Penerapan Algoritma *Boyer Moore* dan *Branch and Bound* Pada Optimasi Peletakan Tetromino Permainan Tetris Sederhana

Husnia Munzayana - 13521077
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13521077@std.stei.itb.ac.id

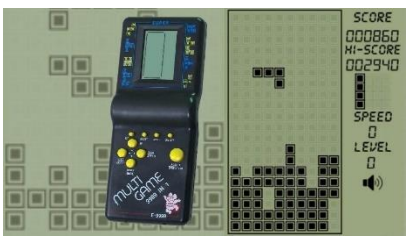
***Abstract*— Permainan Tetris menjadi salah satu permainan video *puzzle* yang paling banyak dimainkan khalayak umum di seluruh dunia. Tujuan utama dalam permainan ini adalah menciptakan baris penuh tanpa celah dengan meletakkan bentuk geometris yang disebut tetromino secara strategis. Dalam penelitian ini, kami menerapkan algoritma *Boyer Moore* dan algoritma *Branch and Bound* untuk mengoptimalkan peletakan tetromino dalam permainan Tetris sederhana. Algoritma *Boyer Moore* digunakan untuk mencocokkan pola tetromino yang sudah ada di area permainan, sedangkan algoritma *Branch and Bound* digunakan untuk mencari solusi optimal dalam menempatkan tetromino baru. Untuk lebih memahami bagaimana algoritma boyer-moore dan algoritma *branch and bound* dapat diterapkan untuk melakukan optimasi peletakan tetromino pada permainan tetris, akan dilakukan studi lebih mendalam pada makalah ini.**

***Keywords*— Algoritma Boyer-Moore, Algoritma Branch and Bound, Tetris, Peletakan Tetromino.**

I. PENDAHULUAN

Permainan Tetris adalah permainan teka-teki video *puzzle* yang terkenal hampir di seluruh dunia. Dalam permainan ini, pemain harus menempatkan tetromino, yaitu bentuk geometris yang terdiri dari empat kotak kecil, secara strategis di area permainan untuk membentuk baris horizontal penuh. Tujuan utama permainan Tetris adalah menciptakan baris penuh tanpa celah dan pemain akan mendapatkan poin.

Pada awalnya, tetris diciptakan sebagai alat untuk mengukur performa komputer. Namun, karena *gameplay*-nya yang sederhana, permainan ini mulai digemari khalayak umum dan hingga saat ini telah dikembangkan di berbagai platform, baik platform digital dalam bentuk aplikasi permainan maupun permainan pada *gimbot* klasik.



Gambar 1.1 Permainan Tetris pada Gimbot Klasik
(Sumber : https://www.youtube.com/watch?v=B6btt-g70XY)

Dalam pengembangan permainan Tetris, salah satu aspek penting yang dapat dioptimalkan adalah peletakan tetromino. Pemilihan lokasi dan orientasi yang tepat untuk tetromino dapat meningkatkan efisiensi permainan dan memungkinkan pemain untuk mencapai skor tertinggi.

Dalam makalah ini, akan dibahas secara detail penerapan algoritma *Boyer-Moore* dan algoritma *Branch and Bound* untuk mengoptimalkan peletakan tetromino dalam permainan Tetris sederhana. Algoritma *Boyer-Moore* akan digunakan untuk mencocokkan pola tetromino yang sudah ada dalam area permainan, sedangkan algoritma *Branch and Bound* akan digunakan untuk mencari solusi optimal dalam menempatkan tetromino baru.

II. TEORI DASAR

A. Algoritma Branch and Bound

Algoritma *Branch and Bound* merupakan salah satu metode dalam pemecahan masalah optimasi. Permasalahan tersebut biasanya memiliki kompleksitas waktu eksponensial dalam penjelajahan semua kemungkinan solusinya pada kasus terburuk. Algoritma ini pertama kali diperkenalkan oleh Ailsa Land dan Alan J. Perlis sekitar tahun 1960. Tujuan utama algoritma ini adalah mencari solusi optimal dengan cara memecah masalah menjadi sub masalah yang lebih kecil dan membatasi penelusuran hanya pada *sub-problem* yang berpotensi menghasilkan solusi yang lebih baik.

Langkah-langkah umum algoritma *Branch and Bound* meliputi:

1. **Inisialisasi**: Definisikan properti algoritma *branch and bound*, seperti simpul hidup/*expand-node*, fungsi estimasi nilai cost, *bounding function*, dan simpul solusi
2. **Pembangkitan Pohon Pencarian**: Bangunlah sebuah pohon pencarian yang mencakup semua kemungkinan solusi mulai dari simpul akar.
3. **Evaluasi Fungsi Pembatas**: Pada setiap simpul pohon pencarian, lakukan evaluasi nilai *cost* dengan *bounding function* yang telah didefinisikan sebelumnya.

4. **Pemotongan atau *pruning***: Gunakan aturan *pruning* untuk mengurangi jumlah simpul yang perlu dieksplorasi. Jika telah ditemukan solusi optimum sementara, maka simpul dengan nilai *cost* yang lebih besar dapat di *pruning*.
5. **Penelusuran dan Pembaruan**: Lakukan penelusuran pada simpul yang belum dipotong. Saat mengeksplorasi simpul, perbarui nilai *cost* paling optimum jika ditemukan solusi yang lebih baik.
6. Lakukan secara **rekursif** hingga tidak ada lagi simpul yang dapat di *expand*.
7. Solusi optimal didapatkan jika seluruh pohon pencarian telah dieksplorasi, solusi optimal adalah solusi dengan nilai batas atas terbaik yang telah ditemukan.

Kompleksitas algoritma *Branch and Bound* tergantung pada kompleksitas pemecahan sub masalah. Dalam beberapa kasus, algoritma ini dapat memberikan solusi optimal dengan kompleksitas waktu eksponensial, tetapi dalam beberapa kasus lainnya, kompleksitas dapat diatasi dengan menggunakan teknik *pruning* yang efisien.

B. Pattern Matching

Pattern Matching adalah proses pencarian pola secara algoritmis dalam suatu data yang diberikan. Terdapat dua istilah dalam *pattern matching*, yaitu:

a. T: Teks (*text*), yaitu (long) *string* yang panjangnya n karakter

b. P: *Pattern*, yaitu *string* dengan panjang m karakter (asumsi m <<< n) yang akan dicari di dalam teks. [4]

Pada algoritma *pattern matching*, akan dilakukan pencarian posisi pertama pola pada *text* yang *exact match* dengan *pattern.* Ada beberapa pendekatan yang dapat digunakan dalam pattern matching, termasuk *brute force*, algoritma *Knuth-Morris-Pratt* (KMP), algoritma *Boyer-Moore*, dan sebagainya. Setiap pendekatan memiliki kelebihan dan kelemahan tersendiri tergantung pada karakteristik masalah dan kompleksitasnya.

C. Algoritma Boyer-Moore

Algoritma Boyer-Moore adalah algoritma pencocokan string yang efisien, yang dikembangkan oleh Robert S. Boyer dan J Strother Moore pada tahun 1977. Algoritma ini digunakan untuk mencari kemunculan pola dalam teks dengan memanfaatkan konsep pemindahan karakter dan heuristis dengan *last occurrence character*. Dalam implementasinya, algoritma *Boyer-Moore* didasarkan pada dua teknik:

a. *The looking-glass technique*
   Looking-glass technique adalah teknik pergeseran *pattern* secara mundur atau bergerak ke arah belakang ketika pencarian pattern pada teks dilakukan.
b. *The character-jump technique*
   Character-jump technique merupakan teknik pergeseran *pattern* yang akan dilakukan saat terjadi ketidakcocokan pola (*mismatch*) antara *pattern* dengan *text.*

Langkah-langkah umum algoritma *Boyer-Moore* meliputi:

1. *Pre-processing*: Tentukan *Last Occurrence Table* yang memetakan posisi kemunculan terakhir setiap karakter *text* pada *pattern*. Jika karakter tidak ditemukan tuliskan -1.
2. Penempatan *pattern*: Letakkan *pattern* pada awal teks sehingga T[0] sejajar dengan P[0].
3. Mulai pencocokan dari karakter terakhir *pattern* kemudian maju, bandingkan dengan karakter pada *text* yang bersesuaian
4. *Looking-glass technique*: Pergeseran *pattern* secara mundur atau bergerak ke arah belakang ketika pencarian pattern pada teks dilakukan.
5. *Character Jump*: Jika ditemukan *mismatch*, periksa *last occurrence table* untuk menentukan banyak pergeseran. Terdapat 3 kasus dalam penentuan banyak pergeseran jika
   $$P[j] \neq T[i] = x$$
   Kasus 1: Jika pada *pattern* P terdapat x di suatu tempat, maka coba geser *pattern* P ke kanan untuk menyejajarkan *last occurrence* x di P dengan T[i]. Pada kasus ini, kita dapat memanfaatkan *last occurrence table* untuk menentukan banyak pergeseran.
   Kasus 2: Jika pada *pattern* P terdapat x di suatu tempat, tetapi posisi x berada di kanan P[j], maka geser P ke kanan sebanyak 1 karakter ke T[i+1].
   Kasus 3: Jika kasus 1 dan 2 tidak berlaku, maka geser P untuk menyejajarkan P[0] dengan T[i+1].
6. Lakukan secara *rekursif* hingga ditemukan pola yang *exact match* atau pencocokan telah sampai di akhir *text*.

Dengan pendekatan ini, *Boyer-Moore* dapat mengurangi jumlah perbandingan yang perlu dilakukan, sehingga dapat mempercepat proses pencocokan. Dalam kasus terburuknya, algoritma *Boyer-Moore* memiliki kompleksitas waktu O(nm + A) dengan A adalah kompleksitas ukuran alfabet. Algoritma ini secara signifikan lebih cepat daripada pencocokan *string* dengan algoritma *brute force.*

D. Permainan Tetris

Tetris adalah permainan teka-teki video *puzzle* yang melibatkan jatuhnya blok-blok geometris untuk mengisi pola blok pada papan permainan. Tetris dikembangkan oleh Alexey Pajitno, seorang ilmuan Uni Soviet, tahun 1984.

Gambar 2.1 Logo Permainan Tetris
(Sumber : https://www.facebook.com/TetrisMessenger/)

Permainan ini, menggunakan blok geometris atau disebut tetromino yang harus disusun secara rapi untuk membentuk baris penuh tanpa celah untuk bisa mendapatkan poin. *Gameplay* yang sederhana dapat menarik perhatian banyak orang hingga dikembangkan ke banyak platform permainan.

Sebuah tetromino, yaitu bentuk geometris yang terdiri dari empat kotak kecil, muncul di atas layar dan mulai jatuh ke bawah. Pemain dapat mengubah posisi dan rotasi tetromino

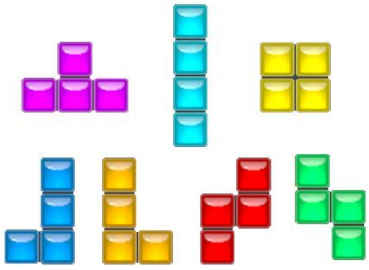
sebelum menempatkannya di area permainan. Pemain menempatkan tetromino di area permainan dengan tujuan membentuk baris horizontal penuh. Permainan berakhir jika area permainan penuh dan tidak ada ruang untuk menempatkan tetromino baru atau blok telah melewati batas bidang permainan.

III. Penerapan Algoritma Pattern Matching dan Branch and Bound Pada Optimasi Peletakan Tetromino Permainan Tetris Sederhana

A. Penerapan Algoritma dan Branch and Bound Pada Optimasi Peletakan Tetromino Permainan Tetris Sederhana

Dalam pengembangannya, beberapa permainan tetris menerapkan beberapa peraturan tambahan, seperti adanya bonus *combo*, *power-ups*, kemampuan untuk "*hold*" blok, dan sebagainya. Namun, untuk mengilustrasikan penerapan algoritma *pattern matching* dan *branch and bound* pada permainan tetris sederhana akan digunakan batasan aturan dasar permainan antara lain:

1. Permainan dimainkan pada bidang permainan persegi panjang yang terdiri dari kolom dan baris. Pada pembahasan kali ini, digunakan ukuran bidang permainan 15 x 20.
2. Permainan Tetris menggunakan berbagai bentuk blok yang disebut tetromino. Terdapat 7 bentuk tetromino seperti pada gambar berikut:



Gambar 3.1 Tetromino Tetris
(Sumber: https://tetris.com)

3. Tetromino jatuh dari bagian atas bidang permainan ke bawah secara perlahan-lahan.
4. Pemain dapat menggerakkan blok ke kiri atau kanan dan dapat memutar blok 90/180/270 derajat searah jarum jam.
5. Tujuan utama dalam Tetris adalah untuk mengisi baris secara horizontal dengan blok-blok secara penuh tanpa adanya celah.
6. Permainan Tetris berakhir ketika blok keluar dari bidang permainan dan poin akan dihitung berdasarkan banyaknya baris yang dapat terisi penuh tanpa celah.

Dalam merepresentasikan sistem permainan tetris sederhana dalam Algoritma *Branch and Bound*, terdapat beberapa properti algoritma yang didefinisikan sebagai berikut:

- Simpul hidup yang menjadi simpul-E(*expand node*) adalah simpul yang mempunyai nilai *cost* terkecil (*least cost search*)
- Nilai *cost* untuk setiap simpul memperhatikan aspek-aspek berikut:
  a. Ketinggian maksimum blok pada bidang permainan.
  b. Banyaknya celah yang terdapat pada bidang permainan.

  Pada persoalan penempatan tetromino, kedua aspek tersebut akan diminimasi dengan prioritas utama tetap meminimalkan ketinggian maksimum blok agar lebih leluasa dalam penempatan blok selanjutnya, sehingga didefinisikan estimasi nilai *cost* untuk simpul ke-i:

$$\hat{c}(i) = ketinggian\ maksimum\ blok + (0.5) * banyak\ celah$$

- Nilai *cost* terkecil sementara yang telah ditemukan disimbolkan sebagai c_{min}
- Fungsi Pembatas atau *bounding function* pada setiap simpul ke-i didefinisikan sebagai berikut:

$$B(i) = ketinggian\ maksimum\ blok \leq 15\ \textbf{and if}\ (solusi\ sudah\ pernah\ ditemukan)\ \textbf{then}\ \hat{c}(i) < c_{min}\ \textbf{else}\ true$$

  sehingga jika $B(i)$ bernilai *true*, maka nilai *cost* terkecil sementara akan berubah:

$$c_{min} = \hat{c}(i)$$

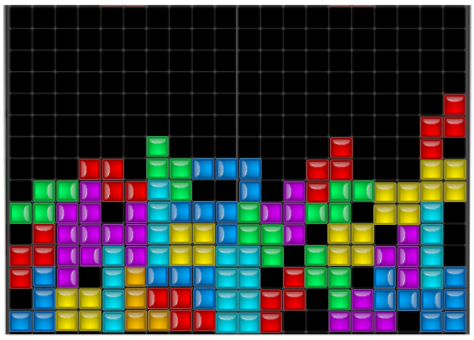
- Simpul solusi didefinisikan sebagai simpul yang dihasilkan setelah melakukan penempatan semua tetromino yang tersedia.

Dengan memanfaatkan properti yang telah didefinisikan, penerapan Algoritma *Branch and Bound* pada optimasi peletakan tetromino dapat dilakukan melalui tahapan berikut:

1. Buat sebuah antrean, misal Q yang berupa *priority queue*, sehingga elemen terurut berdasarkan estimasi nilai *cost* $\hat{c}(i)$ secara menaik (*ascending*).
2. Masukkan simpul akar, yaitu tetromino pertama ke dalam antrean Q. Jika hanya terdapat satu tetromino saja, maka solusi (*goal node*) telah ditemukan.
3. Lakukan pemeriksaan apakah antrean Q kosong. Jika kosong yang berarti sudah tidak ada lagi simpul yang dapat di-*expand*, maka program berhenti dan ditetapkan *node* dengan nilai *cost* c_{min} sebagai *goal node*.
4. Jika Q tidak kosong, pilih dari antrean Q simpul i yang mempunyai nilai estimasi *cost* $\hat{c}(i)$ paling kecil yang berarti elemen pertama pada antrean Q.
5. Jika simpul *i* adalah simpul solusi, maka solusi telah ditemukan. Kemudian, lakukan pemeriksaan *boundary function* pada simpul tersebut. Jika $B(i)$ bernilai *true*, maka tetapkan simpul tersebut menjadi *goal node* sementara dan matikan simpul lain yang memiliki nilai estimasi *cost*, $\hat{c}(i) > c_{min}$
6. Jika simpul *i* bukan simpul solusi, maka bangkitkan semua anak-anaknya. Anak-anak yang dimaksud adalah konfigurasi hasil rotasi tetromino berikutnya.
7. Lakukan perhitungan nilai estimasi *cost* untuk setiap simpul anak. Optimasi nilai *cost* ini akan diimplementasikan dengan algoritma *Pattern Matching* yang akan dibahas pada bagian selanjutnya.
8. Jika simpul anak memenuhi *boundary function* $B(i)$, maka tambahkan simpul anak tersebut ke dalam *priority queue* Q.
9. Ulangi kembali dari langkah 3 hingga *goal node* ditemukan

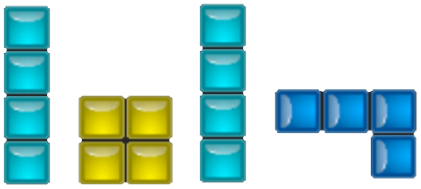
Melalui langkah-langkah tersebut, secara tidak langsung kita telah membangkitkan pohon ruang status persoalan penempatan tetromino dengan algoritma *branch and bound*. Salah satu contohnya adalah sebagai berikut:

Kondisi Papan Awal :

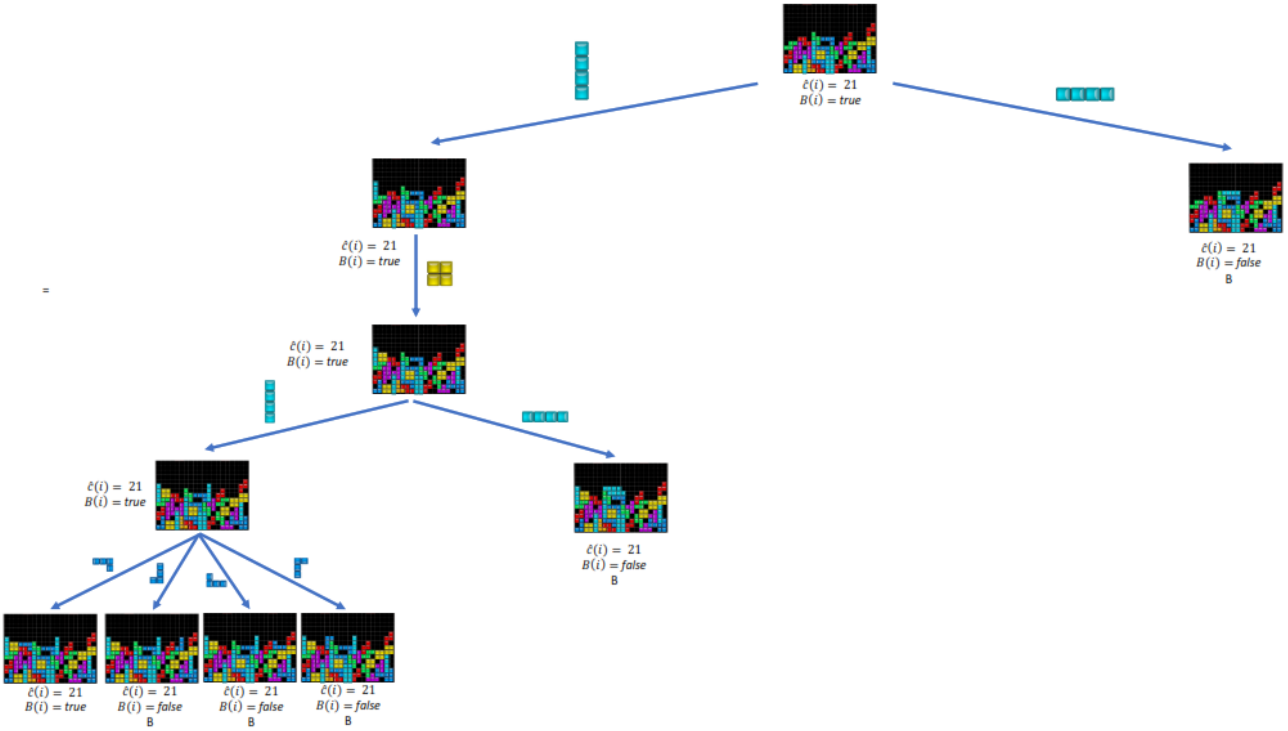


Gambar 3.2 Contoh Kondisi Awal Papan Permainan

Urutan Tetromino:



Gambar 3.3 Contoh Urutan Tetromino



Gambar 3.4 Pohon Ruang Status Optimasi Penempatan Tetromino pada Permainan Tetris Sederhana

B. Penerapan Algoritma Boyer-Moore Pada Penentuan Posisi Peletakan Tetromino

Seperti yang telah disebutkan sebelumnya, Optimasi nilai *cost* setiap tetromino dapat diimplementasikan dengan algoritma *Pattern Matching.* Algoritma *Pattern Matching* atau *String Matching* yang akan diimplementasikan adalah Algoritma Boyer-Moore.

Sebelum mengimplementasikan Algoritma Boyer-Moore, perlu dilakukan representasi bentuk pola blok pada bidang permainan serta blok tetromino ke dalam bentuk *string*. Salah satu alternatif representasi *string* yang dapat dilakukan adalah sebagai berikut:

1. Buat *array* sebagai representasi bentuk pola blok pada bidang permainan, misal *array* T, dengan panjang 19, yaitu sebanyak kolom bidang permainan dikurangi 1.
2. Isi elemen *array* ke-i menjadi selisih ketinggian blok pada kolom ke-i dan ketinggian blok pada kolom ke-(i+1):

   $$T[i] = ketinggian\ kolom\ i - ketinggian\ kolom\ i + 1$$

3. Hasil representasi blok menjadi *array of integer* tersebut kemudian direpresentasikan menjadi *array of character* melalui *hash table* menggunakan representasi karakter pada kode ASCII. Fungsi *hash* dapat didefinisikan sebagai berikut :

   $$T[i] = char(T[i] + 77)$$

4. *Array of character* tersebut kemudian direpresentasikan dalam bentuk *string*.

Melalui tahapan tersebut, kita dapat merepresentasikan pola blok, baik pola pada bidang permainan maupun pola tetromino, menjadi *string*. Perlu diperhatikan bahwa tidak semua

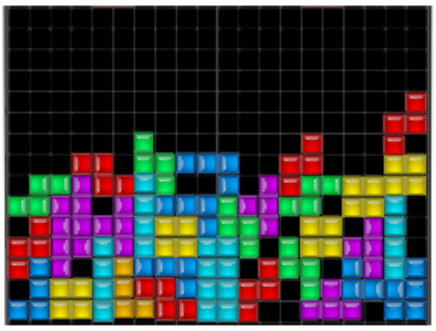
bentuk/konfigurasi tetromino dapat direpresentasikan menjadi *string* dengan cara tersebut.

Bentuk yang tidak dapat direpresentasikan menjadi *string* adalah bentuk tetromino lurus vertikal, karena pada dasarnya tetromino ini dapat ditempatkan pada posisi kolom mana pun, tetapi harus tetap mengikuti aturan *bounding function* yaitu:

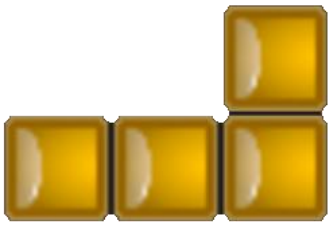
$$ketinggian\ maksimum\ blok \leq ketinggian\ maksimum\ bidang\ permainan$$

Gambar 3.6 Tetromino Vertikal Satu Kolom
(Sumber: https://tetris.com)

Berikut adalah contoh representasi pola blok menjadi bentuk string:



Gambar 3.5 Contoh Pola Blok Papan Permainan



Gambar 3.7 Contoh Pola Blok Tetromino

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
T[i]	+1	0	+1	0	-1	+2	-1	0	0	0	-2	+1	+1	+1	-2	0	0	+3	+1	+1
h(i)	N	M	N	M	L	O	L	M	M	M	K	N	N	N	K	M	M	P	N	N

Tabel 3.1 Representasi *String* Pola Blok pada Papan Permainan

i	0	1
T[i]	0	0
h(i)	M	M

Tabel 3.2 Representasi *String* Pola Blok pada Tetromino

Setelah dilakukan representasi pola menjadi bentuk *string*, pola bidang permainan menjadi *text* dan pola tetromino menjadi *pattern*, kemudian diimplementasikan algoritma *Boyer-Moore* untuk menemukan posisi *pattern* yang *exact match* dengan pola pada *text*..Proses pencocokan pola dilakukan melalui tahapan berikut:

1. *Pre-processing*: Buat *last occurrence* table yang mencatat posisi kemunculan terakhir dari setiap karakter yang terdapat pada *text*. Jika karakter tidak muncul dalam pola, berikan nilai -1 pada entri tabel.
2. Posisikan *pattern* pada awal *text*, yaitu posisi T[0] sejajar dengan P[0].
3. Bandingkan *pattern* dengan *text* dari kanan ke kiri.
4. Jika ditemukan ketidakcocokan pada karakter tertentu (*mismatch*), periksa *last occurrence table* untuk menentukan jumlah pergeseran yang diizinkan. Terdapat 3 kasus dalam penentuan banyak pergeseran:
   Misalkan terjadi ketidakcocokan/*mismatch* pada
   $$P[j] \neq T[i] = x\ ,$$
   a. Kasus 1:
      Jika pada *pattern* P terdapat x di suatu tempat, maka coba geser *pattern* P ke kanan untuk menyejajarkan *last occurrence* x di P dengan T[i].
   b. Kasus 2:
      Jika pada *pattern* P terdapat x di suatu tempat, tetapi posisi x berada di kanan P[j], maka geser P ke kanan sebanyak 1 karakter ke T[i+1].
   c. Kasus 3:
      Jika kasus 1 dan 2 tidak berlaku, maka geser P untuk menyejajarkan P[0] dengan T[i+1].
5. Geser pola ke kanan sejauh pergeseran maksimum yang telah ditentukan sebelumnya.
6. Ulangi langkah tersebut hingga pencarian sampai di akhir *text*.
7. Jika ditemukan lebih dari satu posisi yang *exact match*, maka pilih posisi dengan ketinggian yang lebih rendah.

Berikut adalah contoh implementasi pencocokan pola *pattern* pada *text* dengan algoritma *Boyer-Moore*:

Text : NMNMLOLMMMKNNNKMMPNN
Pattern : MM
Pre-processing : *Last Occurrence Table*

K	L	M	N	O	P
-1	-1	1	-1	-1	-1

Tabel 3.3 *Last Occurrence Table*

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
N	M	N	M	L	O	L	M	M	M	K	N	N	N	K	M	M	P	N	N
M 2	M 1							M 9	M 8							M	M 15		
	M	M 3							M	M 10								M	M 16
			M	M 4							M	M 11							
					M	M 5							M	M 12					
							M 7	M 6							M 14	M 13			

Tabel 3.4 Pencocokan String dengan Algoritma Boyer Moore

Dari proses pencocokan tersebut didapatkan hasil bahwa terdapat 3 posisi pada *text* yang *exact match* dengan *pattern.* Untuk menentukan posisi terbaik, dilakukan perhitungan terhadap estimasi nilai *cost* dan ketinggian posisi *exact match* ditemukan untuk dipilih nilai paling minimum. Sebagai contoh pada kasus di atas, ditemukan posisi paling optimum berada pada T[15].

No.	Posisi T[i]	Nilai *cost*	Ketinggian T[i]
1	7	21	9
2	8	21	9
3	15	21	8

Tabel 3.5 Pencarian Posisi *Exact Match* Paling Optimum

IV. Simpulan

Tetris adalah permainan teka-teki video *puzzle* sudah ada sejak tahun 1984 dan terkenal hampir di seluruh dunia. Dalam permainan ini, pemain harus menempatkan tetromino dengan strategi tertentu sehingga dapat menciptakan baris penuh tanpa celah. Dalam setiap langkah peletakan tetromino, perlu diperhatikan pemilihan lokasi dan orientasi yang tepat untuk dapat meningkatkan efisiensi permainan dan memungkinkan pemain untuk mencapai skor tertinggi. Desain optimasi ini dapat diimplementasikan dengan menerapkan konsep penyelesaian permasalahan optimasi yang dipelajari pada mata kuliah IF2211 Strategi Algoritma, yaitu *pattern matching* dengan algoritma *Boyer-Moore* dan algoritma *Branch and Bound.*

V. Ucapan Terima Kasih

Pertama-tama, penulis panjatkan puji syukur kepada Tuhan Yang Maha Esa atas limpahan berkah, rahmat, dan karunia-Nya, makalah ini dapat diselesaikan dengan baik. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi, M.T., selaku dosen pengajar mata kuliah IF2211 Strategi Algoritma K01 yang telah dengan sabar memberikan banyak ilmu serta memberi motivasi selama perkuliahan. Terima kasih pula pada orang tua dan teman-teman yang selalu memberi dukungan dan semangat sehingga penulis dapat menyelesaikan tugas makalah ini dengan lancar. Penulis menyadari bahwa penulis masih jauh dari kata sempurna, sehingga penulis memohon maaf apabila terdapat kesalahan dalam penulisan makalah ini, baik yang disengaja maupun tidak disengaja. Penulis sangat terbuka terhadap masukan, kritikan, dan saran yang bersifat membangun terhadap makalah ini. Akhir kata, penulis berharap makalah ini dapat bermanfaat bagi banyak orang.

VI. Tautan Video

Video penjelasan terkait makalah ini diunggah dalam Youtube "Penerapan Algoritma *Boyer Moore* dan Algoritma *Branch and Bound* Pada Optimasi Peletakan Tetromino Permainan Tetris Sederhana - IF2211 Strategi Algoritma" yang dapat diakses melalui tautan berikut :

https://youtu.be/e7GnIh2d7E8

Daftar Pustaka

[1] Anonymous. (2022). What is Pattern Matching | Deepchecks. Deepchecks. Diakses pada 20 Mei 2023, dari https://deepchecks.com/glossary/pattern-matching/

[2] GeeksforGeeks. (2023). Branch and Bound Algorithm. GeeksforGeeks. Diakses pada 20 Mei 2023, dari https://www.geeksforgeeks.org/branch-and-bound-algorithm/

[3] Hanna, Yomi (2017). Sejarah Lahirnya Permainan Tetris yang Mendunia - Bobo. Bobo. Diakses pada 20 Mei 2023, dari https://bobo.grid.id/read/08679391/sejarah-lahirnya-permainan-tetris-yang-mendunia?page=all

[4] Program Studi Teknik Informatika STEI-ITB (2021). Bahan Kuliah IF2211 Strategi Algoritm: Pencocokan String (String/Pattern Matching). Diakses pada 19 Mei 2023 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Pencocokan-string-2021.pdf

[5] Rinaldi Munir, Nur Ulfa Maulidevi, Masayu Leylia Khodra (2021). Bahan Kuliah IF2211 Strategi Algoritm: Algoritma Branch and Bound (Bagian 1). Diakses pada 19 Mei 2023 dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Branch-and-Bound-2021-Bagian1.pdf

[6] Ruber.Id. (2023). Sejarah dan Fenomena Game Tetris. ruber.id. Diakses pada 20 Mei 2023, dari https://ruber.id/sejarah-dan-fenomena-game-tetris

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 2023

Husnia Munzayana
13521077