Penerapan Algoritma Transversal pada Graf dan Algoritma Pencocokan *String* dalam Sistem Jual-Beli Tiket Bioskop

Scarletta Julia Yapfrine - 13514074
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13514074@std.stei.itb.ac.id

***Abstrak*—Menonton film merupakan hobi dan sarana hiburan bagi sebagian besar orang. Hobi tersebut biasanya dilakukan di bioskop untuk mendapatkan pengalaman menonton yang maksimal dibandingkan dengan televisi di rumah. Agar bisa menonton di bioskop, seseorang tentunya perlu membeli tiket untuk film yang ingin ditonton terlebih dahulu. Pembelian tiket oleh penonton tersebut perlu disimpan datanya oleh bioskop untuk banyak keperluan. Pengaksesan data tiket baik untuk dibeli maupun direkap dari sistem bioskop tersebut bisa dilakukan dengan mengimplementasikan algoritma transversal graf yaitu BFS (*Breadth First Search*) dan DFS (*Depth First Search*), serta algoritma pencocokan *string* yaitu KMP (*Knuth-Morris-Pratt*) dan BM (*Boyer-Moore*).**

***Kata Kunci*—BFS, bioskop, BM, DFS, KMP, sistem jual-beli tiket**

I. PENDAHULUAN

Salah satu kegiatan terpopuler untuk menghabiskan waktu luang adalah menonton film di bioskop. Bioskop memang memberikan keunggulan tersendiri sebagai salah satu sarana menonton film. Bioskop selalu dilengkapi dengan layar berkualitas tinggi dan *sound system* yang canggih. Ditambah lagi, tempat duduk yang sangat nyaman. Film-film terbaru juga kebanyakan selalu tayang perdana di bioskop.

Karena keunggulan-keunggulan tersebut, bioskop biasanya selalu ramai dikunjungi orang, terutama saat liburan atau malam hari. Setiap harinya, lebih dari ratusan orang yang mengantri di loket pembelian tiket. Saat ada film terkenal yang baru keluar, antrian tersebut biasanya menjadi sangat panjang sehingga ada orang-orang yang tidak kebagian tiket.

Bioskop-bioskop seperti XXI dan blitzmegaplex mulai memunculkan solusi dari permasalahan tersebut. Solusi yang ditawarkan adalah dengan adanya sistem pembelian tiket bioskop secara *online* di situs resmi masing-masing bioskop tersebut. Dengan membeli tiket secara *online*, pembeli bisa melakukannya dengan praktis. Ia hanya perlu membuat akun dan mengisi saldo terlebih dahulu. Setelahnya, tiket bisa dibeli dari rumah tanpa harus mengantri susah-susah. Pada saat membeli secara *online*, pembeli sudah memilih ingin menonton film apa, pukul berapa, di *theater* berapa, dan *seat* berapa.

Bioskop tentunya memerlukan sebuah sistem (*online*) untuk menangani dan menyimpan data-data seperti pembelian tiket film yang diputar, dan pembeli tiket. Data-data tersebut diperlukan untuk keperluan rekap dan dasar pengambilan keputusan ke depannya. Berdasarkan gambaran yang sudah diberikan sebelumnya, sistem tersebut tentunya menyimpan cukup banyak data. Untuk menggunakan sistem tersebut, diperlukan suatu cara pengaksesan data, misalnya saat pembeli ingin memilih film yang ingin ditonton atau pada saat manajer bioskop ingin meliihat data tiket terjual.

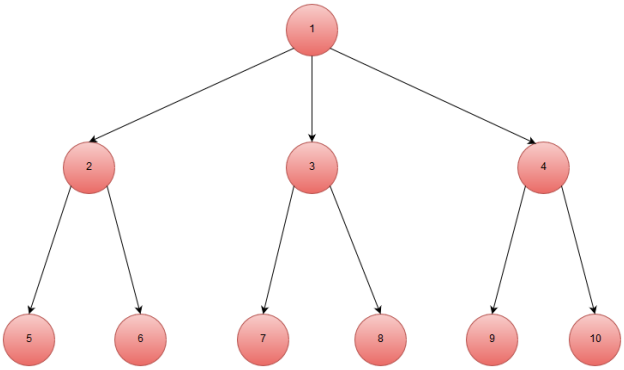
Pengaksesan data tersebut dapat dilakukan dengan memanfaatkan dan mengimplementasikan algoritma pencarian dan algoritma pencocokan *string*. *Breadth First Search* (BFS) dan *Depth First Search* (DFS) merupakan algoritma pencarian yang bisa digunakan. Algoritma pencocokan *string* yang bisa digunakan adalah algoritma *Knuth-Morris-Pratt* (KMP) dan algoritma *Boyer-Moore* (BM).

II. DASAR TEORI

2.1. *Breadth First Search* (BFS)

BFS adalah algoritma transversal pada graf secara melebar. BFS melakukan transversal pada graf dengan cara membangkitkan simpul pertama (simpul akar). Lalu, BFS membangkitkan simpul-simpul yang terhubung dengan simpul pertama. Misalnya S_0 adalah simpul pertama (simpul akar) yang terhubung dengan simpul-simpul S_1, S_2, S_3. Simpul pertama yang dibangkitkan adalah simpul S_0. Berikutnya, BFS membangkitkan S_1, S_2, dan S_3. Simpul yang selanjutnya dibangkitkan adalah simpul-simpul yang terhubung dengan S_1 dan belum dibangkitkan, lalu simpul-simpul yang terhubung dengan simpul S_2 dan belum dibangkitkan. Hal tersebut diulang hingga semua simpul sudah dibangkitkan atau pencarian sudah ditemukan.

Jika graf yang ditransversal berbentuk pohon, maka simpul-simpul pada aras yang sama dikunjungi terlebih dahulu baru mengunjungi simpul-simpul pada aras selanjutnya.



Gambar 1.1 Pohon untuk ditransversal secara BFS

Jika pohon di atas ditransversal secara BFS, maka urutan simpul yang dikunjungi adalah (1, 2, 3, 4, 5, 6, 7, 8, 9, 10).

Salah satu struktur data yang digunakan untuk mengimplementasikan algoritma BFS adalah antrian (*queue*). Berikut adalah langkah-langkah pemrosesan algoritma BFS dengan menggunakan struktur data antrian

a. Simpul akar dimasukkan ke dalam antrian *Q*. Lalu, diperiksa apakah simpul akar tersebut adalah solusi yang dicari (simpul solusi).

b. Jika simpul akar adalah simpul solusi, maka pencarian dihentikan.

c. Jika antrian kosong, berarti tidak ada solusi. Pencarian dihentikan.

d. Simpul yang ada pada antrian pertama dibangkitkan anak-anak simpulnya. Jika simpul tidak mempunyai anak, pencarian kembali ke langkah c. Jika ada, anak-anak simpul tersebut ditempatkan di belakang antrian sesuai dengan urutan dibangkitkannya.

e. Simpul yang ada pada antrian pertama dihapus.

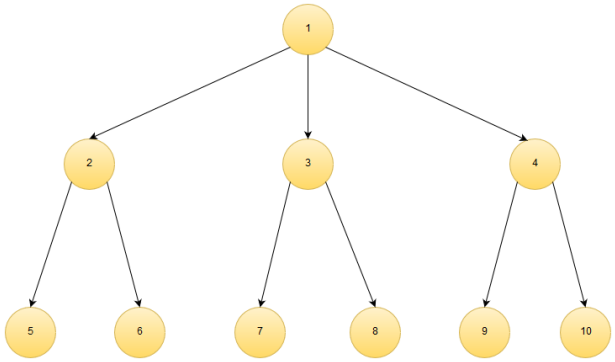
f. Jika salah satu simpul anak adalah simpul solusi, maka solusi ditemukan dan pencarian ditemukan.

g. Jika tidak, maka pencarian kembali ke langkah c.

2.2. *Depth First Search* (DFS)

DFS adalah algoritma transversal pada graf secara mendalam. DFS melakukan transversal pada graf dengan cara membangkitkan simpul pertama (simpul akar). Lalu, DFS membangkitkan satu simpul yang terhubung dengan simpul akar. Misalnya S_0 adalah simpul pertama (simpul akar) yang terhubung dengan simpul-simpul S_1, S_2, S_3. Simpul S_1 kemudian terhubung dengan simpul S_4 dan S_5. Simpul pertama yang dibangkitkan adalah simpul S_0, lalu DFS membangkitkan simpul S_1. Selanjutnya, DFS membangkitkan simpul S_4. Karena simpul yang terhubung dengan S_4 sudah dibangkitkan semua, simpul yang selanjutnya dibangkitkan adalah simpul S_5. Baru kemudian, DFS membangkitkan S_2 lalu S_3.

Dengan kata lain, algoritma DFS pertama membangkitkan simpul akar *v*, lalu, simpul *w* yang bertetangga dengan simpul *v* dikunjungi. DFS kemudian diulangi mulai dari simpul *w*. Ketika mencapai simpul *u* sedemikian sehingga semua simpul yang bertetangga dengannya telah dikunjungi, pencarian dirunut-balik ke simpul terakhir yang dikunjungi sebelumnya dan mempunyai simpul *w* yang belum dikunjungi. Pencarian berakhir jika tidak ada lagi simpul yang belum dikunjungi yang dapat dicapai dari simpul yang telah dikunjungi.



Gambar 1.2 Pohon untuk ditransversal secara DFS

Jika pohon di atas ditransversal secara DFS, maka urutan simpul yang dikunjungi adalah (1, 2, 5, 6, 3, 7, 8, 4, 9, 10).

Salah satu struktur data yang digunakan untuk mengimplementasikan algoritma DFS adalah tumpukan (*stack*). Berikut adalah langkah-langkah pemrosesan algoritma DFS dengan menggunakan struktur data tumpukan.

a. Simpul akar dimasukkan ke dalam tumpukan *S*. Lalu, diperiksa apakah simpul akar tersebut adalah solusi yang dicari (simpul solusi).

b. Jika simpul akar adalah simpul solusi, maka pencarian dihentikan.

c. Jika tumpukan kosong, berarti tidak ada solusi. Pencarian dihentikan.

d. Jika simpul pada tumpukan teratas mempunyai anak, maka simpul yang ada pada tumpukan teratas disimpan, lalu dihapus dari tumpukan.

e. Jika simpul pada tumpukan teratas tidak mempunyai anak, maka simpul dihapus dari tumpukan lalu kembali ke langkah c.

f. Simpul yang disimpan kemudian dibangkitkan anak-anak simpulnya. Anak-anak simpul tersebut ditempatkan ke atas tumpukan sesuai dengan urutan dibangkitkannya.

g. Jika salah satu simpul anak adalah simpul solusi, maka solusi ditemukan dan pencarian ditemukan.

h. Jika tidak, maka pencarian kembali ke langkah c.

2.3. *Knuth-Morris-Pratt*

Algoritma KMP dikembangkan oleh D.E. Knuth, J. H. Morris, dan V. R. Pratt. Pada algoritma KMP, informasi dipelihara untuk melakukan jumlah pergeseran. Algoritma KMP memelihara informasi tersebut untuk membuat pergeseran yang banyak dan lebih pintar, tidak hanya per satu karakter seperti pada algoritma *brute force*. Waktu pencarian karakter dengan menggunakan algoritma KMP berkurang secara signifikan jika dibandingkan dengan *brute force*. Algoritma KMP memroses teks dari kiri ke kanan.

Penentuan pergeseran dari algoritma KMP ini kuncinya terletak pada fungsi pinggiran (*border function*). Fungsi pinggiran hanya bergantung pada karakter-karakter *pattern*. Fungsi pingiran *b*(*j*) merupakan ukuran awalan terpanjang dari suatu *pattern P* yang merupakan akhiran dari *P*[i..j]. Misalkan suatu *pattern P* = ababaa, maka nilai dari *b* untuk masing-masing karakter pada *pattern P* adalah sebagai berikut

j	1	2	3	4	5	6
P[*j*]	A	b	a	b	a	a
b(*j*)	0	0	1	2	3	1

Tabel 2.1 Fungsi Pinggiran dari “ababaa”

Saat terjadi ketidakcocokkan antara karakter pada teks dan karakter pada *pattern* yang sedang dibandingkan, maka dilakukan pergeseran. Banyaknya pergeseran yang dilakukan saat terjadi ketidakcocokkan adalah

$$N = l - b(x)$$

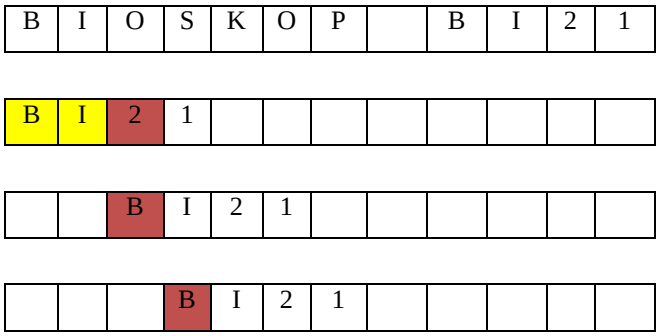
N = banyaknya pergeseran

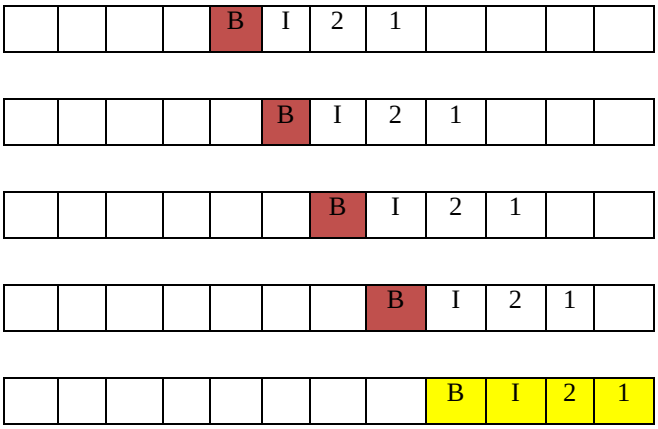
l = awalan dari *pattern* yang bersesuaian dengan teks

b = nilai fungsi pinggiran dari x

x = panjang awalan *pattern* yang bersesuaian

Berikut adalah contoh pencocokan *string* dengan KMP





2.4. *Boyer-Moore*

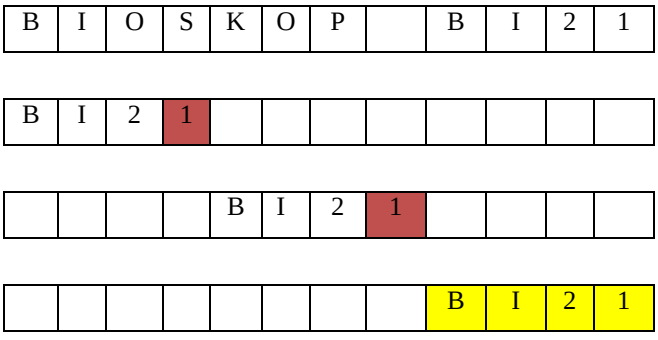
Algoritma Boyer-Moore mencocokkan pola dengan teks dimulai dari akhir pola (dari kanan ke kiri). Jika terjadi ketidakcocokkan karakter, maka ada tiga kemungkinan pergeseran yang dilakukan. Misalkan ketidakcocokkan terjadi pada *T*[*i*] = *x*, dengan *T* adalah teks. Karakter pada pola *P*[*j*] tidak sama dengan *T*[*i*]. Kasus yang pertama adalah jika *P* mengandung *x* pada indeks yang lebih kecil daripada *j*, maka *P* digeser ke kanan untuk menyesuaikan indeks *x* tersebut dengan *T*[*i*]. Kasus yang kedua adalah jika *P* mengandung *x* pada ideks yang lebih besar daripada *j*, maka *P* digeser ke kanan sebanyak satu karakter. Kasus ketiga adalah jika *P* tidak mengandung *x*, maka *P* digeser ke kanan untuk menyesuaikan *P*[1] dengan *T*[i+1].

Algoritma Boyer-Moore melakukan pemrosesan awal terhadap *pattern P* untuk membuat fungsi kemunculan terakhir, *L*(*x*). *L*(*x*) menghasilkan indeks kemunculan terakhir karakter pada *pattern*. Jika suatu karakter *x* tidak ada pada *pattern P*, maka *L*(*x*) = -1. Misalkan suatu *pattern P* = “abacab”, maka fungsi kemunculan terakhirnya adalah sebagai berikut

x	a	B	c	D
L(*x*)	5	6	4	-1

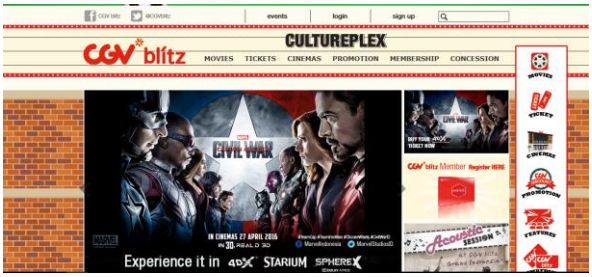
Tabel 2.2 Fungsi Kemunculan Terakhir dari “abacab”

Berikut adalah contoh pencocokan *string* dengan BM



III. GAMBARAN SISTEM JUAL-BELI TIKET BIOSKOP

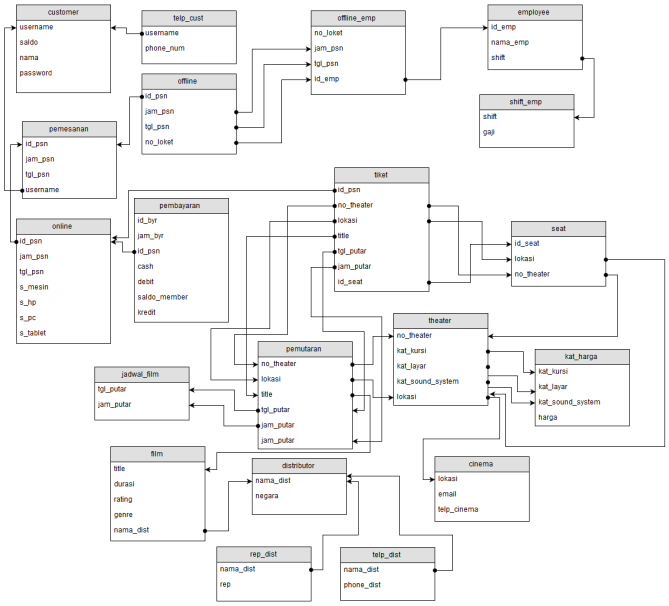
Untuk menonton film di bioskop, seseorang harus membeli tiket terlebih dahulu di loket pada bioskop. Hampir semua bioskop terkemuka di Indonesia juga menyediakan sarana pembelian tiket secara *online*. Contoh dari bioskop yang menyediakan sarana pembelian tiket secara *online* adalah CGV blitz, XXI, dan cinemaxx.



Gambar 3.1 Tampilan Situs CGV Blitz

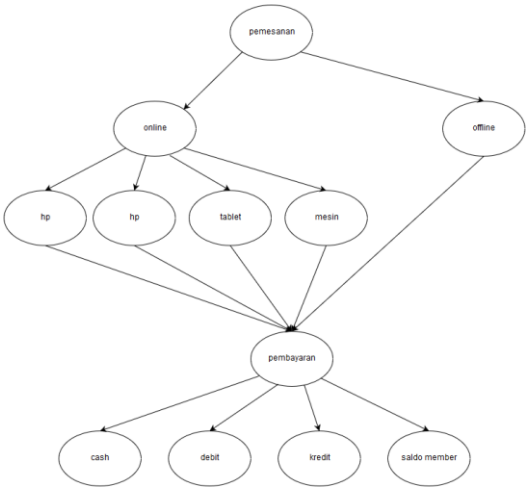
Sumber : https://www.cgvblitz.com/en/

Sistem jual-beli tiket bioskop tersebut pasti mengandung basis data yang cukup kompleks. Basis data dari sistem tersebut menghubungkan banyak komponen seperti pemesan, tiket yang dipesan, film yang sedang main, film yang dipesan, daftar lokasi bioskopnya di seluruh Indonesia, dan masih banyak lagi. Jika dibuat diagram relasi basis data dari sistem bioskop tersebut secara sederhana, kurang lebih akan menjadi seperti ini



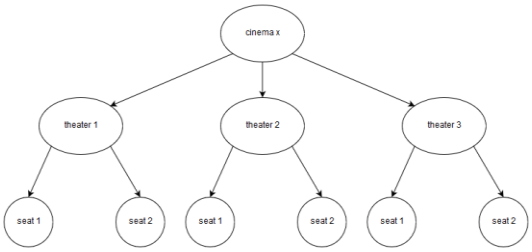
Gambar 3.2 Skema Basis Data Bioskop

Salah satu struktur data yang bisa digunakan untuk memroses data seperti pada diagram relasi di atas adalah struktur data pohon. Dari data yang disimpan, dibuat pohon untuk mempermudah pengaksesan data. Misalnya manajer bioskop ingin mengakses data cara pembayaran dari pembeli tiket. Untuk pengaksesan tersebut, maka dibentuk pohon seperti ini



Gambar 3.3 Pohon untuk Relasi Pemesanan dan Pembayaran

Pohon bisa dibentuk setiap ada pengaksesan data. Pohon yang dibentuk bisa berbeda bergantung pada data yang ingin diakses. Meskipun berbeda, pohon yang dibentuk tetap berdasarkan keterhubungan antar data pada basis data. Dari relasi basis data, misalnya diambil relasi "cinema". Relasi "cinema" terhubung dengan "theater", lalu relasi "theater" terhubung dengan "seat". Dari relasi itu, jika pengguna sistem seperti pembeli tiket ingin memilih *seat*, maka bisa dibentuk pohon pencarian seperti ini



Gambar 3.4 Pohon untuk Relasi Cinema, Theater, dan Seat

IV. PENERAPAN ALGORITMA BFS, DFS, KMP, DAN BM DALAM PENGAKSESAN DATA

Pengaksesan data pada sistem jual-beli bioskop sesungguhnya bisa dilakukan dengan menggunakan *query* yang ada pada bahasa pemrograman SQL. Untuk melakukan hal ini, setiap relasi perlu diberi atribut-atribut masing-masing (sesuai dengan skema relasi yang ada pada bagian 3). Contohnya, jika manajer bioskop ingin mengakses daftar orang-orang yang pernah melakukan pembelian tiket beserta *username*, nama, dan banyak saldo yang dimilikinya, maka *query* yang dimasukkan adalah

select username, nama, saldo from customer;

Hasil yang diberikan dari *query* tersebut kurang lebih berbentuk seperti ini

```
mysql> select username, nama, saldo from customer limit 15;
+-----------------+----------------------+--------+
| username        | nama                 | saldo  |
+-----------------+----------------------+--------+
| A12345678       | Juanika Joanda       |  20000 |
| afifahresti     | Afifah Resti         |  65000 |
| allenius        | Thomas Allen         |  45000 |
| AmirCokro       | Cokro Amirullah      |  10000 |
| AmirHasudungan  | Amir Hasudungan      |  60000 |
| Anananananan    | Jeremy Tan           |  85000 |
| andikaputra     | Andika Putra         |  35000 |
| AngkasaBintang  | Rey Angkasa          | 200000 |
| AntonHasudungan | Anton Hasudugan      |  55000 |
| apl1234567      | Ariadne Puan Lestari |  10000 |
| AS2005          | Arsyad Sitompul      |  30000 |
| Atang6753       | Tatang Suteja        | 100000 |
| BapakBijak      | Jono Soejono         |  60000 |
| BarryWoniogoO   | Barry Woniogo O      |  35000 |
| beautyandbeast  | Gaston Wicked        | 200000 |
+-----------------+----------------------+--------+
```

Gambar 4.1 Hasil *Query* untuk Menampilkan Data Pembeli

Query juga bisa digunakan untuk mengakses cara pembayaran yang dilakukan oleh manajer untuk melihat transaksi pembayaran yang dilakukan oleh pembeli, baik pembeli yang melakukan pemesanan secara *online* maupun pembeli yang melakukan pemesanan secara *offline*. Dari skema relasi yang ada pada bagian 3, bisa dilihat bahwa ada tabel pembayaran, serta table *online* dan *offline*. Tabel pembayaran merupakan tabel yang menyimpan semua data pembayaran yang dilakukan pembeli, sementara tabel *online* menyimpan data pemesanan yang dilakukan secara *online*, dan tabel *offline* menyimpan data pemesanan yang dilakukan secara *offline*. Jika manajer ingin mengakses data pembayaran yang dilakukan pembeli yang memesan secara *offline*, maka *query* yang dimasukkan adalah

select * from pembayaran where id_psn in (select id_psn from offline);

Hasil yang diberikan *query* tersebut kurang lebih seperti ini

```
mysql> select * from pembayaran where id_psn in (select id_psn from offline);
+--------+----------+--------+------+-------+-------------+--------+
| id_byr | jam_byr  | id_psn | cash | debit | saldo_member | kredit |
+--------+----------+--------+------+-------+-------------+--------+
| byr021 | 10:08:12 | psn021 | yes  | no    | no          | no     |
| byr120 | 10:09:07 | psn120 | no   | yes   | no          | no     |
| byr080 | 10:45:18 | psn080 | yes  | no    | no          | no     |
| byr058 | 13:59:32 | psn058 | no   | no    | no          | yes    |
| byr012 | 14:54:09 | psn012 | no   | no    | no          | yes    |
| byr089 | 16:08:59 | psn089 | no   | no    | no          | no     |
| byr065 | 16:09:01 | psn065 | no   | yes   | no          | no     |
| byr066 | 16:16:38 | psn066 | yes  | no    | no          | no     |
| byr068 | 16:40:14 | psn068 | yes  | no    | no          | no     |
| byr069 | 17:05:45 | psn069 | no   | yes   | no          | no     |
| byr070 | 17:09:54 | psn070 | no   | yes   | no          | no     |
| byr045 | 17:12:46 | psn045 | no   | no    | no          | yes    |
| byr095 | 20:13:54 | psn095 | yes  | no    | no          | no     |
| byr097 | 10:09:08 | psn097 | no   | no    | no          | yes    |
| byr110 | 11:39:28 | psn110 | yes  | no    | no          | no     |
| byr060 | 14:09:33 | psn060 | no   | no    | no          | yes    |
| byr088 | 15:01:09 | psn088 | no   | no    | no          | yes    |
| byr100 | 15:09:03 | psn100 | yes  | no    | no          | no     |
| byr063 | 15:43:27 | psn063 | yes  | no    | no          | no     |
| byr030 | 15:54:00 | psn030 | yes  | no    | no          | no     |
| byr067 | 16:34:12 | psn067 | yes  | no    | no          | no     |
| byr125 | 16:58:12 | psn125 | yes  | no    | no          | no     |
| byr071 | 17:29:30 | psn071 | yes  | no    | no          | no     |
| byr008 | 17:37:07 | psn008 | no   | yes   | no          | no     |
| byr073 | 17:45:09 | psn073 | no   | no    | no          | yes    |
| byr047 | 19:12:46 | psn047 | no   | yes   | no          | no     |
+--------+----------+--------+------+-------+-------------+--------+
```

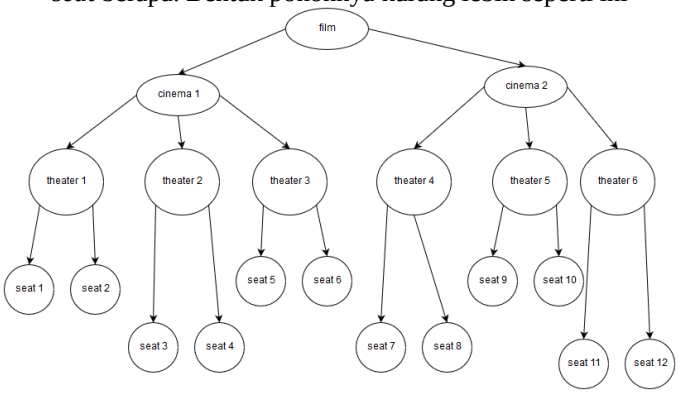
Gambar 4.2 Hasil *Query* untuk Menampilkan Data Pembayaran dari Pembeli yang Memesan secara *Online*

Query di atas menggunakan *nested query* untuk mengambil data pemesanan secara *offline*, lalu *query* mencocokkan id_psn (atribut penghubung atau *foreign key*) dari tabel pembayaran dan tabel pemesanan.

Namun, *query* lebih cocok digunakan oleh pihak manajemen yang mengakses data untuk keperluan rekap dan pengambilan keputusan ke depannya. Untuk pengaksesan data oleh pembeli, maka *query* kurang cocok dengan tampilan seperti itu. Pengaksesan data oleh pembeli biasanya dilakukan di aplikasi yang memiliki *user interface* yang menarik. Di balik *user interface* tersebut, tentunya ada algoritma yang diimplementasikan. Algoritma yang bisa diimplementasiikan untuk hal ini adalah algoritma BFS, DFS, KMP, dan BM. Struktur data yang digunakan untuk menyimpan data adalah struktur data pohon.

4.1. Penerapan algoritma BFS

Misalkan pembeli ingin membeli tiket untuk menonton film X yang diputar di beberapa bioskop (*cinema*). Masing-masing *cinema* terdiri dari tiga *theater*, dan masing-masing *theater* terdiri dari banyak *seat*. Untuk bisa membeli tiket, tentunya pembeli harus memilih untuk menonton film apa, di bioskop apa, *theater* berapa, dan *seat* berapa. Bentuk pohonnya kurang lebih seperti ini



Gambar 4.3 Pohon untuk Data Film, Cinema, Theater, dan Seat

Struktur data yang digunakan algoritma BFS adalah struktur data antrian yang menyimpan *path* (simpul yang dilalui dari akar hingga daun). Misalkan pembeli sudah memilih film dan ingin menonton di cinema 2, theater 5, seat 9. Algoritma BFS akan membangkitkan simpul-simpul dengan urutan

a. Film yang dipilih dibangkitkan pertama, lalu dimasukkan ke dalam antrian.
b. Simpul anak dari simpul film ada dua, yaitu cinema 1 dan cinema 2. Film disalin jadi dua sesuai dengan banyak anaknya. Lalu, cinema 1 dan cinema 2 dimasukkan ke antrian yang telah berisi film. Isi antrian = {(film, cinema 1), (film, cinema 2)}
c. Isi pertama pada antrian, yaitu (film, cinema 1) mempunyai 2 anak, maka (film, cinema 1) dihapus dari antrian. Lalu, di bagian belakang antrian ditambahkan (film, cinema 1, theater 1), (film, cinema 1, theater 2), (film, cinema 1 , theater 3).

d. (film, cinema 1) dihapus dari antrian.
e. Langkah c diulangi untuk (film, cinema 2).
f. Antrian terdepan sekarang adalah (film, cinema 1, theater 1).
g. (film, cinema 1, theater 1) memiliki dua anak, maka (film, cinema 1, theater 1, seat 1) dan (film, cinema 1, theater 1, seat 2) ditambahkan ke antrian.
h. (film, cinema 1, theater 1) dihapus dari antrian.
i. (film, cinema 1, theater 2, seat 3) dan (film, cinema 1, theater 2, seat 4) ditambahkan ke antrian.
j. (film, cinema 1, theater 2) dihapus dari antrian.
k. (film, cinema 1, theater 3, seat 5) dan (film, cinema 1, theater 3, seat 6) ditambahkan ke antrian.
l. (film, cinema 1, theater 3) dihapus dari antrian.
m. (film, cinema 2, theater 4, seat 7) dan (film, cinema 2, theater 4, seat 8) ditambahkan ke antrian.
n. (film, cinema 2, theater 4) dihapus dari antrian.
o. (film, cinema 2, theater 5, seat 9) ditambahkan ke antrian. Solusi ketemu. Pencarian dihentikan.

4.2. Penerapan algoritma DFS

Selain algoritma BFS, algoritma DFS juga bisa digunakan dalam pengaksesan data. Struktur data yang digunakan pada algoritma DFS ini adalah tumpukan (*stack*). Urutan pembangkitan oleh algoritma DFS untuk kasus yang sama seperti pada BFS adalah

a. Film yang dipilih dimasukkan ke dalam tumpukan. Film disimpan untuk dibangkitkan anak-anaknya, lalu dihapus dari tumpukan.
b. (film, cinema 1) dan (film, cinema 2) dimasukkan ke dalam tumpukan sesuai dengan urutan pembangkitannya.
c. (film, cinema 2) disimpan, lalu dihapus dari tumpukan.
d. (film, cinema 2, theater 4), (film, cinema 2, theater 5), dan (film, cinema 2, theater 6) dimasukkan ke bagian atas tumpukan.
e. (film, cinema 2, theater 6) disimpan, lalu dihapus dari tumpukan.
f. (film, cinema 2, theater 6, seat 11) dan (film, cinema 2, theater 6, seat 12) dimasukan ke bagian atas tumpukan.
g. (film, cinema 2, theater 6, seat 12) dihapus dari tumpukan.
h. (film, cinema 2, theater 6, seat 11) dihapus dari tumpukan.
i. (film, cinema 2, theater 5) disimpan, lalu dihapus dari tumpukan.
j. (film, cinema 2, theater 5, seat 9) dan (film, cinema 2, theater 5, seat 10) dimasukkan ke bagian atas tumpukan.
k. (film, cinema 2, theater 5, seat 10) dihapus dari tumpukan.
l. (film, cinema 2, theater 9, seat 9) merupakan solusi yang dicari. Pencarian dihentikan.

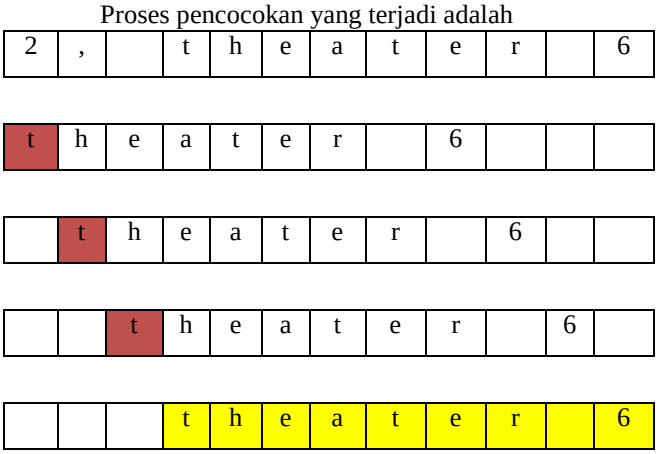
4.3. Penerapan algoritma KMP

Algoritma KMP digunakan pada saat mencocokkan masukan dari pembeli dengan isi dari simpul pohon. Sebagai contoh, misalnya masukan dari pengguna adalah *string* “theater 6”. Fungsi pinggirannya adalah

j	1	2	3	4	5	6	7	8	9
P[*j*]	t	h	e	a	t	e	r		6
b(*j*)	0	0	0	0	1	0	0	0	0

Tabel 4.1 Fungsi Pinggiran untuk “theater 9”

Proses pencocokan yang terjadi adalah



String sesungguhnya yang akan dicocokkan adalah *string* berbentuk *tuple* masukan pengguna, contohnya (film, cinema 1, theater 1, seat 1). Ilustrasi di atas hanya memberikan gambaran bagaimana KMP bekerja pada sistem bioskop ini, namun bukan gambaran persis.

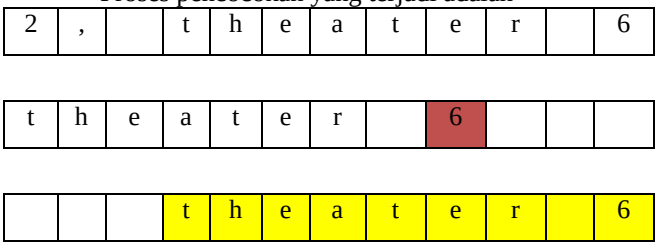
4.4. Penerapan algoritma BM

Algoritma BM digunakan pada saat mencocokkan masukan dari pembeli dengan isi dari simpul pohon. Sebagai contoh, misalnya diambil sebagian masukan dari pengguna adalah *string* “theater 9”. Fungsi kemunculan terakhirnya adalah

x	d	i	t	h	e	a	r		6
L(*x*)	-1	-1	5	2	6	4	7	8	9

Tabel 4.2 Fungsi Kemunculan Terakhir untuk “theater 9”

Proses pencocokan yang terjadi adalah



String sesungguhnya yang akan dicocokkan adalah *string* berbentuk *tuple* masukan pengguna, contohnya (film, cinema 1, theater 1, seat 1). Ilustrasi di atas hanya memberikan gambaran bagaimana KMP bekerja pada sistem bioskop ini, namun bukan gambaran persis.

V. KESIMPULAN

Algoritma transversal pada graf berupa *Breadth First Search* (BFS) dan *Depth First Search* (DFS) berguna dalam pengaksesan data pada sistem jual-beli tiket bioskop. Jika struktur data yang digunakan pada penyimpanan data bioskop adalah pohon, maka algoritma BFS bisa menggunakan struktur data antrian (*queue*) untuk melakukan transversal pada pohon sehingga ditemukan solusi sesuai masukan pengguna. Algoritma DFS bisa menggunakan struktur data tumpukan (*stack*) untuk melakukan transversal pada pohon. Algoritma pencocokan *string* berguna untuk membantu algoritma transversal pada graf untuk mencocokkan apakah data yang disimpan pada suatu simpul sama dengan yang ingin diakses. Algoritma pencocokan *string* yang digunakan adalah algoritma *Knuth-Morris-Pratt* (KMP) dan algoritma *Boyer-Moore* (BM). Algoritma KMP mencocokkan karakter pada *pattern* dari kiri ke kanan dan menggunakan fungsi pinggiran untuk melakukan pergeseran. Algoritma BM mencocokkan karakter pada *pattern* dari kanan ke kiri dan menggunakan fungsi kemunculan terakhir untuk melakukan pergeseran.

VI. UCAPAN TERIMA KASIH

Pertama-tama, penulis ingin menyampaikan puji syukur kepada Tuhan Yang Maha Esa atas rahmat dan berkat-Nya sehingga penulis bisa menyelesaikan makalah ini. Penulis juga ingin menyampaikan terima kasih kepada Dr. Ir. Rinaldi Munir, M.T. dan Dr. Nur Ulfa Maulidevi, ST., M.Sc. sebagai Dosen Mata Kuliah Strategi Algoritma yang telah membimbing penulis.

DAFTAR PUSTAKA

[1] Davidson, Andrew. 240-301, Computer Engineering Lab III (Software), Semester 1, 2006-2007

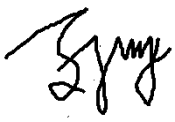
[2] http://www.cinemaxxtheater.com/, diakses pada 6 Mei 2016

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

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, 6 Mei 2016



Scarletta Julia Yapfrine – 13514074

LAMPIRAN

Gambar Skema Relasi Bioskop dalam Ukuran Lebih Besar