Aplikasi Algoritma *BFS* dan *String Matching* pada *Tag Suggestions* di *Facebook*

Catherine Pricilla 13514004[1]
Program Studi Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13514004@std.stei.itb.ac.id

***Abstraksi*—Fitur *photo tagging* di Facebook selalu berkembang untuk semakin memudahkan para pengguna dalam menandai dan membagikan foto-foto yang mereka unggah ke situs tersebut. Dengan adanya fitur *tag suggestions* untuk melengkapinya, pengguna tidak perlu lagi secara manual menandai foto secara satu persatu. Fitur *tag suggestions* pada Facebook dapat mendeteksi wajah yang berada pada foto dan menyocokannya dengan wajah pengguna-pengguna Facebook sehingga Facebook dapat menentukan wajah siapa yang berada pada foto. Teknologi ini dikembangkan dengan menggunakan algoritma pencocokan pola. Pada makalah ini akan dianalisis cara kerja fitur *tag suggestions* pada Facebook dengan menggunakan algoritma *Breadth-First-Search* dan berbagai jenis algoritma pencocokan pola yaitu algoritma pencocokan string *Brute Force*, *Knuth-Morris-Pratt* (KMP), serta *Boyer-Moore* (BM).**

Kata Kunci—Algoritma Breadth-First-Search, Algoritma Pencocokan Pattern, Facebook, Tag Suggestions

I. PENDAHULUAN

Seiring dengan kemajuan teknologi yang semakin berkembang dengan pesat, berbagai macam media sosial bermunculan serta berlomba-lomba untuk menjadi yang terbaik. Berbagai usaha telah dilakukan dengan menghadirkan berbagai fitur-fitur menarik dan inovatif. Facebook adalah salah satu media sosial yang sudah berdiri sejak awal era media sosial. Sejak Facebook.com berdiri pada tahun 2004, sudah banyak bermunculan media-media sosial yang baru, jika Facebook tidak mengembangkan fitur-fitur yang ada tentu Facebook lama kelamaan akan terlupakan.

Jaman sekarang, orang-orang lebih memilih untuk mengunggah foto digital ke internet dibandingkan dengan mencetaknya dan menaruhnya ke dalam album foto. Hal ini dikarenakan oleh mudahnya membagikan foto-foto ini dengan teman-teman. Facebook adalah salah satu media sosial yang mempunyai fitur untuk mengunggah foto secara online. Fitur ini juga dilengkapi dengan fitur-fitur menarik lainnya seperti membuat album, photo tagging dan sebagainya. Tetapi, sudah bermunculan berbagai media sosial baru untuk menggunggah dan membagikan foto secara online dengan fitur-fitur yang serupa seperti Instagram. Fitur photo tagging pada Facebook pun sudah terasa tidak istimewa lagi dikarenakan Instagram juga mempunyai fitur yang serupa dengannya.

Namun, sebenarnya fitur *photo tagging* pada Facebook kurang efisien dikarenakan *tagging* harus dilakukan secara manual untuk setiap orang pada suatu foto. Jadi bayangkan saja jika seorang pengguna Facebook mengunggah 100 foto ke dalam suatu album dengan setiap foto terdapat 10 orang yang berbeda, maka pengguna tersebut harus menandai 1000 orang pada foto secara satu persatu. Untuk mengatasi masalah ini, Facebook pun membuat terobosan baru dengan fitur *tag suggestions*-nya. Dengan fitur ini, Facebook dapat mendeteksi setiap wajah-wajah yang ada pada foto dan menentukan nama dari pemilik wajah tersebut.

Facebook menyimpan data wajah dari setiap pengguna yang terdaftar pada Facebook melalui foto-foto dari pengguna tersebut yang sudah ada pada *database* Facebook. Data-data ini berasal baik melalui foto yang pengguna tersebut unggah secara pribadi maupun foto milik pengguna lain yang sudah ditandai untuk diri pengguna tersebut. Dengan kumpulan foto dari pengguna, Facebook menyimpan data seperti jarak mata, bentuk bibir, bentuk wajah seseorang ke dalam teks. Sehingga saat ada foto lain yang diunggah, facebook dapat menyocokan pola dari foto tersebut dengan data wajah para pengguna yang sudah ada pada *database* Facebook itu.

Teknik pendeteksian wajah ini dilakukan dengan menggunakan algoritma pencocokan string, gambar yang diunggah dijadikan bentuk string dan mencocokan polanya dengan pola gambar yang lan. Pada makalah ini akan dibahas tiga buah algortima pencocokan string untuk fitur *tag suggestions* pada Facebook yaitu algoritma *Brute Force*, *Knuth-Morris-Pratt* (KMP), serta *Boyer-Moore* (BM). Dan pencarian data yang cocok pada database menggunakan algoritma *Breadth-First-Search*.

II. LANDASAN TEORI

2.1 *Photo Tagging* pada Facebook

Facebook merupakan sosial media yang menawarkan berbagai macam fitur mulai dari menuliskan status, mengirim pesan singkat antar pengguna Facebook, mencari informasi, hingga mengunggah gambar secara *online*. Untuk fitur pengunggahan gambar pada Facebook, Facebook menawarkan banyak pilihan tambahan seperti mengunggah foto-foto menjadi satu album dan men-*tag* teman pada foto tersebut. Fitur *photo tagging* yang dikembangkan oleh Facebook merupakan hal yang sangat inovatif dan memudahkan

pengguna dikarenakan oleh mudahnya berbagi foto kepada teman-teman yang terlibat dengan foto tersebut. Dengan photo tagging, berbagi gambar kepada teman atau pengguna lain menjadi mudah karena gambar tersebut langsung terhubung dengan orang yang bersangkutan tanpa harus dibagikan lagi, Kemudahan ini juga bertambah dengan fitur *photo tagging* yang dapat langsung mendeteksi wajah-wajah manusia yang berada pada gambar.

Gambar 1. Fitur *photo tagging* pada suatu foto di Facebook

Namun, sebenarnya men-*tag* wajah teman pada setiap foto yang diunggah dapat menjadi suatu pekerjaan yang cukup melelahkan dikarenakan Facebook hanya mendeteksi wajah pada foto tanpa mengenali siapa pemilik wajah tersebut.

Fitur terbaru dari Facebook berupa *tag suggestions* memberikan solusi bagi permasalahan diatas. Dengan *tag suggestions*, Facebook dapat men-*tag* pengguna lain secara otomatis saat pengguna mengunggah foto ke Facebook sehingga pengguna tidak perlu lagi secara manual men-tag semua orang yang berada pada foto. Facebook menggunakan algoritma pencocokan pola untuk mendeteksi serta mengenali seluruh wajah pada foto tersebut.

2.2 Algoritma *String Matching*

Berdasarkan oleh World Agreement Agenda O812, *string* merupakan deretan simbol yang tidak tertentu panjangnya, yang dianggap sebagai panjang satu unit. *String* dapat tersusun atas beberapa hal, baik itu berupa huruf, angka, karakter khusus, maupun karakter Unicode.

Pencarian *string* di dalam teks dapat juga disebut sebagai pencocokan *string* (*pattern/string matching*) yaitu melakukan pencarian semua kemunculan string pendek dengan ukuran *n* yang dinamakan *pattern* dalam suatu *string* panjang dengan ukuran *m* yang dinamakan teks, dimana *n* memiki panjang yang sama atau lebih kecil dibandingkan *m*.

Jadi bisa dirumuskan menjadi;

1. teks (*text*), merupakan *string* berukuran panjang *n* karakter yang lebih panjang dari *pattern*.
2. pola (*pattern*), merupakan *string* berukuran panjang *m* karakter lebih pendek dari teks dan akan dicari di dalam teks.
3. *n* <= *m*.

Algoritma pencarian s*tring* pun terdiri dari berbagai algoritma seperti pendekatan pencocokan *pattern* dan teks dari kiri ke kanan maupun pendekatan pencocokan *pattern* dan teks dari kanan ke kiri. Adapun algoritma-algortima tersebut diuraikan sebagai berikut:

2.2.1 Algoritma *Brute Force*

Algoritma pencocokan string *brute force* tergolong dalam algoritma yang menyocokan *pattern* dengan teks dari kiri ke kanan. Algoritma ini bekerja dengan cara:

1. Pertama-tama menyocokan suatu *pattern* misalkan *P* pada awal suatu teks *T*.
2. Lalu dibandingkan dari kiri ke kanan setiap karakter pada *pattern* dengan dengan karakter yang bersesuaian di dalam teks sampai semua karakter yang dibandingkan cocok atau dijumpai ketidakcocokan karakter.
3. Jika belom ditemukan kecocokan antara semua karakter pada *pattern* dengan teks dan teks belum habis, maka proses ini berulang terus dengan menggeser *pattern* 1 karakter ke kanan sampai karakter teks habis.

```
Pattern: NOT
Teks: NOBODY NOTICED HIM

  NOBODY NOTICED HIM
1 NOT
2  NOT
3   NOT
4    NOT
5     NOT
6      NOT
7       NOT
8        NOT
```

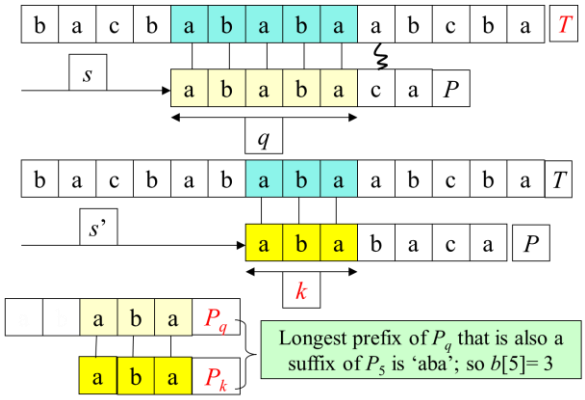
Gambar 2. Pencarian *string* menggunakan algoritma *brute force*

2.2.2 Algoritma *Knuth-Morris-Pratt*

Seperti halnya dengan algoritma *brute force*, algoritma *Knuth-Morris-Pratt* yang dikembangkan

oleh D. E. Knuth, J. H. Morris dan V. R. Pratt, merupakan algoritma pencarian *string* dengan memanfaatkan metode pencocokan *pattern* dengan teks yang bergerak dari kiri ke kanan. Tetapi, berbeda dengan algoritma *brute force*, algoritma *Knuth-Morris-Pratt* tidak menggeser karakter secara satu persatu melainkan memelihara informasi pecocokan sebelumnya dengan menggunakan fungsi pinggiran untuk menentukan jumlah pergeseran karakter. Hal ini membuat algoritma *Knuth-Morris-Pratt* lebih mangkus dibandingkan dengan algoritma *brute force* dikarenakan pemakaian fungsi pinggitan dapat mengurangi waktu secara signifikan dibandingkan melakukan pencarian menggunakan algoritma *brute force*.

Algortima *Knuth-Morris-Pratt* memanfaatkan fungsi pinggiran atau *border function* untuk menentukan jumlah lompatan karakter saat ditemui ketidakcocokan antara *pattern* dan teks. Fungsi pinggiran dapat dihitung sebelum mencocokan *pattern* dan teks dikarenakan hanya dibutuhkan karakter-karakter dari *pattern* untuk menghitung fungsi pinggiran. Jadi untuk mendapatkan fungsi pinggiran dari setiap karakter pada *pattern*, algoritma Knuth-Morris-Pratt memproses *pattern* itu sendiri untuk mencari kecocokan *prefix* dan *suffix*-nya.



Gambar 3. Ilustrasi untuk mencari fungsi pinggiran b(5)

Fungsi pinggiran b(k) untuk *pattern P* dikatakan sebagai ukuran dari *prefix* dari P[1..k] terpanjang yang juga merupakan *suffix* dari P[1..k].

Sebagai contoh, sebuah *pattern P* "ababca" dicari dalam sebuah teks *T* "abaababababcab".

Fungsi pinggiran untuk *pattern* "ababca" adalah sebagai berikut:

k	1	2	3	4	5	6
P[k]	a	b	a	b	c	a
b[k]	0	0	1	2	0	1

Maka, proses pencocokan *string* dengan algoritma *Knuth-Morris-Pratt* dapat diilustrasikan sebagai berikut:

i	1	2	3	4	5	6	7	8	9	10	11
T[i]	a	b	a	b	a	a	b	a	b	c	a

j	1	2	3	4	5	6
P[j]	a	b	a	b	c	a

a	b	a	b	c	a

a	b	a	b	c	a

a	b	a	b	c	a

2.2.3 Algoritma *Boyer-Moore*

Berbeda dengan algoritma *brute force* dan algoritma *Knuth-Morris-Pratt*, algoritma *Boyer-Moore* ini melakukan perbandingan *pattern* dengan teks dimulai dari sisi kanan pattern lalu bergerak ke kiri. Algoritma ini juga mencari *string* dengan melompat maju sejauh mungkin seperti halnya pada *Knuth-Morris-Pratt*, tetapi untuk menentukan banyaknya lompatan karakter, algoritma *Boyer-Moore* menggunakan metode yang berbeda dengan algoritma *Knuth-Morris-Pratt*. Algoritma ini menggunakan dua buah teknik yaitu:

1. *Looking-Glass Technique*

   Teknik ini adalah teknik pencocokan *pattern* dan teks yang begerak mundur, sehingga pencocokan karakter dimulai dari bagian ujung *pattern* dan bergerak menuju bagian awal *pattern*.

2. *Character-Jump Technique*

   Saat ditemukan ketidak cocokan antara karakter *pattern P* dan teks *T*, maka digunakan teknik pelompatan ini. Saat ketidak cocokan terjadi karakter pada T[i] tidak sama dengan karakter pada P[j], dimana karakter pada T[i] adalah x. Maka ada 3 kemungkinan kasus. Kasus pertama adalah terdapat karakter x pada *pattern*. Maka, *P* digeser untuk mensejajarkan karakter x pada *P* dengan T[i]. Kasus kedua adalah terdapat x pada *pattern* namun tidak memungkinkan untuk *P* digeser ke kanan. Maka, *P* digeser ke kanan satu karakter ke T[i+1]. Kasus ketiga adalah jika tidak termasuk dalam kasus

pertama dan kedua, maka P[1] disejajarkan dengan T[i+1].

Untuk memudahkan teknik pelompatan karakter diatas, maka pada awal pembacaan seluruh karakter dari *pattern P* ditentukan dahulu fungsi kemunculan terakhir atau *last occurrence function*. Fungsi ini dibuat dengan menentukan kemunculan terakhir seluruh karakter yang terkandung pada teks pada *pattern*.

Sebagai contoh, sebuah *pattern P* "ababca" dicari dalam sebuah teks *T* "abaababababcab".

Fungsi kemunculan terakhir untuk *pattern* "ababca" adalah sebagai berikut:

x	a	b	c
bc(x)	6	4	5

Maka, proses pencocokan *string* dengan algoritma *Boyer-Moore* dapat diilustrasikan sebagai berikut:

i	1	2	3	4	5	6	7	8	9	10	11
T[i]	a	b	a	b	a	a	b	a	b	c	a

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

a	b	a	b	c	a

a	b	a	b	c	a

a	b	a	b	c	a

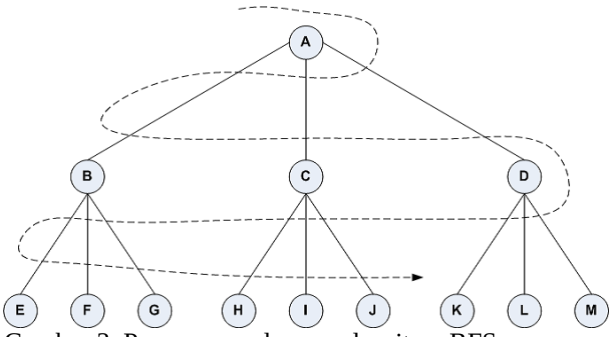
a	b	a	b	c	a

a	b	a	b	c	a

2.3 Algoritma *Breadth-First-Search*

Algoritma *Breadth-First Search* (BFS) adalah algoritma yang melakukan pencarian secara melebar. Jadi pemrosesan pencarian ini dimuali dari satu simpul lalu mengunjungi semua simpul yang bertetangga dengan simpul tersebut dahulu. Lalu, simpul yang belum dikunjungi dan bertetangga dengan simpul-simpul yang tadi dikunjungi, demikian seterusnya. Jika graf berbentuk pohon berakar, maka semua simpul pada aras d dikunjungi lebih dahulu sebelum simpul-simpul pada aras d+1.

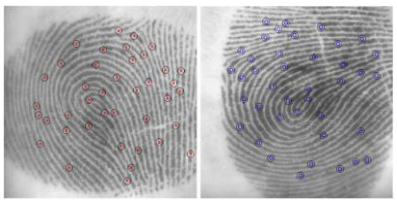
Algoritma *Breadth-First-Search* membutuhkan sebuah antrian *q* untuk menyimpan semua simpul yang telah dikunjungi olehnya. Dimana simpul-simpul yang telah dimasukkan ke dalam antrian akan dijadikan acuan untuk langkah selanjutnya. Setiap simpul yang telah dikunjungi hanya masuk satu kali ke dalam antrian, lalu terdapat juga nilai *boolean* untuk setiap impul untuk menandakan simpul yang telah dikunjungi.



Gambar 3. Pemrosesan dengan algoritma BFS

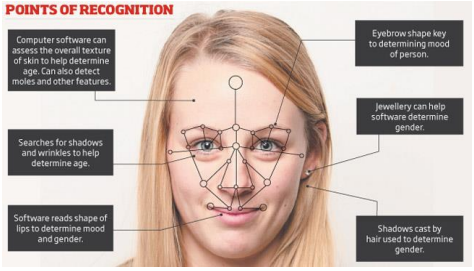
III. Analisis Citra pada *Tag Suggestions* Facebook

Tag suggestions pada Facebook memanfaatkan algoritma *pattern* atau *string matching* yang digunakan untuk melakukan analisis citra dari semua foto yang telah diunggah oleh pengguna-penggunanya. Konsep ini analisis citra ini mirip dengan pemanfaatan algoritma *pattern matching* untuk mengidentifikasi sidik jari dengan menggunakan analisis citra.


Gambar 4. Analisis citra untuk sidik jari

Tertulis pada media sosial Facebook, Facebook memproses setiap citra wajah dari semua pengguna Facebook menjadi sebuah kode unik yang mereka sebut dengan "*template"*, lalu memasukan kode-kode unik ini ke dalam database. Setiap citra yang mengandung wajah dari seorang pengguna, baik itu merupakan foto profil pengguna, foto yang diunggah sendiri oleh pengguna, maupun foto pengguna lain yang mengandung *tag* untuk pengguna tersebut akan diproses untuk membuat sebuah *template string* yang unik. Sehingga, setiap pengguna Facebook yang memiliki foto wajahnya terunggah pada media sosial tersebut memiliki sebuah *template string* yang menyimpan data wajahnya. Kode unik pada template ini didapatkan dengan memperhitungkan fitur-

fitur dari wajah sesorang seperti jarak antara mata, bentuk alis, jarak antara hidung, mata, mulut, dan telinga.

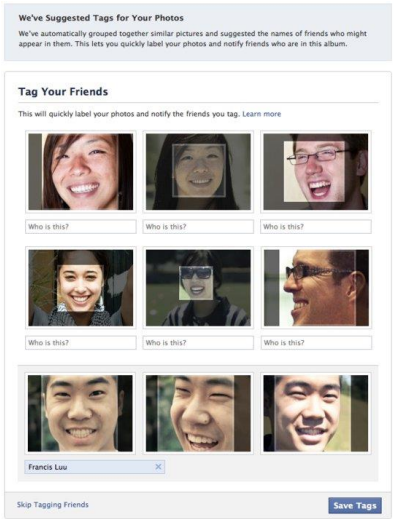


Gambar 5. Poin-poin utama dalam pendeteksian wajah

Dengan poin-poin pendeteksian wajah ini, Facebook dapat menentukan *template* unik untuk setiap wajah, sehingga Facebook dapat mengenali wajah yang sama walau dengan ekspresi yang berbeda-beda pada foto.

Gamba 6. *Facial recognation* dengan ekspresi wajah yang berbeda pada Facebook

Namun, jika suatu pengguna tidak mempunyai foto wajah dirinya sebelumnya di Facebook, Facebook tidak dapat mengenali wajahnya saat mendeteksi wajah tersebut pada foto dikarenakan wajah pengguna tersebut belum memiliki *template* kode unik. Facebook belum dapat seratus persen mengenali setiap wajah pengguna dalam foto, sebagai gantinya Facebook selalu mengelompokan wajah yang serupa menjadi satu grup untuk memudahkan pengguna untuk men-*tag* jika Facebook tidak mengenali wajah tersebut dan tidak men-*tag*-nya secara otomatis.



Gambar 7. *Tag Suggestions* di Facebook

Facial recognition bekerja dengan cara memindai sebuah foto dan mencatat geometri dari wajah-wajah yang terdapat pada foto tersebut, lalu poin-poin wajah ini diubah menjadi suatu *string*. Lalu *string* ini akan dibandingkan dengan *template* wajah pengguna yang terdapat pada *database.* Proses pencocokan ini menggunakan algoritma *pattern* atau *string matching* yang kemudian menghasilkan kecocokan atau ketidak cocokan dengan *template* lain yang berada pada *database* atau foto lain yang diunggah secara bersamaan dalam suatu album foto.

IV. APLIKASI ALGORITMA *BFS* DAN *STRING MATCHING* PADA *TAG SUGGESTIONS*

Untuk melakukan pengenalan antara satu wajah dengan wajah yang lain dibutuhkan algoritma pencocokan *string* untuk mencari kecocokan atau kemiripan antar wajah. Dengan menemukan kecocokan antara wajah pada foto dengan *template* yang berada pada *database*, Facebook dapat dengan otomatis menentukan dan men-*tag* pengguna pada suatu foto tanpa harus dilakukan secara manual oleh pengguna yang mengunggahnya.

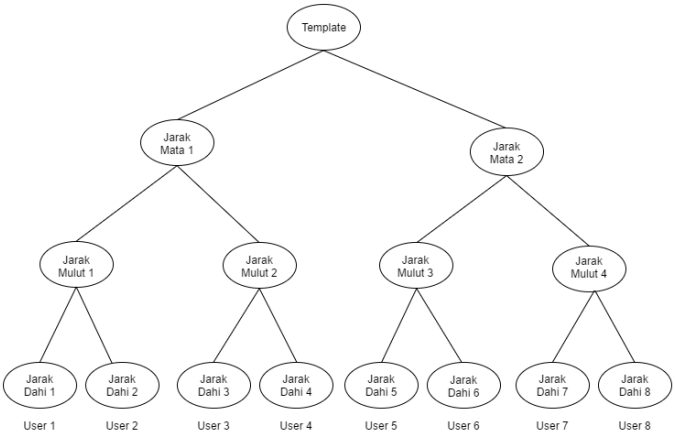
Pada makalah ini akan dianalisis tiga macam algoritma pencocokan *string* untuk memproses *tag suggestions* di Facebook. Pencarian kecocokan antara template pada database dan template yang sedang diproses pada foto dilakukan dengan gabungan antara algoritma BFS dan algoritma *string matching.*

Pertama, seluruh *template* wajah dari setiap pengguna yang berbeda disimpan dalam database *template*. Pengaksesan *database* dilakukan dengan algoritma *Breadth-First-Search* untuk mencari hasil kecocokan yang maksimal dengan cepat.

Misalkan pada *database* terdapat 8 orang pengguna, maka untuk setiap poin kunci pada wajahnya diciptakan sebuah kode unik. Misalkan 3 buah kode unik pada wajah setiap pengguna, yaitu:

1. Jarak antar mata
2. Jarak hidung dengan mulut
3. Jarak dahi

Dimana setiap pengguna dapat mempunyai beberapa kode yang sama namun tidak mungkin mempunyai seluruh kode yang sama dikarenakan pada wajah sesorang ada fitur-fitur wajah yang sama tetapi tidak mungkin dua orang yang berbeda mempunyai seluruh fitur wajah yang benar-benar sama. Gabungan tiga buah kode unik yang berbeda akan membentuk sebuah *template* unik bagi setiap user Sehingga dibentuklah pohon data dari *template* 8 orang pengguna tadi untuk mengilustrasikan pencarian data dengan menggunakan algortima BFS.

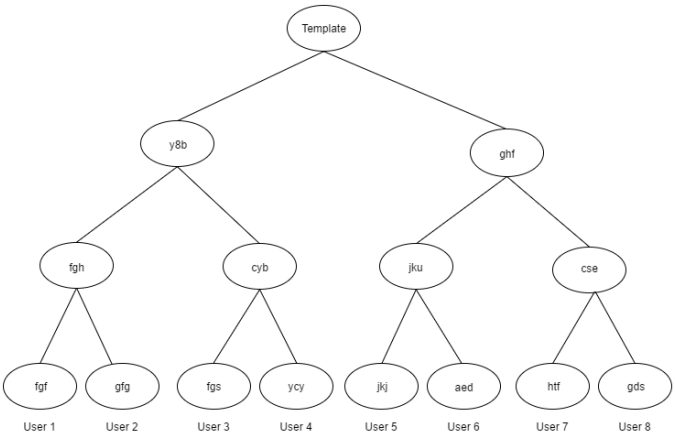


Gambar 8. Pohon *Template*

Lalu pada foto yang diunggah oleh pengguna dilakukan proses pemindaian wajah. Jika ada wajah manusia yang terdeteksi, maka dibuat sebuah *string* panjang berupa teks yang menggabungkan seluruh informasi poin-poin penting pada wajahnya yaitu tiga buah kode unik yang merupakan informasi penting tadi.

Misalkan *string* yang terbentuk setelah memindai foto adalah "cyby8bycy". *String* yang terbentuk ini dijadikan menjadi teks dan kode-kode unik dari setiap pengguna adalah *pattern* yang harus dicocokan dengan teks. Jika semua kode unik untuk satu pengguna ditemukan kecocokan untuk teks tersebut, maka ditemukan identitas pemilik dari wajah pada foto tersebut.

Kode-kode unik yang dijadikan *pattern* dapat dilihat pada pohon dibawah ini:



Gambar 9. Pohon *Template* dengan kode unik sebagai *pattern*

Pencarian dengan *Breadth-First-Search* dilakukan dengan proses pertama-tama adalah melakukan pengecekan yang dimulai dari membangkitkan node *Template*. Lalu dilakukan pengecekan tetangga-tetangga node *Template*. Bergerak ke node "y8b" dan mengalami kecocokan antara *pattern* dan teks, maka node "y8b" dibangkitkan. Lalu, bergerak ke tetangga node *Template* selanjutnya yaitu node "ghf", karena node "ghf" tidak mengalami kecocokan antara *pattern* dan teks maka node tersebut tidak dibangkitkan. Maka proses langsung berlanjut ke semua tetangga dari node "y8b". Lanjut ke node "fgh", karena tidak ditemukan kecocokan *string* maka node tersebut tidak dibangkitkan. Berlanjut pada node "cyb", karena ditemukan kecocokan *string* maka node tersebut dibangkitkan. Node selanjutnya adalah node "fgs" karena tidak ditemukan kecocokan maka berlanjut ke node "ycy". Pada node "ycy" ditemukan kecocokan antara *pattern* dan teks, maka kecocokan wajah telah ditemukan yaitu pengguna 4.

Selanjutnya akan dibahas bagaimana contoh pencocokan *pattern* yang berupa kode unik "y8b" dengan teks "cyby8bycy" dengan 3 buah algoritma pencocokan *string* yang berbeda:

1. Algoritma Brute Force

i	1	2	3	4	5	6	7	8	9
T[i]	c	y	b	y	8	b	y	c	y

j	1	2	3
P[j]	y	8	b

1

y	8	b

2 3

y	8	b

4

y	8	b

5 6 7

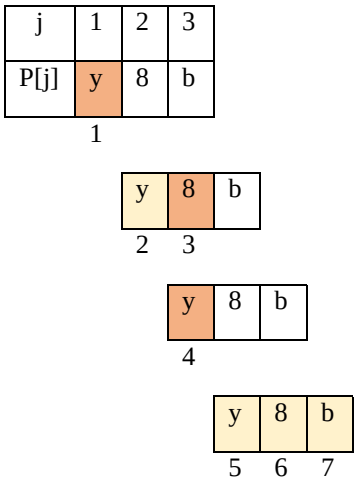
Pada algoritma *brute force*, terjadi 7 buah pencocokan karakter untuk menemukan kecocokan yang tepat antara *pattern* "y8b" dengan teks "cyby8bycy".

2. Algoritma Knuth-Morris-Pratt

Fungsi pinggiran untuk *pattern* "y8b":

k	1	2	3
P[k]	y	8	b
b[k]	0	0	0

i	1	2	3	4	5	6	7	8	9
T[i]	c	y	b	y	8	b	y	c	y



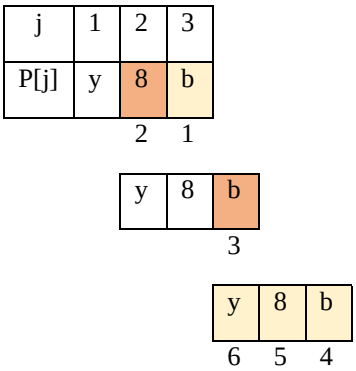
Sama seperti pada algoritma *brute force*, untuk algoritma *Knuth-Morris-Pratt* terjadi 7 buah pencocokan karakter untuk menemukan kecocokan yang tepat antara *pattern* "y8b" dengan teks "cyby8bycy".

3. Algoritma *Boyer-Moore*

Fungsi kemunculan terakhir untuk *pattern* "y8b":

x	y	b	8
bc(x)	1	3	2

i	1	2	3	4	5	6	7	8	9
T[i]	c	y	b	y	8	b	y	c	y



Pada algoritma *Boyer-Moore* terjadi 6 buah pencocokan karakter untuk menemukan kecocokan yang tepat antara *pattern* "y8b" dengan teks "cyby8bycy".

Membandingkan tiga buah proses pencocokan *string* di atas, algoritma *Boyer-Moore* merupakan algoritma yang paling mangkus untuk mencari kecocokan antara *pattern* "y8b" dengan teks "cyby8bycy".

V. Kesimpulan

Tag suggestions pada Facebook adalah suatu fitur yang sangat memudahkan pengguna media sosial tersebut dalam membagikan foto. Algoritma *Breadth-First-Search* dan algortima pencocokan *string* adalah gabungan yang cocok digunakan untuk fitur tag suggestions tersebut. Algoritma Breadth-First-Search digunakan untuk mengakses data secara cepat dan algoritma pencocokan *string* digunakan untuk mencocokan *template* pengguna yang berada pada database dengan *template* yang dihasilkan pada pindaian foto yang sedang diproses.

VI. Ucapan Terima Kasih

Penulis ingin mengucapkan terima kasih kepada Tuhan Yang Maha Esa, karena berkat, kasih, dan karunia-Nya penulis dapat menyelesaikan makalah ini dengan baik. Penulis ingin berterima kasih untuk orang tua serta keluarga dari penulis atas perhatian, dukungan, dan doa yang diberikan. Penulis juga ingin berterima kasih kepada dosen mata kuliah IF2211 Strategi Algoritma, Bapak Dr. Ir. Rinaldi Munir, M.T. dan Ibu Dr. Nur Ulfa Maulidevi, S.T., M.Sc. karena berkat bimbingannya selama ini penulis dapat menulis makalah ini.

Referensi

[1] Munir, Rinaldi. *Diktat Kuliah IF2211 Strategi Algoritma*. 2009. Bandung: Institut Teknologi Bandung.

[2] Davidson, Andrew. 240-301, *Computer Engineering Lab III (Software), Pattern Matching*. 2006.

[3] *How does Facebook suggests tag?*, diakses pada 6 Mei 2016, https://www.facebook.com/help/122175507864081

[4] *What information does Facebook use to tell that a photo looks like me and to suggest that friends tag me?*, diakses pada 6 Mei 2016, https://www.facebook.com/help/218540514842030

[5] *Fitur-Fitur Facebook.* Diakses pada 7 Mei 2016, http://www.facebook.com

Pernyataan

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

Bandung, 8 Mei 2016

Catherine Pricilla
13514004