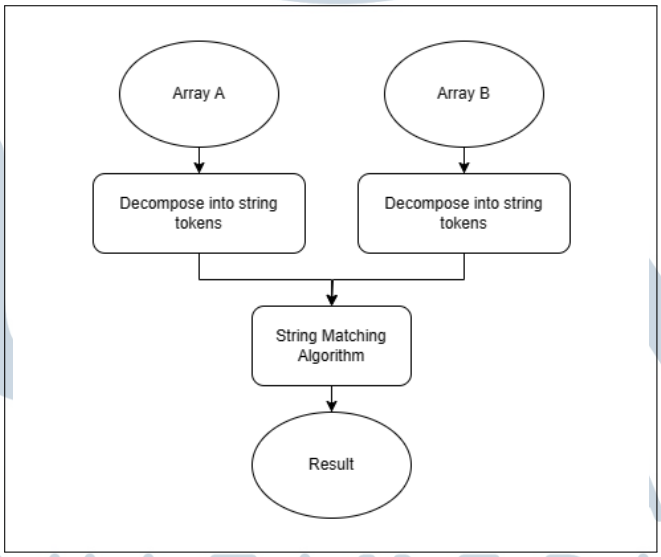
BAB 2
LANDASAN TEORI

2.1 String Matching

String matching adalah metode yang digunakan untuk mencocokkan simbol atau huruf (*string*) dalam suatu rangkaian tertentu [20]. Teknik ini dipelajari untuk memahami bagaimana komputer dapat menemukan pola atau rangkaian pola yang telah diberikan. Prinsip dasar dari *string matching* adalah bagaimana menemukan simbol atau huruf dalam teks dari suatu kasus [20].

Dalam suatu kasus, misalnya diberikan teks dengan x sebagai karakter dalam array $A[1 \ldots x]$ dan pola lain yaitu dengan Y sebagai karakter dalam array $B[1 \ldots Y]$. Bagaimana menemukan bilangan bulat s yang disebut dengan valid shift. Valid shift ini harus memenuhi kriteria sebagai berikut dimana $0 \leq s < x - y$ dan $A[s+1 \ldots s+y] = B[1 \ldots y]$. Jika B sebagai substring dari A, maka selanjutnya akan diketahui apakah B ada dalam A. Elemen-elemen B dan A dapat berupa karakter atau alfabet seperti $\{0,1\}$ atau $\{A,B,C,D,E,\ldots,Z,a,b,c,d,\ldots,z\}$ [20].



Gambar 2.1. Ilustrasi *String Matching*

Gambar 2.1 Ilustrasi *String Matching* merepresentasikan apa yang telah dituliskan sebelumnya. Terdapat *Array* A dan *Array* B yang dimana kedua array itu akan di proses decompose menjadi *token string*. Setelah menjadi *token string*, barulah algoritma *string matching* ini diterapkan dan dapat dilihat hasil dari prosesnya [20].

Untuk kasus yang membutuhkan *string matching* dengan efisiensi tinggi dari aplikasi, dapat menggunakan algoritma *Boyer Moore* [21]. Contoh penggunaan algoritma ini adalah untuk menyunting suatu teks, dan substitusi komentar kedalam teks. Algoritma ini memungkinkan *string matching* dengan ukuran alfabet moderat dan pola teks yang relatif panjang dalam kecepatan tinggi [21]. Selama pengujian ada kemungkinan penempatan pola A dalam data teks B. jika karakter B[i] = c tidak sebanding dengan pola A[j] maka akan berjalan ketentuan tertentu :

Jika c tidak berisi dan dalam A boleh dimana saja, maka akan geser pola A melalui T[i] Disaat kondisi lain, geser A sampai karakter c sehingga A dapat lurus dengan B[i]. Teknik ini agar menghindari terjadinya tumpukan antara perbandingan pola pergeseran relatif terhadap data teks. Dapat disimpulkan bahwa algoritma *Boyer-Moore* relatif lebih cepat pada alfabet besar (panjang pola teks), hasilnya bukan untuk *string* biner atau membentuk pola pendek, untuk membuat string biner lebih diutamakan menggunakan algoritma *knuth-morris-pratt*. Sedangkan untuk pola-pola yang pendek dapat menggunakan algoritma *naïve* karena saat eksekusi berlangsung worst case akan berbentuk *quadratic* [22].

2.2 Algoritma Boyer-Moore

Algoritma *Boyer Moore* adalah metode pencarian *string* yang sangat efisien saat ini [23]. Algoritma ini, yang ditemukan oleh Bob Boyer dan J. Strother Moore, telah menjadi standar dalam banyak literatur pencarian *string* [23]. Karakteristik kunci dari algoritma *Boyer Moore* adalah melakukan pencocokan string dari kanan ke kiri. Dengan karakteristik ini, ketidakcocokan saat membandingkan *string* akan membuat pergerakan pola melompat lebih jauh untuk menghindari perbandingan karakter *string* yang diperkirakan akan gagal [23] [24].

Boyer Moore adalah salah satu algoritma pencocokan pola yang cukup terkenal. Algoritma ini menggunakan beberapa kasus pengecekan teks (karakter input yang akan dibaca) dengan pola (pola yang akan disaring) [23] [24]. Algoritma Boyer Moore mencari dengan cara membandingkan huruf dengan huruf yang ada di pola yang dicari, dan menggeser pola tersebut hingga posisinya sama dengan teks

yang dicari dan membandingkan kata tersebut. Proses ini disebut lompatan karakter.

Teknik lompatan karakter melakukan suatu aksi ketika perbandingan antara dua karakter berbeda. Ada dua aksi tergantung pada teks s dan kata w yang dimiliki, jika p yaitu karakter pada s yang sedang diproses yang tidak cocok maka ada dua kemungkinan aksi. Mencari karakter yang sesuai dan cara menggeser karakter perbandingan terakhir [24] [25].

Karakteristik utama dari algoritma *Boyer Moore* adalah algoritma ini melakukan pencocokan string mulai dari kanan (belakang) dengan karakteristik tersebut, ketidakcocokan saat terjadi perbandingan *string* akan membuat pergerakan pola melompat lebih jauh untuk menghindari karakter pada string yang diperkirakan gagal. Sehingga proses pencarian string akan lebih optimal [25] [26].

Algoritma *Boyer Moore* adalah algoritma pencarian *string* yang mencocokkan karakter dari kanan ke kiri, dan menggunakan dua aturan untuk menggeser pola ke kanan jika terjadi ketidakcocokan, yaitu aturan *bad character* dan aturan *good suffix*. Algoritma ini membutuhkan dua tabel pra-pencarian, yaitu tabel *bmBc* dan tabel *bmGs*, yang menyimpan jarak antara setiap karakter atau akhiran dengan akhir pola. Rumus matematis algoritma *Boyer Moore* adalah

$$s = \max\{bmBc[\text{teks}[i]] - m + 1 + j, bmGs[j]\} \tag{1}$$

Persamaan 1 merupakan umus matematis s algoritma *Boyer Moore*.

jika terjadi ketidakcocokan, dan

$$s = bmGs[0] \tag{2}$$

Persamaan 2 mewakili nilai awal dari s dalam algoritma *Boyer-Moore*, di mana nilai tersebut diatur ke nilai pada indeks pertama dari *array Good Suffix* (bmGs).

jika pola cocok dengan teks, di mana m adalah panjang pola, i adalah posisi di teks, dan j adalah posisi di pola.

Algoritma Boyer Moore memiliki beberapa keunggulan, seperti kompleksitas waktu terbaik $O(n/m)$, kompleksitas ruang $O(m+\sigma)$, dan dapat menangani kasus-kasus yang sulit. Contoh-contoh penerapan algoritma *Boyer Moore* adalah mencari pola `PAN` di teks `ANPANMAN` dan pola `ABRA` di teks `ABRACADABRA`. Contoh dari penerapan algoritmanya adalah sebagai berikut.

1. ANPANMAN.
   - Teks: ANPANMAN
   - Pola: PAN
   - Tabel bmBc: 'A': 2, 'N': 1, 'P': 3, 'M': 4, 'default': 4
   - Tabel bmGs: [4, 4, 3, 1]
   - Proses pencarian:

Tabel 2.1. Contoh Penerapan Pada Teks ANPANMAN

Teks	Pola	Ketidakcocokan	Geser
ANPANMAN	PAN	-	-
ANPANMAN	PAN	A $\neq$ N	1
ANPANMAN	PAN	N $\neq$ M	4
ANPANMAN	PAN	Cocok	1

2. ABRACADABRA.
   - Teks: ABRACADABRA
   - Pola: ABRA
   - Tabel bmBc: 'A': 1, 'B': 2, 'R': 3, 'C': 4, 'D': 4, 'default': 4
   - Tabel bmGs: [4, 4, 4, 1]
   - Proses pencarian:

Tabel 2.2. Contoh Penerapan Pada Teks ABRACADABRA

Teks	Pola	Ketidakcocokan	Geser
ABRACADABRA	ABRA	-	-
ABRACADABRA	ABRA	C $\neq$ B	2
ABRACADABRA	ABRA	Cocok	1
ABRACADABRA	ABRA	Cocok	1

Terdapat beberapa langkah yang dilakukan ketika dalam tahap proses perhitungan. Pertama, dilakukan pra-pemrosesan pada *pattern* untuk mengisi tabel *bad character* dan tabel *good suffix*. Tabel *bad character* berisi jarak pergeseran yang harus dilakukan jika terjadi ketidakcocokan pada suatu karakter. Tabel *good suffix* berisi jarak pergeseran yang harus dilakukan jika terjadi kesesuaian pada suatu akhiran [23] [26]. Kedua, dilakukan pencocokan antara *pattern* dan teks dengan memulai dari karakter terakhir *pattern*. Jika terjadi kesesuaian, maka pencocokan dilanjutkan ke karakter sebelumnya. Jika terjadi ketidakcocokan, maka dilakukan pergeseran *pattern* dengan memaksimalkan nilai dari tabel *bad character* dan tabel

good suffix [26]. Ketiga, pencocokan diulangi sampai *pattern* mencapai akhir teks atau ditemukan kesesuaian penuh antara pattern dan teks. Jika ditemukan kesesuaian penuh, maka nilai kesamaan antara pattern dan teks adalah 100%. Jika tidak, maka nilai kesamaan antara *pattern* dan teks dapat dihitung dengan rumus berikut:

$$\text{Matching Score} = \frac{\text{Jumlah } \textit{pattern} \text{ yang cocok}}{\text{Jumlah } \textit{pattern}} \times 100\% \quad (3)$$

Persamaan 3 menghitung *Matching Score* dalam suatu algoritma pencocokan *string*. *Matching Score* dihitung dengan membagi jumlah *pattern* yang cocok dengan jumlah *pattern*, kemudian dikalikan dengan 100% untuk mendapatkan hasil dalam bentuk persentase.

2.3 YAKE (Yet Another Keyword Extractor)

YAKE (Yet Another Keyword Extractor) adalah metode ekstraksi kata kunci sederhana, otomatis, tanpa pengawasan yang memilih kata kunci paling penting dari teks berdasarkan fitur statistik teks yang diekstraksi dari satu dokumen [27] [28]. YAKE memiliki beberapa fitur diantaranya YAKE merupakan algoritma dengan pendekatan *Unsupervised Learning*. YAKE tidak perlu dilatih tentang kumpulan dokumen tertentu dan tidak bergantung pada kamus, corpora eksternal, ukuran teks, bahasa, dan domain [27] [28]. Maka dari itu, YAKE dapat digunakan dengan teks dalam bahasa apa pun dan domain apa pun. Lalu YAKE dirancang untuk bekerja dengan satu dokumen, sehingga ideal untuk tugas-tugas seperti ekstraksi kata kunci dari artikel individual atau *posting blog* [27]. YAKE juga dapat digunakan pada teks apa pun tanpa memerlukan korpus pelatihan. YAKE menggunakan pendekatan *Unsupervised Learning* yang dibangun berdasarkan fitur yang diekstraksi dari teks dan dapat diterapkan pada dokumen yang ditulis dalam berbagai bahasa tanpa memerlukan pengetahuan tambahan [27] [28]. Hal ini berguna untuk banyak tugas dan situasi di mana akses ke korpus pelatihan terbatas atau terbatas [27].

2.4 Sortation Algorithm

Sortation Algorithm dalam konteks Python biasanya mengacu pada berbagai algoritma pengurutan yang tersedia dalam bahasa pemrograman Python [29]. Python sendiri memiliki fungsi pengurutan bawaan seperti sort() dan sorted() yang menggunakan algoritma *Timsort* [29]. *Timsort* adalah algoritma pengurutan *hybrid* yang berasal dari *merge sort* dan *insertion sort* yang dirancang untuk bekerja dengan baik pada berbagai tipe data dunia nyata [30]. Algoritma ini implementasikan oleh Tim Peters pada tahun 2002 untuk digunakan pada bahasa pemrograman Python [30].

2.5 Curriculum Vitae

Curriculum vitae (CV) adalah dokumen yang menjelaskan resume. Isi resume mencakup informasi lengkap dan kronologis tentang data pribadi, latar belakang pendidikan, prestasi, pengalaman kerja, skill, deskripsi, dan sebagainya [6]. CV umumnya digunakan sebagai salah satu dokumen yang diperlukan saat melamar pekerjaan.

2.6 PY2PDF

PY2PDF adalah sebuah *library* Python yang digunakan untuk memanipulasi PDF. *Library* ini menyediakan berbagai fungsi untuk memanipulasi konten PDF, seperti membagi, menggabungkan, memotong, dan mengubah orientasi halaman. PyPDF2 juga mendukung enkripsi dan dekripsi file PDF [31]. *PY2PDF* juga dapat digunakan untuk mengubah kode Python menjadi format PDF. *Library* ini sangat berguna untuk dokumentasi dan pembuatan laporan, karena memungkinkan pengguna untuk menghasilkan representasi visual dari kode mereka dalam format yang mudah dibaca dan dibagikan [31].