Aplikasi String Matching untuk Mengidentifikasi Topik dan Algoritma pada Judul Makalah Strategi Algoritma

Vito Ghifari - 13520153
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520153@std.stei.itb.ac.id

***Abstrak*—Pada mata kuliah IF2211 Strategi Algoritma, terdapat tugas makalah yang diberikan kepada mahasiswa untuk melakukan eksplorasi pada materi yang telah diajarkan. Setiap makalah strategi algoritma mempunyai judul tertentu yang berisi nama algoritma dan topik yang dibahas. Sebagian besar dari makalah yang telah dibuat mempunyai pola tertentu. Pada makalah ini, dibahas mengenai kata kunci algoritma yang digunakan serta pola untuk mengekstrak topik yang dibahas melalui judul makalah tersebut menggunakan *regular expression*.**

Kata kunci—string matching; Knuth-Morris-Pratt; Boyer-Moore; regular expression; makalah strategi algoritma

I. PENDAHULUAN

Algoritma merupakan suatu langkah demi langkah yang terhingga dari instruksi-instruksi yang terdefinisikan dengan jelas untuk memecahkan masalah tertentu. Pada konteks pemrograman, algoritma digunakan sebagai penyelesaian untuk masalah ataupun submasalah pada program yang dibuat. Tidak sembarang instruksi dapat disebut sebagai algoritma. Oleh karena itu, agar suatu kumpulan instruksi dapat disebut sebagai algoritma, kumpulan instruksi tersebut harus mempunyai ciri-ciri: jelas dan tidak ambigu, masukan yang terdefinisi, luaran yang terdefinisi, terbatas (tidak mengadung kondisi perulangan yang tak terbatas), layak (*feasible*), dan tidak terikat oleh bahasa pemrograman tertentu.

Dari algoritma ini, dikenal istilah bernama strategi algoritma, yaitu merupakan pendekatan umum untuk memecahkan persoalan secara algoritmis sehingga dapat diterapkan pada bermacam-macam persoalan dari berbagai bidang komputasi. Strategi algoritma terhubung terhadap bidang keinformatikaan sehingga menjadi salah satu mata kuliah wajib dari program studi Teknik Informatika Institut Teknologi Bandung dengan kode mata kuliah IF2211. Mata kuliah ini membahas berbagai jenis algoritma yang digunakan untuk memecahkan permasalahan yang ada, salah satunya adalah *integer knapsack problem*.

Pada mata kuliah strategi algoritma, terdapat beberapa tugas yang diberikan kepada mahasiswa untuk mengasah kemampuan mahasiswa tersebut terhadap penggunaan algoritma yang telah dipelajari. Salah satu tugas mata kuliah tersebut adalah pembuatan makalah. Makalah strategi algoritma yang dibuat dapat berisi penerapan atau implementasi suatu algoritma pada permasalahan dunia nyata. Contohnya adalah penerapan algoritma *greedy* untuk membantu penjadwalan pengerjaan tugas. Selain itu, terdapat juga makalah berisi penyelesaian dari permainan dan *puzzle* dengan pendekatan algoritma *greedy*, *branch and bound*, dan lain sebagainya.

Lingkup makalah strategi algoritma yang dibuat sebagian besar tertutup pada algoritma yang telah diajarkan pada mata kuliah strategi algoritma. Karena itu, jenis algoritma yang diimplementasikan dari judul makalah strategi algoritma dapat diidentifikasi dengan suatu kamus pencarian. Pola kalimat pada sebagian besar judul makalah strategi algoritma juga cenderung mirip. Oleh karena itu, terdapat kemungkinan bahwa jenis algoritma yang digunakan dan topik yang dibahas pada makalah strategi algoritma dapat diperoleh.

Makalah Mahasiswa Tahun 2021

1. Mencari Rute Pembelian Barang Dalam Acara Uang Kaget Dengan Algoritma Branch and Bound
   Oleh: Hafid Abi Daniswara - 13519028
2. Pengaruh Pemilihan Fungsi Heuristik pada Algoritma AStar Terhadap Kinerja Aplikasi Pencarian Rute Terpendek Diantara Dua Titik
   Oleh: Nicholas Chen - 13519029
3. Penerapan Algoritma String Matching Dalam Personal Assistant Google
   Oleh: Ferdy Irawan Firdaus - 13519030
4. PPenerapan Strategi Algoritma Greedy pada Pemilihan Investasi Cryptocurrency
   Oleh: Daniel Mario Reynaldi / 13519031
5. Penerapan Algoritma Boyer Moore Pada Game Word Search Puzzles
   Oleh: Muhammad Fahkry Malta - 13519032
6. Penerapan Algoritma Pencocokan String pada Penilaian Jawaban Pendek
   Oleh: Muhammad Dzaki Razaan Faza - 13519033
7. Penerapan Algoritma Backtracking dalam Permainan Mainarizumu
   Oleh: Ruhiyah Faradishi Widiaputri/ 13519034
8. Penerapan Algoritma Greedy pada Permainan Kartu 41
   Oleh: Fakhri Nail Wibowo - 13519035
9. Penerapan Algoritma Backtracking dalam Permainan Tents and Trees
   Oleh: Andrew 13519036
10. Pengaplikasian Algoritma DFS dalam Pencarian Jalur Terpendek pada Permainan Ular Tangga
    Oleh: Arsa Daris Gintara 13519037

Gambar 1. Daftar judul makalah strategi algoritma yang dibuat pada tahun 2021

II. LANDASAN TEORI

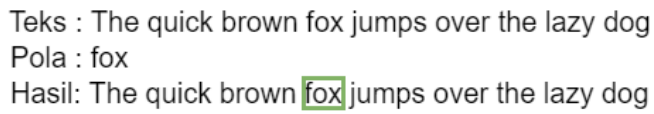
A. *String*

String merupakan tipe data yang berisi deretan karakter sebagai representasi dari teks dengan panjang tertentu. String dapat mengandung huruf, angka, spasi, ataupun simbol-simbol lainnya. Pada umumnya, isi dari string diapit dengan tanda petik untuk menyatakan bahwa suatu deretan karakter merupakan string. Implementasi string pada bahasa pemrograman dapat berupa string yang mungkin berubah (*mutable*) atau yang tidak mungkin berubah (*immutable*).

Pada umumnya, terdapat notasi indeks yang dimulai dari nol dan ditandai dengan kurung siku untuk mendapatkan karakter pada indeks tersebut, contohnya pada string T terdapat T[0] sebagai karakter pertama, T[1] sebagai karakter kedua, dan seterusnya. Notasi ini sering digunakan karena pencarian string akan memanfaatkan posisi suatu karakter pada teks. Sementara itu T[0..i] merupakan substring dari string T, yaitu string yang berisi karakter pada T dari indeks 0 hingga indeks i.

B. *String Matching*

Pencarian string, pencocokan string, atau *string matching* merupakan algoritma untuk mencari suatu string (pola/*pattern*) pada string yang lebih panjang (*text*). Sebuah pola dinyatakan terdapat pada teks apabila teks tersebut memiliki suatu *substring* yang isinya sama dengan string dari pola. Sebaliknya, pola yang tidak terdapat pada *substring* dari teks artinya pola tidak terdapat pada teks.



Gambar 3. Contoh pencocokan string
(Sumber: dokumentasi pribadi)

Pencarian string digunakan di berbagai bidang, beberapa di antaranya adalah pencarian teks dalam *text editor*, *web search engine*, pencocokan rantai asam amino pada rantai DNA, pemeriksa ejaan, *spam filter*, dan deteksi plagiarisme.

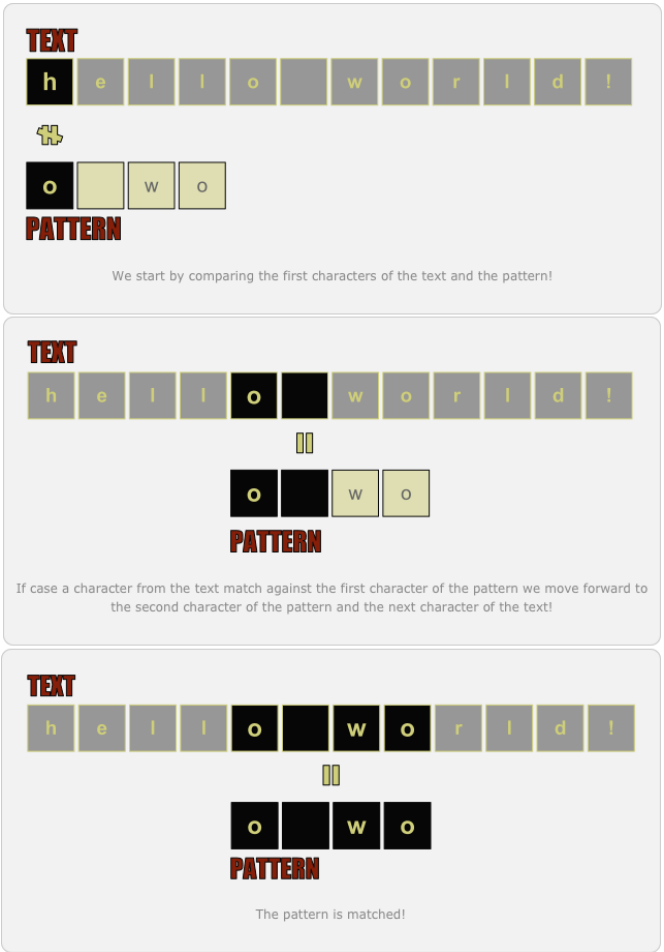
Pada pencarian string, terdapat beberapa algoritma yang umum digunakan. Beberapa di antaranya adalah *naïve algorithm* atau *brute-force*, Knuth-Morris-Pratt (KMP), dan Boyer-Moore. Berikut akan dibahas mengenai ketiga algoritma tersebut.

1) Brute-force

Naïve algorithm atau *brute-force* merupakan algoritma yang paling dasar digunakan untuk melakukan pencocokan string dengan penelusuran dari kiri ke kanan teks. Algoritma ini disebut sebagai algoritma paling dasar karena tidak terdapat pemrosesan awal sebelum melakukan pencocokan dan pendekatan yang digunakan relatif mudah.

Langkah-langkah untuk menemukan pola pada string adalah dengan algoritma *brute-force* adalah sebagai berikut.

1. Periksa karakter pertama pada pola P, yaitu P[0], dan karakter pertama pada teks T, yaitu T[0].
2. Jika kedua karakter ditemukan kecocokan, lanjutkan ke langkah berikutnya. Jika kedua karakter tersebut tidak cocok, lanjutkan pencocokan karakter kedua pada teks dengan karakter pertama pola. Artinya, bandingkan T[1] dengan P[0]. Jika tidak cocok, lanjutkan ke karakter ke-3 pada teks dan seterusnya hingga terdapat kecocokan pada pola atau mencapai akhir teks.
3. Setelah ditemukan kesesuaian karakter indeks *i* pada teks, lanjutkan pencocokan pada karakter ke-2 pada pola dan indeks *i*+1 pada teks. Teruskan hingga akhir pola atau saat ditemukan ketidakcocokan. Apabila ditemukan karakter yang tidak sesuai, ulangi dari P[0] dengan memulai dari indeks *i*+1 pada teks.
4. Pada langkah ke-3, jika terdapat kesesuaian hingga akhir pola, pola tersebut ditemukan pada teks. Pola tesebut dimulai pada indeks *i*. Jika tidak ada kesesuaian, pola ini tidak terdapat pada teks.
5. Apabila pola ditemukan dan T[i] bukan merupakan karakter akhir teks, pencarian dapat dilanjutkan jika ingin menemukan lebih dari satu pola atau dihentikan jika hanya ingin menemukan pola pertama.



Gambar 6. Ilustrasi pencarian string secara *brute-force*
(Sumber: dzone.com)

Algoritma *brute-force* sebagai pencarian string bukan merupakan algoritma yang paling efisien. Pada kasus terburuk, algoritma ini mempunyai kompleksitas waktu sebesar O(mn) dengan n adalah panjang string teks dan m adalah panjang string pola. *Brute-force* bekerja lambat apabila sebagian besar substring pada teks mempunyai karakter yang sama dengan beberapa karakter awal pada pola.

2) Knutt-Morris-Pratt Algorithm (KMP Algorithm)

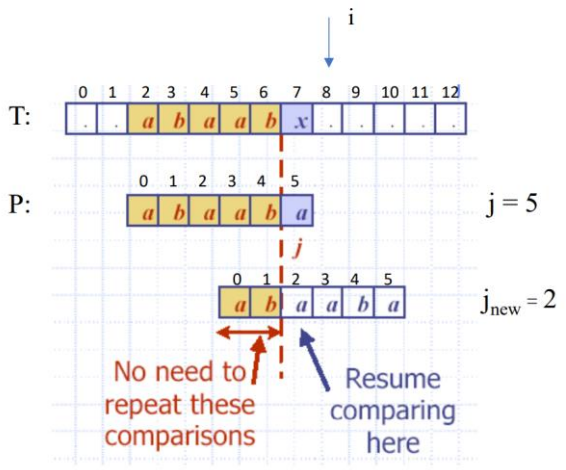
Algoritma Knuth-Morris-Pratt (KMP) merupakan algoritma pencocokan string dengan urutan dari kiri ke kanan, sama seperti algoritma *brute-force*. Namun, algoritma KMP melakukan pergeseran (*shifting*) yang lebih baik karena mempertimbangkan karakter yang terdapat pada pola. Oleh karena itu, algoritma KMP lebih efisien daripada *brute-force*.

Contoh kasus yang dapat dilakukan lebih baik oleh algoritma KMP adalah apabila terdapat ketidaksesuaian antara teks T dengan pola P pada P[j], yaitu T[i] ≠ P[j]. Pada kasus ini, pergeseran yang dilakukan dapat lebih dari satu karakter untuk menghindari perbandingan yang sia-sia. Pergeseran ini adalah sebesar panjang prefiks terbesar P[0..j-1] yang juga merupakan sufiks dari P[1..j-1]. Karena itu, algoritma KMP memerlukan pemrosesan awal sebelum pencocokan string dilakukan.

Untuk suatu pola P yang ingin dicari pada teks T dengan algoritma KMP, langkah penggunaan algoritma ini adalah sebagai berikut.

1. Mulai pencocokan string seperti *brute-force*, yaitu membandingkan karakter pertama pola dengan teks dari indeks 0 hingga terdapat kecocokan pada karakter pertama pola dan teks atau sampai teks habis.
2. Apabila terjadi kecocokan, lanjutkan hingga terjadi ketidakcocokan. Jika kecocokan terjadi sampai akhir pola, pola ini ditemukan pada teks.
3. Jika terjadi ketidakcocokan, geser pencarian sehingga melewati sejumlah karakter dengan pergeseran sebesar panjang dari prefiks terpanjang yang juga merupakan sufiks.
4. Apabila pencarian sudah berada di ujung kanan teks dan tidak terdapat kecocokan pola sampai akhir pola, teks tersebut dinyatakan tidak mengandung pola tersebut.

Contoh pergeseran ketika terjadi ketidakcocokan divisualisasikan pada gambar 7. Pada gambar 7, pola yang dicari adalah *abaaba*. Ketidakcocokan ini terjadi pada indeks ke-4, sehingga substring yang digunakan adalah abaab. Prefiks terpanjang dari substring tersebut yang juga merupakan sufiks adalah ab, karena **ab**aab dan aba**ab**. Panjang prefiks tersebut adalah 2 dan *j* sebagai indeks awal perbandingan dari pola diubah menjadi 2. Akibatnya, jumlah pergeseran yang dilakukan adalah sebanyak panjang substring tersebut dikurangi dengan panjang prefiks, yaitu 5 – 2 = 3.



Gambar 7. Pergeseran pencarian teks pada algoritma KMP
(Sumber: dzone.com)

Pada implementasinya, pergeseran ini memerlukan *failure table* untuk memetakan jumlah pergeseran karakter yang harus diambil. Isi dari *failure table* dihitung menggunakan fungsi pinggiran KMP atau b(k), yaitu untuk mencari panjang prefiks terpanjang P[0..k] yang merupakan sufiks dari P[1..k]. Dengan ini, semua panjang prefiks terpanjang untuk setiap karakter pada pola dapat diketahui.

Setelah dilakukan pengisian *failure table*, perbandingan karakter pada pola dengan karakter pada teks dapat dijalankan. Jika terjadi ketidakcocokan pada indeks *j* dari pola, pencarian dilanjutkan dengan mengubah *j* yang baru sebagai hasil dari fungsi b(k) dengan k = j – 1.

(k = j-1)

j	0	1	2	3	4	5
P[j]	a	b	a	a	b	a

k	0	1	2	3	4
b(k)	0	0	1	1	2

Gambar 8. *Failure table* yang dihitung dengan fungsi pinggiran untuk pola abaaba.
(Sumber: Pencocokan String, Slide IF2211 Strategi Algoritma)

Algoritma KMP untuk pencarian string mempunyai kompleksitas waktu sebesar O(m+n). Pada penghitungan fungsi pinggiran, kompleksitasnya adalah O(m). Sementara itu, pencarian string mempunyai kompleksitas waktu sebesar O(n). Keunggulan algoritma KMP dibanding *brute-force* adalah algoritma ini tidak pernah mundur pada teks karena fungsi pembatas yang telah dibuat sebelumnya, sehingga performa algoritma ini baik dalam hal pemrosesan file yang berukuran sangat besar ataupun file dari jaringan. Di sisi lain, KMP tidak dapat bekerja dengan baik apabila jenis karakter pada pola dan teks sangat banyak karena mudah terjadi

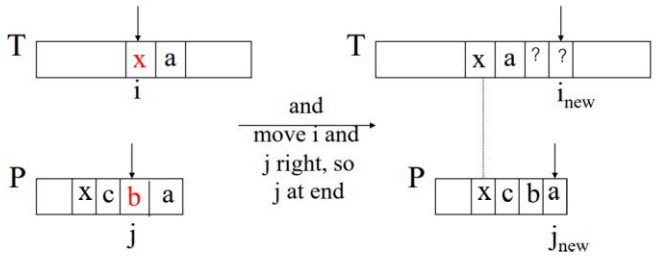
ketidakcocokan. Padahal, algoritma KMP cukup cepat apabila terdapat ketidakcocokan bukan di awal pola.

3) Boyer-Moore Algorithm

Algoritma Boyer-Moore merupakan algoritma yang mempertimbangkan karakter pada pola, sehingga terdapat pemrosesan awal seperti algoritma KMP. Namun, terdapat perbedaan pada algoritma ini dibandingkan algoritma *brute-force* ataupun KMP, yaitu pencarian pola pada teks dimulai dari akhir pola tersebut.

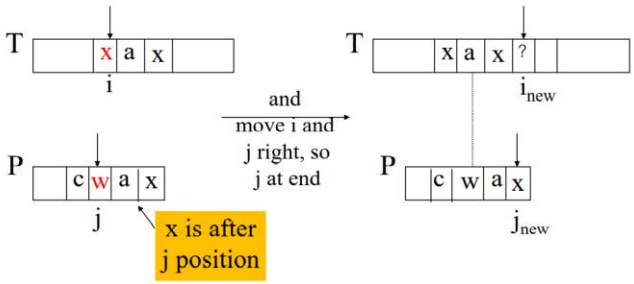
Dasar dari algoritma Boyer-Moore adalah teknik *looking-glass* dan teknik *character-jump*. Pada teknik *looking-glass*, pencarian pola P pada teks T dilakukan dengan melakukan pencocokan mundur P yang bermula pada karakter akhir P. Teknik *character-jump* akan digunakan saat P[j] tidak cocok dengan T[i]. Ketidakcocokan ini terbagi menjadi tiga kasus sebagai berikut dengan contoh karakter T[i] adalah x.

1. Kasus 1
   Pada kasus ini, karakter x terdapat di suatu bagian pada P. Geser P ke kanan hingga kemunculan x paling belakang sejajar dengan x pada T[i].



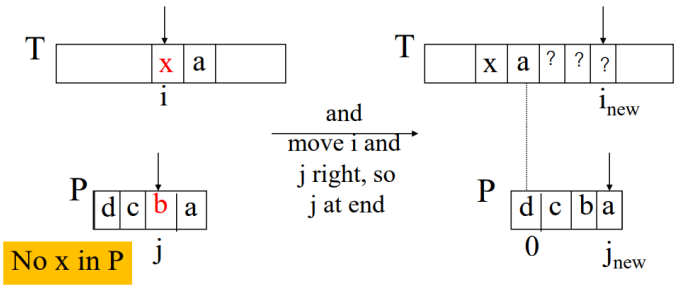
Gambar 9. Kasus 1 pada *character-jump*.
(Sumber: Pencocokan String, Slide IF2211 Strategi Algoritma)

2. Kasus 2
   Karakter x terdapat pada P, tetapi tidak mungkin untuk menggeser P ke kanan karena x berada di sebelah kanan *j*. Jika terjadi kasus ini, geser P ke kanan sebesar 1 karakter menjadi T[i+1].



Gambar 10. Kasus 2 pada *character-jump*.
(Sumber: Pencocokan String, Slide IF2211 Strategi Algoritma)

3. Kasus 3
   Kasus ini terjadi apabila kasus 1 dan 2 tidak berlaku. Geser P agar P[0] sejajar dengan T[i+1].

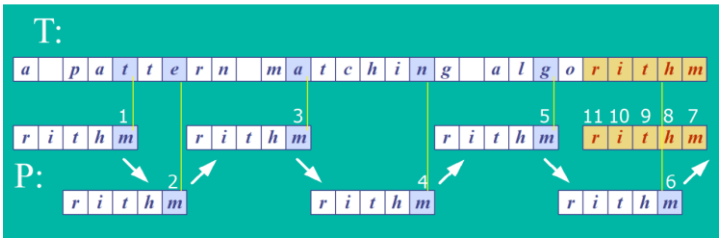


Gambar 11. Kasus 3 pada *character-jump*.
(Sumber: Pencocokan String, Slide IF2211 Strategi Algoritma)

Dari jenis kasus tersebut, algoritma Boyer-Moore dapat diterapkan dengan langkah-langkah di bawah ini.

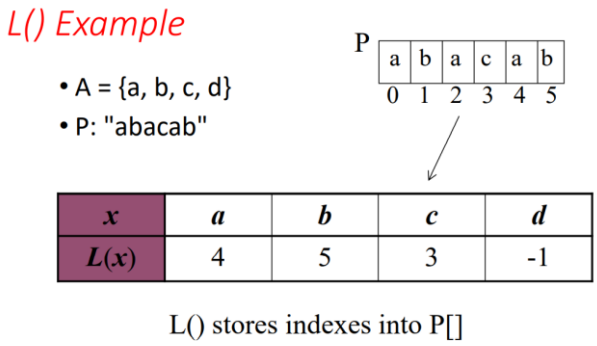
1. Terdapat pola P dan teks T. Lakukan perbandingan antara T[i] dengan P[j], ketika *i* sama dengan *j* dan *j* merupakan indeks dari karakter akhir pola.
2. Lanjutkan dengan perbandingan mundur antara P dan T.
3. Jika karakter pola bersesuaian dengan teks hingga awal karakter pada pola, T mengandung pola P. Jika ketidakcocokan terjadi, lakukan pergeseran P berdasarkan kasus pada teknik *character-jump* dan kembali ke langkah ke-2.
4. Pencarian string selesai apabila P telah ditemukan atau telah mencapai akhir teks.

Contoh penggunaan algoritma Boyer-Moore dipaparkan pada gambar 12. Teks T yang digunakan adalah “a pattern matching algorithm” dan pola P yang dicari pada teks adalah “rithm”.



Gambar 12. Contoh kasus pencocokan string dengan algoritma Boyer-Moore
(Sumber: Pencocokan String, Slide IF2211 Strategi Algoritma)

Pada implementasi algoritma Boyer-Moore, terdapat fungsi kemunculan terakhir atau *last occurrence function* yang dilambangkan dengan L sebagai pemrosesan awal. Fungsi ini memetakan alfabet A pada pola untuk mendapatkan hasil pemetaan berupa indeks terakhir kemunculan suatu karakter apabila teknik *character-jump* dilakukan. Jika karakter tidak terdapat pada pola, indeksnya diganti dengan -1.

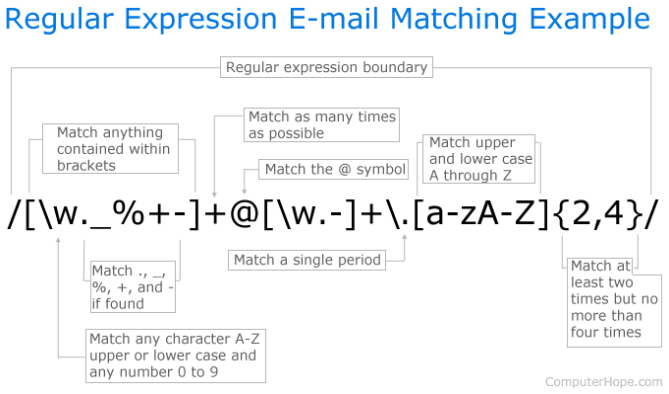


Gambar 13. Contoh *last occurrence function* pada pola "abacab" dan alfabet a, b, c, d
(Sumber: Pencocokan String, Slide IF2211 Strategi Algoritma)

Analisis pada algoritma Boyer-Moore yaitu kompleksitas waktunya adalah sebesar O(nm + A). Algoritma Boyer-Moore akan mencari pola dengan cepat apabila ukuran alfabet (A) besar. Namun, algoritma ini tidak bekerja dengan baik apabila ukuran alfabet kecil, contohnya alfabet yang hanya terdiri atas dua huruf.

C. *Regular Expression*

Regular expression atau disingkat sebagai regex merupakan sebuah string yang mendefinisikan sebuah pola pencarian sehingga pencocokan, pencarian, dan manipulasi teks dapat dengan lebih mudah dilakukan. Regex dapat digunakan untuk melakukan validasi data, mencari pola dalam suatu teks, dan melakukan *find and replace* pada teks yang bersangkutan. Implementasi regex dalam beberapa bahasa pemrograman menggunakan tanda garis miring di awal dan di akhir string regex seperti pada gambar 14.



Gambar 14. Pola regex sederhana untuk e-mail
(Sumber: computerhope.com)

Pada penggunaan regex, sintaks yang digunakan perlu diperhatikan. Suatu sintaks regex dapat menerima sekumpulan string tertentu dan menolak sisanya. Beberapa arti sintaks pada regex dijelaskan pada tabel 1.

Karakter	Arti
\d	Angka dari 0 hingga 9
\w	Huruf, angka, dan garis bawah
\s	*Whitespace* seperti spasi, tab, line break
.	Semua karakter dengan panjang satu
[abcde]	Salah satu karakter pada kurung siku. Pada kasus ini: a, b, c, d, atau e
(abc)	Mengelompokkan satu grup atau lebih
+	Mencocokkan satu karakter atau lebih yang berada di depan tanda ini
?!	Karakter setelah tanda ini tidak boleh ada pada teks
?:	Jangan lakukan *capture* pada grup
(abc|cde|def)	Mencocokkan salah satu dari abc, cde, atau def
*	Mencocokkan nol karakter atau lebih yang berada di depan tanda

Tabel 1. Sintaks regex dan artinya

D. *Makalah Strategi Algoritma*

Pada mata kuliah IF2211 Strategi Algoritma, terdapat tugas makalah yang diberikan kepada mahasiswa yang mengambil mata kuliah tersebut. Dengan pembuatan makalah, diharapkan mahasiswa IF2211 tidak hanya mampu membangun program aplikasi, tetapi juga mampu menulis karya ilmiah. Dengan membuat tulisan, berbagai pemikiran, karya, maupun penelitian dalam bidang informatika dapat dikomunikasikan ke tengah masyarakat.

Adapun tujuan penulisan makalah strategi algoritma adalah sebagai berikut.

1. Memotivasi mahasiswa agar memiliki kemampuan menulis untuk menuangkan ide-ide atau hasil risetnya;
2. Melakukan eksplorasi terhadap isu, metode, dan masalah yang dipelajari dalam pengembangan serta menyebarkan aplikasi yang mendukung teknologi informasi;
3. Sebagai media untuk berbagi informasi hasil-hasil pemikiran dan penelitian.

Pembuatan makalah strategi algoritma oleh mahasiswa berupa *technical report* yang berkaitan dengan topik di bawah ini.

- Algoritma *brute-force*.
- Algoritma *greedy*.
- Algoritma *divide and conquer*.
- Algoritma *decrease and conquer*.
- DFS dan BFS.
- Algoritma runut balik (*backtracking*).

- Algoritma *branch and bound*.
- Algoritma UCS (*uniform-cost search*), *greedy best first search*, dan A*.
- Program dinamis.
- Pencocokan string dan regex.
- Teori P, NP, dan NP Complete.

Penggunaan salah satu atau beberapa algoritma yang telah disebutkan pada umumnya terdapat pada judul. Selain itu, terdapat juga informasi tentang penerapan algoritma tersebut terhadap hal yang diteliti. Sebagai contoh sebuah judul makalah "Penerapan Algoritma Greedy sebagai Solusi dari Permainan A" membahas tentang algoritma *greedy* dan Permainan A. Dari hal ini, terdapat dua hal yang dapat diekstrak informasinya dari judul: jenis algoritma dan topik/objek yang dibahas. Implementasi pencocokan string pada judul makalah dapat menggunakan algoritma seperti *brute-force*, KMP, ataupun Boyer-Moore. Namun, pada makalah ini, regex digunakan karena terdapat berbagai macam kemungkinan kata-kata untuk topik pada judul.

III. Implementasi dan Pengujian

A. *String Regular Expression*

Pembuatan string dari regex diimplementasikan dalam bahasa Python. Penggunaan regex di sini dipisah menjadi dua bagian, yaitu regex pertama untuk identifikasi algoritma yang digunakan dan regex kedua sebagai identifikasi topik.

Regex pertama, yaitu pengelompokan algoritma, memanfaatkan struktur data *dictionary* untuk menyimpan *key* dan *value*. Dengan adanya *key* dan *value*, suatu kata kunci yang terasosiasi dengan algoritma tertentu dapat ditemukan dan diklasifikasikan sebagai algoritma tersebut.

Key	Value
brute-force	brute force
greedy	greedy
divide and conquer	divide and conquer, divide & conquer
BFS	bfs, breadth first search
DFS	dfs, depth first search
backtracking	Runut balik, backtrack
branch and bound	branch and bound, branch & bound
path planning	a star, a*, astar, uniform cost search, ucs
string matching	string, matching, boyer moore, kmp, knuth morris pratt, expression, regex
dynamic programming	dynamic programming, pemrograman dinamis, program dinamis
p vs np	np

Tabel 2. Daftar algoritma dan kata kuncinya pada *dictionary*

Setelah kata kunci yang sesuai ditentukan, *key* dan *value* dari tabel tersebut dibalik sehingga *key* menjadi *value* dan *value* menjadi *key*. Pada akhirnya, *value* yang menjadi *key* ini digunakan sebagai string regex untuk menemukan jenis algoritma dari judul makalah.

Pada bagian identifikasi topik, diperlukan satu string regex untuk mendapatkan sebagian besar topik dari makalah strategi algoritma. Untuk mendapatkan string yang dimaksud, pertama ditentukan dahulu kata-kata yang mungkin muncul sebelum frasa topik. Kata-kata tersebut adalah "pada", "dalam", "untuk", "permainan", "solusi", "penentuan", dan sejenisnya. Setelah itu, kata setelahnya tidak boleh mengandung string "algoritma" agar tidak ter-*capture* kembali nama algoritma yang digunakan. Kata-kata terus di-*capture* hingga akhir kalimat atau kata "dengan", "menggunakan", dan "berbasis". Hasil akhir berupa *raw string* untuk regex yang terbentuk disisipkan sebagai satu baris kode Python di bawah ini.

```
topic_pattern =
r"(?:pada|dalam|untuk|terhadap|dengan|
optimisasi|solusi|game|permainan|perma
salahan|penentuan)
((?!algoritma))(?:[\w\"'\.:\(\)]+
*(?:dengan|menggunakan|berbasis)?)+"
```

B. *Data Uji*

Data uji yang tersedia untuk menguji regex yang telah dibuat adalah kumpulan judul makalah strategi algoritma pada tahun 2021. Judul makalah ini beragam dan beberapa di antaranya terdapat makalah dengan judul dan isi dalam bahasa Inggris. Di bawah ini merupakan tiga judul teratas makalah strategi algoritma pada tahun 2021.

- Mencari Rute Pembelian Barang Dalam Acara Uang Kaget Dengan Algoritma Branch and Bound,
- Pengaruh Pemilihan Fungsi Heuristik pada Algoritma AStar Terhadap Kinerja Aplikasi Pencarian Rute Terpendek Diantara Dua Titik,
- Penerapan Algoritma String Matching Dalam Personal Assistant Google.

Semua data judul makalah pada tahun 2021 sebanyak 214 makalah dibantu dengan pustaka Beautiful Soup pada Python. Pustaka tersebut berguna untuk melakukan *parsing* pada dokumen HTML dan XML sehingga judul makalah dapat dengan mudah didapatkan dari kode sumber web.

C. *Pengujian*

Pengujian regex pada data uji dibantu dengan program dalam bahasa Python. Sebelum melakukan pengujian, judul makalah dibersihkan dari tanda hubung terlebih dahulu. Ini berfungsi untuk mengurangi jumlah pencarian, contohnya adalah *divide-and-conquer* dengan *divide and conquer* yang sebenarnya kedua hal tersebut sama. Selain itu, semua huruf kapital akan dijadikan huruf kecil dengan alasan yang sama.

Judul	Algoritma yang Teridentifikasi
mencari rute pembelian barang dalam acara uang kaget dengan algoritma branch and bound	Branch & bound
pengaruh pemilihan fungsi heuristik pada algoritma astar terhadap kinerja aplikasi pencarian rute terpendek diantara dua titik	Path planning
penerapan algoritma string matching dalam personal assistant google	String matching
optimasi solusi dynamic programming dalam knapsack problem dengan pendekatan bfs	BFS, dynamic programming
penerapan strategi algoritma branch and bound dan dfs pada micromouse	BFS, DFS
penerapan algoritma backtracking dan regular expression dalam pencarian solusi permainan wordscapes	Backtracking, string matching
implmentasi algoritma minimax dan alpha beta pruning pada ai dalam permainan connect 4	-
penerapan algoritma greedy dan brute force dalam permainan big two	Greedy, Brute-force
penentuan rute evakuasi tsunami di pantai pangandaran dengan algoritma a*	Path planning
approximating annuity using binary search algorithm and ternary search algorithm	-

Tabel 3. Hasil pengujian identifikasi algoritma pada sebagian judul makalah

Hasil dari pengelompokkan algoritma dari judul makalah strategi algoritma disertakan pada tabel 3. Pada hasil tersebut, terdapat beberapa judul yang menggunakan lebih dari satu algoritma. Di sisi lain, terdapat judul yang tidak teridentifikasi karena kata kunci yang berada di luar kamus pencarian. Sebagai ringkasan, tabel 4 merupakan jumlah masing-masing algoritma yang teridentifikasi pada makalah strategi algoritma tahun 2021 diurutkan menurun berdasarkan jumlah.

Algoritma	Jumlah
String matching	50
Greedy algorithm	37
Branch & bound	26
Backtracking	23
Path planning	23
Dynamic programming	21
Brute-force	15
BFS	9
-	9
DFS	8
Decrease & conquer	6
Divide & conquer	3
P vs NP	2

Tabel 4. Algoritma yang digunakan pada makalah 2021 dan jumlahnya

Pada tabel 4, dapat terlihat bahwa string matching merupakan bahasan yang paling banyak untuk makalah tahun 2021. Terdapat 9 judul makalah yang tidak dapat diketahui penerapan algoritmanya karena berada di luar kamus pencarian. Artinya, makalah yang berhasil diidentifikasi adalah sebanyak 214 dikurang 9, yaitu 205 makalah.

Pengujian regex kedua, yaitu identifikasi topik, disajikan pada tabel 5. Hasilnya, sebagian besar topik pada makalah berhasil diidentifikasi. Beberapa di antaranya gagal karena regex melakukan *capture* pada nama algoritma atau karena makalah yang dibuat mempunyai judul berbahasa Inggris.

Judul	Topik Teridentifikasi
mencari rute pembelian barang dalam acara uang kaget dengan algoritma branch and bound	Acara uang kaget
pengaruh pemilihan fungsi heuristik pada algoritma astar terhadap kinerja aplikasi pencarian rute terpendek diantara dua titik	kinerja aplikasi pencarian rute terpendek diantara dua titik
penerapan algoritma string matching dalam personal assistant google	personal assistant google
penerapan algoritma greedy pada permainan kartu 41	permainan kartu 41
boyer moore and greedy algorithms to display personalized contents on social media feed	-

aproksimasi perhitungan integral tentu menggunakan algoritma divide and conquer	-
aplikasi uniform cost search untuk minimaxing artefak karakter pada gim "genshin impact"	minimaxing artefak karakter gim "genshin impact"
penggunaan branch and bound dalam menentukan pembelian permainan pada steam sale	menentukan pembelian permainan steam sale
analisis sentimen pada tweet di twitter terhadap larangan mudik 2021 dengan pencocokan string dan library textblob	tweet di twitter larangan mudik 2021
penentuan beban maksimum truk pickup dengan algoritma backtracking	penentuan beban maksimum truk pickup

Tabel 5. Hasil pengujian identifikasi topik pada beberapa judul makalah

IV. KESIMPULAN

Informasi pada judul makalah strategi algoritma dapat diekstrak dengan cara membuat string regex. Program yang dibuat dengan bahasa Python untuk mengidentifikasi algoritma yang digunakan pada makalah sudah baik, yaitu berhasil mendapatkan jenis algoritma pada 205 dari 214 judul makalah yang ada. Untuk bagian identifikasi topik, terdapat berbagai pola pada judul yang ada, tetapi berhasil untuk mendapatkan informasi dari sebagian judul makalah dengan regex. Beberapa judul tidak dapat diidentifikasi karena berbahasa Inggris atau tidak sesuai pola pada string regex.

VIDEO LINK PADA YOUTUBE

Video penjelasan mengenai makalah ini terdapat pada https://youtu.be/qfiegRYKw48

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan yang Maha Esa yang telah memberikan rahmat serta karunia-Nya, sehingga penulis dapat menyelesaikan makalah yang berjudul "Aplikasi String Matching untuk Mengidentifikasi Topik dan Algoritma pada Judul Makalah Strategi Algoritma" tepat pada waktunya. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi Munir, MT sebagai dosen mata kuliah IF2211 Strategi Algoritma pada K3 karena telah memberikan sumber belajar dan mendukung dalam proses penulisan makalah ini.

REFERENSI

[1] Prabhu, R. (2022). "Introduction to Algorithms". https://www.geeksforgeeks.org/introduction-to-algorithms/, diakses pada 14 Mei 2022.

[2] Munir, R. (2022). "Pengantar Strategi Algoritma". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Pengantar-Strategi-Algoritma-(2022).pdf, diakses pada 14 Mei 2022.

[3] Popov, S. (2012). "Algorithm of the Week: Brute Force String Matching". https://dzone.com/articles/algorithm-week-brute-force, diakses 21 Mei 2022.

[4] Muhardian, A. (2020). "Apa itu Regex? dan Apa Manfaatnya dalam Pemrograman?". https://www.petanikode.com/regex/, diakses 21 Mei 2022.

[5] Leylia, M. (2019). "String Matching dengan Regular Expression". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf, diakses 21 Mei 2022.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 20 Mei 2022

Vito Ghifari, 13520153