformasi menjadi token pada tahap *tokenize*. Figure 3‑6 merupakan contoh hasil dari tahap *shorten names*.



Figure 3‑6 Hasil python source code setelah diproses shorten names

#### *Tokenize*

Pada tahap ini, sistem mengganti seluruh *identifier* variabel, tipe, dan konstanta menjadi token, kecuali *language construct* seperti lambang operator. Penggantian ini membuat porsi kode dengan variabel yang berbeda nama dapat terdeteksi sebagai *clone*.



Persamaan 3‑1 Hasil python source code setelah diproses tahap tokenize

### Membandingkan Token

Sistem yang akan dirancang akan mendeteksi *code clone* pada level fungsi. Satu fungsi yang telah ditemukan akan dibandingkan dengan satu fungsi lain untuk diperiksa *code* *pattern* menggunakan algoritma *string matching*. Kedua fungsi yang memiliki *code* *pattern* yang sama dengan fungsi lain disebut *clone pair.*

Sistem yang akan dirancang memiliki dua model algoritma *string matching* yang digunakan untuk menjadi eksperimen efisiensi mendeteksi *code clones*. Model pertama menggunakan algoritma *Knuth Morris Pratt pattern searching*. Sistem akan mencatat waktu yang dibutuhkan untuk menjalankan algoritma *string matching* yang dipakai. Hasil *clone pair* dan waktu berjalan setiap model algoritma merupakan *output* dari tahap ini.

### *Format Output*

Pada tahap ini, sistem mengeluarkan seluruh informasi fungsi yang ada pada *source codes*, *clone pairs*, dan waktu yang diperlukan untuk mendeteksi *code clones*. Penamaan function yang dikeluarkan sistem ditambah dengan nama file sebagi *prefix* yang dipisah oleh tanda “.”, misalkan fungsi A terdapat pada file bernama X sehingga penamaan fungsi menjadi “X.A”.

*Output* pada tahap ini berupa file berformat excel yang memiliki dua kolom, *function pairs* dan *clone pairs*.Kolom *function pairs* berisi pasangan dua fungsi yang ada pada *source codes* dan kolom *clone pairs* berisi nilai “yes” atau “no” tergantung kepada pasangan dua fungsi kolom *function pairs*. Kolom *clone pairs* bernilai “yes” jika pasangan dua fungsi merupakan *code clones* dan bernilai “No” jika pasangan dua fungsi bukan merupakan *code clones*. Informasi waktu yang diperlukan untuk mendeteksi *code clones* terdapat pada baris terakhir pada file excel yang dipisahkan dengan baris dengan nilai *function pairs* “End”. Tabel 3‑1 adalah contoh *output* dari tahap ini.

|  |  |
| --- | --- |
| Function Pairs | Clone Pairs |
| File A.Function A-File A.Function B | Yes |
| File A.Function A-File A.Function C | No |
| End |  |
| Time | 59 (minutes) |

Tabel 3‑1 Format Output Sistem Pendeteksi Code Clones

## Evaluasi Hasil

Hasil dari tahap format output akan dibandingkan dengan hasil dari penelitian “A Code Clone Oracle” oleh Krutz dan Le [3]. Sistem akan menghitung jumlah prediksi clone pairs yang benar sesuai hasil penelitian Krutz dan Le dan menghitung akurasi dengam menggunakan Equation 3‑1 Perhitungan Akurasi Sistem Pendeteksi Code Clones. Untuk mengevaluasi performansi waktu, nilai waktu yang didapatkan pada format output akan dibandingkan dengan berapa Line Of Codes dari source codes, sehingga akan didapatkan nilai waktu yang diperlukan untuk menjalankan source codes dengan LOC tertentu.

Equation 3‑1 Perhitungan Akurasi Sistem Pendeteksi Code Clones