Penerapan Algoritma Backtracking dalam Strategi 4 Wide pada Permainan Tetris Modern

Muhammad Naufal Satriandana - 13520068
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13520068@std.stei.itb.ac.id

Abstract*— Tetris adalah sebuah permainan puzzle yang diciptakan programmer Bernama Alexey Pajitnov pada tahun 1984. Pemain harus menyusun potongan puzzle, yang disebut Tetrimino, yang jatuh dari atas matriks permainan, sehingga terbentuk baris horizontal yang penuh. Seiring berjalannya waktu, varian Tetris yang lain, yang disebut Tetris modern mulai muncul dengan beberapa aturan yang berbeda dengan Tetris klasik. Pemain Tetris kemudian menciptakan berbagai strategi untuk memenangkan permainan Tetris modern, salah satunya adalah strategi 4 wide. Akan tetapi, untuk memaksimalkan potensi strategi 4 wide, dibutuhkan gerakan-gerakan yang tepat pada setiap langkahnya. Penerapan hal tersebut bisa menggunakan Algoritma backtracking. Dengan memanfaatkan Algoritma backtracking, dapat ditmukan tujuan akhir gerakan dan langkah-langkah yang harus diambil untuk mencapai tujuan itu berdasarkan beberapa informasi seperti hold queue, current Tetrimino, dan next queue. Setelah mencapai tujuan akhir, Algoritma backtracking tersebut dijalankan kembali hingga tidak ditemukan lagi goal node atau permainan telah selesai.

***Keywords*— Tetris, Backtracking, Tree, Depth First Search**

I. PENDAHULUAN

Pada era modern ini, semua berkembang dan berubah, termasuk permainan-permainan yang biasa dimainkan oleh manusia di dunia. Hal demikian juga terjadi pada permainan puzzle, salah satunya adalah permainan Tetris yang kini sudah berkembang menjadi versi baru, yakni Tetris modern.

Tetris adalah sebuah permainan (*video game*) puzzle yang diciptakan seorang programmer komputer Alexey Pajitnov pada tahun 1984. Alexey Pajitnov membuat permainan elektronik tersebut karena terinspirasi dari permainan puzzle favoritnya, Pentominos. Pada tetris, pemain dapat menyusun potongan puzzle yang jatuh dari atas layar secara *real-time*. Potongan-potongan puzzlenya merupakan tujuh potongan geometrik yang berbeda yang setiap potongannya terdiri dari empat kotak. Pajitnov menyebut permainan ini Tetris, yang merupakan kombinasi dari "tetra" (Bahasa Yunani dari empat) dan "tenis" (olahraga favoritnya).

Seiring berjalannya waktu, tetris menjadi semakin populer dan mulai diikuti di berbagai perlombaan, yang menjadikan permainan ini kompetitif. Selain itu, tetris mulai bermunculan di berbagai platform, seperti komputer, beberapa konsol permainan, ponsel, dan lain-lain. Kemudian, beberapa variasi dari permainant Tetris mulai bermunculan. Dalam variasi-variasi Tetris yang baru ini, beberapa perubahan dibuat, seperti sistem pengacakan potongan puzzle yang diberikan kepada pemain, gerakan potongan yang bisa dilakukan oleh pemain, tujuan permainan, dan lain-lain. Lama kelamaan, orang-orang mulai mengkategorikan Tetris menjadi dua, yakni Tetris klasik dan Tetris modern.

Berbeda dengan Tetris klasik yang bertujuan untuk mendapatkan poin sebanyak-banyaknya, pada variasi Tetris modern, khususnya Tetris modern yang melibatkan dua pemain untuk bertarung (contohnya tetr.io dan Puyo Puyo Tetris), kedua pemain berusaha untuk memenangkan permainan dengan cara membuat pemain lawan tidak bisa menaruh potongan puzzlenya lagi. Tetris modern mengutamakan kecepataan dan strategi untuk memenangkan permainan. Oleh karena itu, banyak strategi-strategi yang mulai bermunculan untuk memenangkan permainan Tetris modern.

Salah satu strategi yang ada dalam Tetris modern adalah strategi 4 wide (*4 column wide*). Pada strategi ini, pengetahuan atas potongan puzzle selanjutnya dan dampaknya jika suatu potongan puzzle diletakkan merupakan sesuatu yang penting. Suatu kesalahan dalam meletakkan potongan puzzle dapat berdampak pada kurangnya pemaksimalan potensi strategi 4 wide atau bahkan kegagalan strategi tersebut. Oleh karena itu, dibutuhkan suatu algoritma untuk menentukan langkah-langkah yang diambil dalam menentukan setiap gerakan agar strategi 4 wide dapat digunakan dengan optimal.

Setiap kemungkinan gerakan dan susunan (kondisi) puzzle pada saat tertentu dapat dimodelkan sebagai sebuah pohon. Kemudian, dapat ditentukan suatu tujuan yang ingin dicapai lalu dengan algoritma backtracking, dapat ditentukan gerakan yang mengarah ke tujuan. Oleh karena itu, penulis membuat makalah dengan judul "Penerapan Algoritma Backtracking dalam Strategi 4 Wide pada Permainan Tetris Modern".

II. TEORI DASAR

A. *Algoritma Backtracking*

Backtracking atau runut balik bisa dilihat sebagai sebuah fase dalam algoritma traversal depth first search (DFS) atau bisa juga dilihat sebagai sebuah metode pemecahan masalah yang efektif, terstruktur, dan sistematis untuk sebuah persoalan. Backtracking merupakan algoritma perbaikan dari exhaustive search yang dalam hal ini, exhaustive search mencari dan

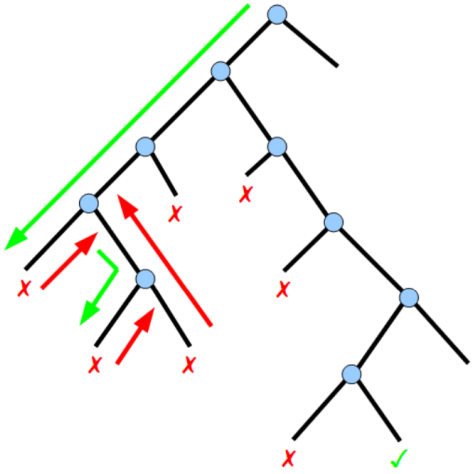
mengeksplorasi semua kemungkinan solusi dan mengevaluasinya satu per satu, sedangkan pada algoritma runut balik, pilihan yang dieksplorasi dan dievaluasi hanya pilihan yang mengarah ke solusi dan pilihan lainnya tidak dipertimbangkan lagi. Hal tersebut dilakukan dengan cara memangkas simpul-simpul yang tidak mengarah ke solusi.

Algoritma backtracking pertama kali ditemukan pada tahun 1950 oleh D. H. Lehmer, yang kemudian uraian umum tentang algoritma tersebut disajikan oleh R. J. Walker, Golomb, dan Baumert.

Algoritma backtracking umumnya memiliki tiga properti, antara lain adalah:

1) Solusi persoalan, yakni dinyatakan sebagai sebuah vector dengan n-tuple X, yang dalam hal ini X = (x1, x2, …, xn) dan xi ∈ Si. Pada umumnya, S1 = S2 = … = Sn.
2) Fungsi pembangkit (untuk sebuah nilai x_k), yang dinyatakan sebagai predikat T(), yang dalam hal ini, T(x[1], x[2], …, x[k-1]) membangkitkan nilai untuk x_k, dan x_k merupakan sebuah komponen pada vektor solusi.
3) Fungsi pembatas (*bounding function*), yang dinyatakan sebagai sebuah predikat B(x1, x2, …, xk), yang dalam hal ini B bernilai true jika x1, x2, …, xk mengarah ke solusi (tidak melanggar kendala (*constraints*)). Jika *bounding function* bernilai true, maka pembangkitan untuk x_{i+1} dilanjutkan, tetapi jika false, maka (x1, x2, …, xk) dibuang.

Semua kemungkinan dari sebuah solusi persoalan merupakan ruang solusi atau solution space yang diorganisasikan ke dalam struktur pohon berakar. Status persoalan dilambangkan oleh setiap simpul pohon, sedangkan sisi dilabeli dengan nilai-nilai x_i. Lintasan dari akar ke daun merupakan sebuah solusi yang mungkin, dan ruang solusi dibentuk dari seluruh lintasan dari akar ke daun. Pengorganisasian pohon ruang solusi dinamakan pohon ruang status atau *state space tree*.



Gambar 1. Ilustrasi pencarian solusi dengan algoritma backtracking

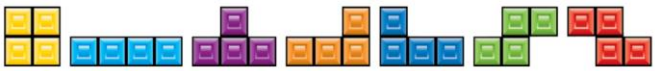
Dikutip dari https://www.w3.org/2011/Talks/01-14-steven-phenotype/ [2]

Dengan algoritma backtracking, solusi dicari dengan membangkitkan simpul-simpul status menggunakan aturan DFS hingga akhirnya menghasilkan lintasan dari akar ke daun. Simpul-simpul yang dibangkitkan dinamakan sebagai simpul hidup (*live node*), sedangkan simpul hidup yang sedang diperluas dinamakan sebagai *expand node* atau simpul E. Lintasan yang dibangun akan bertambah panjang jika simpul E diperluas. Selain itu, fungsi pembatas akan mematikan simpul E yang tidak mengarah ke solusi, sehingga simpul yang dimatikan tersebut menjadi simpul mati (*dead node*). Dengan melakukan hal tersebut, maka secara tidak langsung simpul-simpul anaknya telah terpangkas. Proses pencarian akan runut balik ke simpul atasnya jika pembentukan lintasan berakhir dengan simpul mati. Kemudian, pembangkitan simpul anak yang lainnya diteruskan, yang dalam hal ini simpul anak yang dibangkitkan tersebut akan menjadi simpul E yang baru. Jika telah sampai pada goal node atau semua goal node telah ditemukan, pencarian dihentikan.

B. Tetris

Tetris adalah permainan (*video game*) yang diciptakan oleh Alexey Pajitnov pada tahun 1984. Dalam permainan Tetris, pemain harus memanipulasi potongan-potongan puzzle geometrik berbeda, yang terdiri dari empat kotak terhubung (Tetrimino) yang diberikan, yang jatuh dari atas ke sebuah matriks persegi panjang. Pemain berusaha untuk membersihkan (*clear*) sebuah baris dengan mengisi baris secara horizontal dengan potongan-potongan puzzle (Tetrimino) tanpa ada ruang kosong pada baris tersebut, kemudian baris tersebut disingkirkan dari matriks.

Dalam permainan Tetris pada umumnya, terdapat 7 Tetrimino yang dapat dirotasi, digeser, atau dijatuhkan, antara lain sebagai berikut:



Gambar 2. Ketujuh tetrimino umum dalam permainan tetris

Dikutip dari Tetris Design Guideline 2009 [4]

1) O-Tetrimino, berwarna kuning dan berbentuk kotak yang dalam hal ini empat kotak 2x2 membentuk 1 kotak
2) I-Tetrimino, berwarna biru muda, berbentuk seperti huruf I kapital, dan merupakan 4 kotak yang berjajar pada satu garis lurus
3) T-Tetrimino, berwarna ungu, berbentuk huruf T kapital, dan terdiri dari tiga kotak berjajar ditambah satu kotak di atas kotak tengah.
4) L-Tetrimino, berwarna oranye, berbentuk seperti huruf L kapital, dan terdiri dari tiga kotak berjajar ditambah satu kotak di atas kotak kanan.
5) J-Tetrimino, berwarna biru tua, berbentuk seperti huruf J kapital, dan terdiri dari tiga kotak berjajar ditambah satu kotak di atas kotak kiri
6) S-Tetrimino, berwarma hijau, berbentuk seperti huruf S kapital, dan merupakan dua buah, dari dua kotak yang berjajar secara horizontal, saling ditumpuk dengan yang diatas digeser satu kotak ke kanan
7) Z-Tetrimino, berwarma merah, berbentuk seperti huruf S kapital, dan merupakan dua buah, dari dua kotak yang berjajar secara horizontal, saling ditumpuk dengan yang diatas digeser satu kotak ke kiri.

Dalam permainan Tetris, pada umumnya terdapat matriks utama berukuran 10 kolom dan 20 baris untuk tempat bermain. Dalam hal ini, Tetrimino yang jatuh dari atas matriks ini dapat dimanipulasi oleh pemain untuk mengisi sebuah tempat pada matriks ini untuk *clear* sebuah baris. Baris tersebut kemudian dihilangkan dari matriks ini.

Beberapa hal yang umum ada dalam permainan tetris dan perlu diketahui dalam makalah ini antara lain adalah:

1) *Block* merupakan sebuah kotak yang terkunci pada suatu sel pada matriks
2) *Line clear*, terjadi ketika keseluruhan sebuah baris horizontal terisi block. Baris tersebut kemudian disingkirkan dari matriks dan semua block diatas baris tersebut digeser satu baris kebawah untuk mengisi kekosongan.
3) *Hold queue*, merupakan tempat penyimpanan sebuah Tetrimino agar pemain dapat menyimpan Tetrimino yang sedang jatuh. Ketika digunakan, Tetrimino yang sedang jatuh akan bertukar posisi dengan Tetrimino yang ada di *hold queue*
4) *Next queue*, menampilkan antrian Tetrimino selanjutnya yang akan diberikan kepada pemain dan dibangkitkan di atas matriks. Jika memungkinkan, 6 Tetrimino selanjutnya akan ditampilkan
5) *Block out*, merupakan kondisi *Game Over*, yang dalam hal ini, bagian atau seluruh bagian dari Tetrimino yang baru dibangkitkan terhalang karena adanya *block* lain yang mengisi tempat tersebut.
6) *Super rotation system* (*SRS*), sistem yang memungkinkan perputaran sebuah Tetrimino dimanapun letaknya pada matriks.

Pemain akan kalah dalam permainan Tetris jika terjadi *block out*.

Adapun Tetris Modern yang penulis maksud dalam makalah ini, adalah variasi dari permainan Tetris, yang memiliki beberapa aturan yang berbeda dari permainan Tetris klasik. Dalam Modern Tetris yang penulis maksud, permainan merupakan permainan yang bisa melibatkan dua atau lebih pemain, dan tujuan permainannya adalah membuat pemain lawan *block out* dengan cara mengirimkan serangan yang berupa baris *block* tambahan kepada pemain lawan, dengan cara melakukan sejumlah *line clear*. Sedangkan dalam Tetris klasik yang penulis maksud, permainan biasanya dimainkan oleh satu orang (walaupun dalam permainan Tetris klasik kompetitif, permainan bisa dimainkan dua orang atau lebih) dan tujuan permainannya adalah dengan meraih sebanyak banyaknya poin dengan cara melakukan sejumlah *line clear* sebelum terjadi *block out*. Selain itu, terdapat beberapa perbedaan lain antara Tetris klasik dan Tetris modern, antara lain yakni Tetris Modern memiliki *hold queue*, sedangkan Tetris klasik tidak. Selain itu, pembangkitan Tetrimino selanjutnya pada Tetris klasik merupakan suatu hal yang sepenuhnya acak, namun pada Tetris Modern, pembangkitan dilakukan dalam *bag* yang diskrit, yang dalam hal ini, *Bag* merupakan deretan Tetrimino yang unik, biasanya berjumlah 7. Hanya isi dari *bag* ini kemudian diacak, yang mengakibatkan tidak mungkin ada Tetrimino yang sama dalam 1 bag (7 deretan Tetrimino) dan tidak mungkin ada lebih dari 2 Tetrimino sama yang berurutan kapan pun. Untuk penjelasan yang lebih lengkap, dapat mengacu ke Tabel 1.

Tetris Klasik	**Tetris Modern**
Tetrimino selanjutnya yang dibangkitkan sepenuhnya acak	Tetrimino dibangkitkan dalam sebuah *Bag* yang berisi 7 Tetrimino yang diacak urutannya
Tidak memiliki fungsi hold dan *Hold Queue*	Memiliki fungsi hold dan *Hold Queue*
Tetrimino yang sudah menyentuh bagian atas block lain tidak dapat digeser	Tetrimino yang sudah menyentuh bagian atas block lain bisa digeser
Pemain harus menunggu Tetrimino jatuh hingga posisi yang diinginkan	Terdapat fungsi *Hard drop* yang bisa langsung membuat Tetrimino jatuh dalam sekejap
Tetrimino yang tampak pada *Next queue* hanya satu	Tetrimino yang tampak pada *Next queue* bisa mencapai 6

Dimainkan 1 orang	Bisa dimainkan lebih dari 1 orang
Tujuan permainan adalah mendapatkan sebanyak-banyaknya poin sebelum *Block out*	Tujuan permainan untuk pemain lebih dari 1 adalah menyebabkan pemain lawan *Block out* sebelum diri sendiri *Block out*

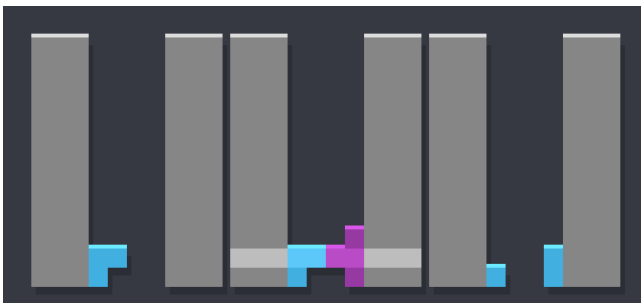
Tabel 1. Perbedaan Tetris klasik dan tetris modern

Karena pada Tetris modern tujuan permainannya adalah untuk menyebabkan pemain lawan *Block out* dengan cara mengirimkan baris-baris *block*, dan untuk mengirimkan baris *block* dibutuhkan untuk melakukan sejumlah *line clear* lebih cepat dibanding pemain (agar tidak terkena *Block out* lebih dulu), dibutuhkan pula strategi untuk melakukannya. Pada berbagai variasi permainan Tetris modern, jumlah baris *block* yang dikirim ke pemain lawan bisa berbeda-beda tergantung berapa jumlah *line clear* sekaligus, berapa jumlah *line clear* beruntun (*combo*), dengan teknik apa *line clear* dilakukan, *game* itu sendiri, dan lainnya. Oleh karena itu, terdapat beberapa strategi yang bisa diimplementasikan dengan tujuan memaksimalkan jumlah baris yang dikirim atau melindungi diri sendiri dari *Block out*, diantaranya adalah strategi 4 wide.

C. Strategi 4 Wide

Pada varian Tetris modern pada umumnya, semakin banyak jumlah *line clear* beruntun (kombo), maka semakin banyak pula baris *block* yang bisa dikirim ke pemain lawan. Oleh karena itu, mengoptimasi kombo merupakan salah satu hal yang dapat dilakukan untuk memenangkan permainan Tetris modern. Ada beberapa cara untuk mengoptimasi kombo, salah satunya adalah dengan Strategi 4 wide.

Strategi 4 wide adalah metode menumpuk *block* dengan cara mengosongkan 4 kolom dari matriks dengan menyisakan 3 *block* sisa (*residual blocks*). Dengan melakukan hal tersebut, melakukan *line clear* pada sebuah baris akan menyisakan 3 *residual block* lagi. Tujuan utama dari strategi ini adalah melakukan *line clear* pada sebuah baris secara beruntun agar mencapai kombo yang tinggi. Untuk ilustrasi dari strategi 4 wide, dapat mengacu ke gambar 3.

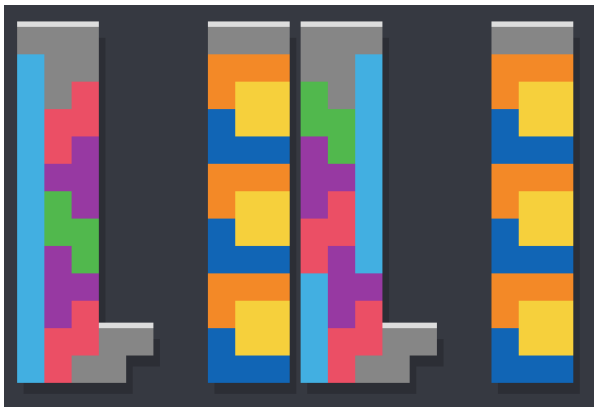


Gambar 3. Empat kolom tidak terisi dengan tiga *residual blocks* yang dikenakan *line clear* akan menghasilkan Empat kolom dengan tiga *residual blocks* juga.

Dikutip dari https://four.lol/stacking/4-wide [5]

Terdapat juga strategi 4 wide dengan menyisakan 4, 5, dan 6 *residual blocks*, namun karena kompleksitasnya, pada makalah ini hanya akan difokuskan pada 4 wide dengan menyisakan 3 *residual blocks*.

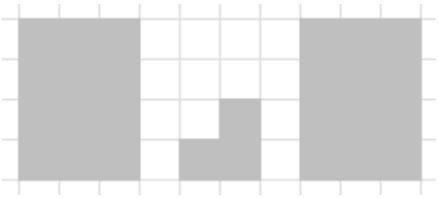
Untuk menyusun Tetrimino sehingga membentuk prekondisi untuk strategi 4 wide (mengosongkan empat kolom dan menyisakan tiga *residual block*) dapat dilakukan dengan beberapa cara. Namun agar memudahkan proses tersebut, terdapat sebuah pola yang umum dibuat. Hal ini bisa dilakukan karena sistem pembangkitan dari sistem modern, yang dalam hal ini 7 Tetrimino yang datang selanjutnya selalu sama walaupun urutannya mungkin berbeda. Untuk ilustrasi agar lebih jelas, dapat mengacu ke gambar 4.



Gambar 4. Beberapa pola untuk menyusun Tetrimino dalam strategi 4 wide

Dikutip dari https://four.lol/stacking/4-wide [5]

Tiga *residual block* yang tersisa tidak menjamin Tetrimino selanjutnya yang dibangkitkan bisa melakukan *line clear*. Bisa atau tidaknya suatu Tetrimino melakukan *line clear* tergantung pada susunan *residual blocks* dan Tetrimino yang sedang jatuh. Sebagai contoh, pada gambar 4, jika sebuah tetrimino hendak dijatuhkan, semua jenis Tetrimino selain J-Tetrimino bisa melakukan *line clear*, tetapi pada gambar 5, tidak ada Tetrimino selain I-Tetrimino yang dapat melakukan *line clear*.



Gambar 5. Contoh susunan *residual block* yang menyebabkan tidak ada Tetrimino selain I-Tetrimino yang bisa melakukan *line clear*

III. PEMBAHASAN

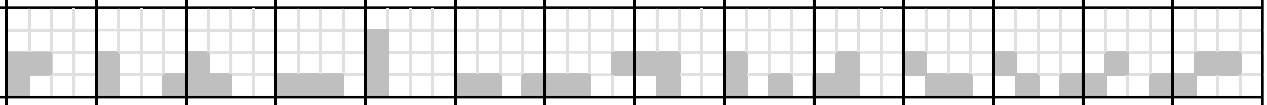
Penyelesaian masalah dengan menggunakan algoritma backtracking membutuhkan pendefinisian simpul dan sisi dari *state space tree* nya terlebih dahulu. Selain itu, properti-properti dari algoritma backtracking juga harus didefinisikan terlebih dahulu. Properti-properti tersebut antara lain adalah solusi persoalan, fungsi pembangkit dan fungsi pembatas.

A. Penentuan Simpul dan Sisi dari State Space Tree

Dalam implementasi ini, suatu simpul atau state (simpul dan state digunakan secara bergantian) dapat didefinisikan sebagai susunan *residual block* digabung dengan Tetrimino yang sedang jatuh (*current Tetrimino*) beserta Tetrimino yang ada dalam *hold queue*. Hal tersebut karena bisa atau tidaknya suatu kombo dilanjutkan, dalam hal ini berarti bisa atau tidaknya melakukan *line clear* bergantung pada kombinasi susunan *residual block*, *current Tetrimino*, dan Tetrimino dalam *hold queue*. Sebagai contoh, gambar 6 merupakan sebuah state dengan *current Tetrimino* adalah L-Tetrimino, Tetrimino dalam *hold queue* adalah O-Tetrimino, dan susunan *residual block* merupakan susunan yang tampak pada gambar. Untuk kemudahan, mulai dari sini, *current Tetrimino* akan disebut sebagai current, Tetrimino dalam *hold queue* akan disebut sebagai hold, dan susunan *residual block* akan disebut residual dan akan dilambangkan dengan sebuah kode yang merujuk pada tabel 2. Selain itu, setiap Tetrimino hanya akan dilambangkan dengan hurufnya saja. Jadi, state gambar 6 memiliki atribut yakni (current:L, hold:O, residual:1)

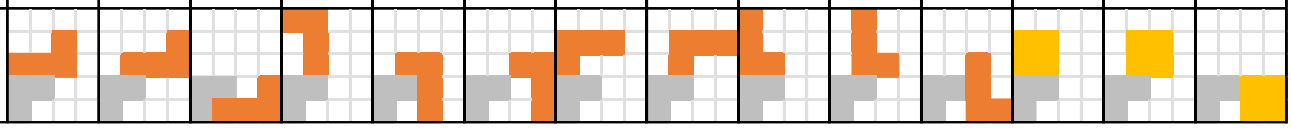
Suatu sisi dalam *state space tree* implementasi ini bisa didefinisikan sebagai gerakan yang mungkin dilakukan dalam sebuah state. Sebagai contoh, pada gambar 6, ada 14 gerakan yang dapat dilakukan dari state tersebut. Untuk ilustrasi, dapat merujuk tabel 3. Namun pada implementasi ini, sebagai heuristik, gerakan yang dijadikan sisi hanyalah gerakan yang menyebabkan terjadinya *line clear*. Setiap gerakan akan diberi nama sesuai dengan residual sebelum gerakan, digabung dengan nama Tetrimino, digabung dengan residual setelah gerakan (contoh: 1L6 berarti residual sebelum dan sesudah diletakkan L-Tetrimino adalah 1 dan 6 secara berurutan). Walaupun mungkin berbeda, gerakan yang menyebabkan residual 0 disamakan penamaannya karena residual 0 berarti state tersebut tidak akan memiliki simpul anak.

Perlu diperhatikan bahwa walaupun gerakan 1L6 tampak tidak bisa dilakukan karena block yang menghalangi, tetapi gerakan tersebut bisa dilakukan dengan memutar Tetrimino-L pada waktunya. Hal ini bisa terjadi karena *Super Rotation System* yang terdapat pada variasi Tetris modern. Selain itu, Tetrimino pada *Hold queue* dianggap sebagai Tetrimino yang dapat diletakkan karena bisa menggunakan fungsi hold untuk menukarnya dengan *current Tetrimino*.

Kode	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Residual														
Jumlah Gerakan Lanjutan	9	9	8	6	6	6	5	4	4	3	3	3	2	2
Tetrimino Lanjutan	I, O, T, L, S, Z	I, O, T, L, J, S	I, O, T, L, J, Z	I, T, L, J, S	I, T, L, J	I, T, J, Z	I, T, J, Z	I, O, T, Z	I, L, J	J, O	I, L, J	I, T, L	I, S	I, L

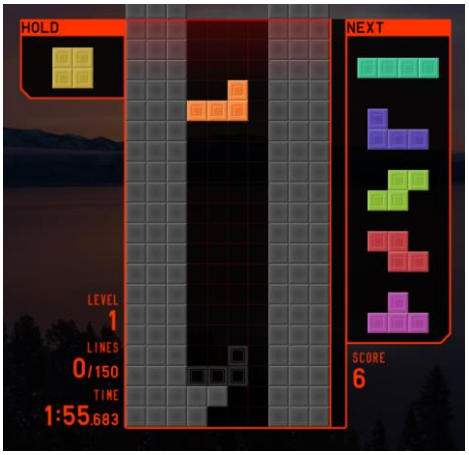
*Catatan: susunan residual block yang hanya memiliki Tetrimino lanjutan I akan diberi kode 0. Susunan residual yang merupakan pencerminan dari susunan yang sudah ada ditabel diberi kode [Susunan yang dicerminkan]' (contoh: pencerminan residual 1 akan diberi kode 1') dan semua Tetrimino lanjutan dicerminkan juga (Z dan S ditukar, J dan L ditukar)

Tabel 2. Susunan residual block yang mungkin

Gerakan	1L0	1L0	1L6	1L0	1L0	1L0	1L0	1L0	1L0	1L0	1L0	1O0	1O0	1O6'
Ilustrasi														

*Catatan, Gerakan yang tidak mennyebabkan *clear* sebuah line tidak akan dianggap (tidak dijadikan sebuah sisi). Dalam hal ini, hanya 1L6 dan 1O6' yang menjadi sebuah sisi

Tabel 3. Gerakan yang mungkin dilakukan untuk Gambar 6. Hanya 1L6 dan 1O6' yang dijadikan sisi



Gambar 6. Contoh sebuah simpul atau state

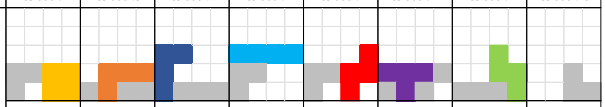
Dibuat dengan bantuan aplikasi tetr.io (https://tetr.io/) dan tetr.io plus (https://gitlab.com/UniQMG/tetrio-plus/)

B. Penentuan Solusi Persoalan

Tujuan akhir dari Tetris modern adalah dengan menyebabkan musuh *block out*, tujuan tersebut tidak menjadi solusi persoalan karena kekalahan pemain lawan tidak hanya berdasarkan peletakkan Tetrimino pemain saja, tetapi juga berdasarkan beberapa faktor lain seperti waktu, strategi pemain lawan, peletakkan Tetrimino pemain lawan, dan lainnya. Selain itu, Oleh karena itu, dibutuhkan solusi persoalan lain.

Dalam implementasi ini, dapat ditentukan bahwa dengan asumsi kedalaman maksimal yang dapat dibangkitkan adalah n, yang dalam hal ini n adalah jumlah Tetrimino yang tampak pada *next queue* ditambah dengan Tetrimino pada *hold queue* ditambah dengan *current Tetrimino* (Contohnya pada Gambar 6, n adalah 7), maka goal node adalah simpul yang berada di kedalaman n dan kode residual dari simpul tersebut tidak 0 (kode residual 0 artinya kombo tidak bisa dilanjutkan lagi kecuali dengan meletakkan Tetrimino I secara horizontal). Akan tetapi, timbul sebuah masalah, yakni jika kedalaman maksimal yang dapat dibangkitkan sama dengan jumlah Tetrimino yang tampak, maka tidak ada cara untuk mengetahui Tetrimino ke-n+1. Sebagai contoh, tinjau gambar 6. Setelah melakukan gerakan ke-6 (node E ada di kedalaman 6), node di kedalaman ke-7 tidak bisa dibangkitkan dengan mengisi atribut hold dan current. Atribut yang bisa diisi hanyalah residual, yakni didapatkan dari gerakan yang dilakukan pada node E. Selain itu, pada simpul di kedalaman ke-6, atribut hold tidak bisa diisi karena Tetrimino yang tersisa tinggal 1. Jika ada sebuah Tetrimino di hold namun tidak di current, maka secara automatis Tetrimino tersebut akan dipindahkan ke current. Untuk penjelasan lebih detail, dapat merujuk ke Tabel 4, yakni salah satu contoh lintasan dari akar ke daun, yakni salah satu simpul pada kedalaman 7 pada gambar 6. Dapat dilihat setelah state (current:S, hold:None, residual:7), gerakan apapun yang dilakukan akan menyebabkan state selanjutnya tidak memiliki current maupun hold. Oleh karena itu, penentuan goal node hanya dapat ditinjau dari residual node tersebut.

Goal node pada implementasi ini adalah simpul yang terletak pada kedalaman n, tidak memiliki current maupun hold, dan kode residual tidak sama dengan 0. Penentuan goal node ini dapat diartikan sebagai state yang dalam hal ini semua Tetrimino yang tampak telah terpakai (kombo berlanjut dengan sempurna) dan juga masih bisa dilanjutkan dengan setidaknya dua Tetrimino yang berbeda (jika kode residual sama dengan 0, maka yang bisa melanjutkan hanya I-Tetrimino, setelah itu tidak bisa dilanjutkan lagi). Jadi, solusi pencarian pada implementasi ini adalah semua lintasan dari simpul akar ke goal node.

State	current:L hold:O residual:1	current:I hold:L residual:6'	current:J hold:I residual:4'	current:S hold:I residual:1	current:Z hold:S residual:1	current:T hold:S residual:12'	current:S hold:None residual:7	current:None, hold:None, residual:10'
Gerakan								
	1O6'	6'L4'	4'J1	1I1	1Z12'	12'T7	7'S10'	None

Tabel 4. Salah satu contoh lintasan dari state awal gambar 6

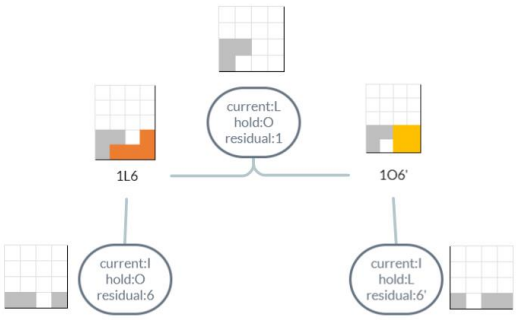
Dalam implementasinya, bisa saja goal node tidak ditemukan. Jika terjadi demikian, maka goal node adalah lintasan terpanjang dari simpul akar ke suatu simpul. Hal ini karena semakin panjang lintasan maka semakin banyak pula *line clear* yang terjadi.

Jika goal node yang ditemukan lebih dari 1, maka akan dipilih goal node dengan Tetrimino lanjutan terbanyak. Tetrimino lanjutan sebuah state dapat dilihat dari tabel 2.

C. Penentuan Fungsi Pembangkit

Fungsi pembangkit dalam implementasi ini bergantung pada fungsi pembatas. Suatu simpul E hanya akan dibangkitkan jika fungsi pembatas membolehkan. Selain itu, fungsi pembangkit dalam implementasi ini juga membutuhkan informasi tentang Tetrimino yang dibangkitkan selanjutnya dan apakah current ditukar dengan hold. Informasi ini bisa didapat dari *hold* queue dan *next queue*.

Fungsi pembangkit akan membangkitkan simpul berdasarkan gerakan yang bisa dilakukan dari state itu. Atribut dari simpul anak yang dibangkitkan akan diisi berdasarkan Tetrimino selanjutnya dan Tetrimino yang ada di *hold* queue. Sebagai contoh, state pada gambar 6 bisa dibangkitkan menjadi 2 simpul anak, yakni dengan melakukan gerakan 1L6 dan 1O6' (dapat merujuk ke tabel 3). 2 simpul anak tersebut adalah simpul beratribut (current:I, hold:O, residual:6) dan simpul beratribut (current:I, hold:L residual:6'). Setelah simpul anak dibangkitkan, simpul E berpindah sesuai aturan DFS, yakni ke salah satu simpul anak yang telah dibangkitkan tersebut. Untuk ilustrasi, dapat merujuk ke gambar 7.



Gambar 7. Pembangkitan simpul anak untuk state pada gambar 6

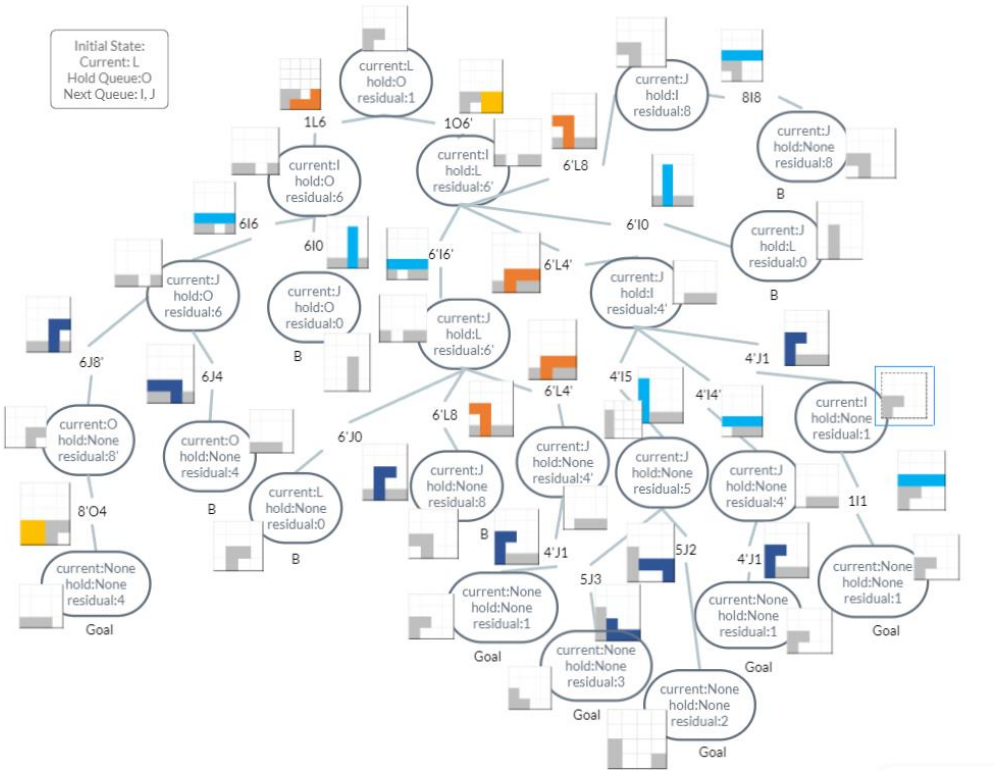
Dibuat dengan bantuan https://my.visme.co/

D. Penentuan Fungsi Pembatas

Dalam strategi 4 wide, pemain tidak bisa sembarang meletakkan Tetrimino karena bisa jadi gerakan tersebut menyebabkan "jalan buntu", yang dalam hal ini susunan *residual block* hasil penempatan Tetrimino tersebut menyebabkan Tetrimino selanjutnya atau Tetrimino pada *hold queue* tidak bisa melakukan *line clear*. Hal ini bisa menjadi dasar dari fungsi pembatas untuk implementasi ini.

Fungsi pembatas dalam implementasi ini dapat didefinisikan sebagai fungsi yang mengembalikan true jika node E memiliki minimal 1 gerakan dari state tersebut yang bisa menyebabkan line clear, dan mengembalikan false jika tidak demikian. Fungsi pembatas ini dapat diartikan sebagai fungsi yang mematikan state jika tidak ada Tetrimino yang bisa melanjutkan kombo dari state tersebut. Artinya state tersebut merupakan state buntu. Akan tetapi, state ini tidak dibuang dari *state space tree* karena bisa jadi merupakan kandidat dari goal node jika tidak ada goal node yang ditemukan (seperti yang telah dibahas di penentuan solusi persoalan).

Implementasi fungsi pembatas bisa menggunakan atribut residual, hold, current, dan juga Tetrimino lanjutan dari sebuah residual sebuah state. Tetrimino lanjutan dapat diketahui dari tabel 2. Penentuan true atau false nya fungsi pembatas ini adalah dengan memeriksa apakah salah satu (atau keduanya) dari Tetrimino yang ada di hold dan current terdapat pada Tetrimino lanjutan untuk residual state tersebut atau tidak. Jika ada, kembalikan true, selain itu kembalikan false (dengan catatan I-Tetrimino di hold maupun di current akan mengembalikan true, apapun kode residualnya). Sebagai contoh, jika node E memiliki atribut (current:L, hold:O, residual: 10), fungsi pembangkit akan mengembalikan true karena O ada pada Tetrimino lanjutan untuk residual 10, artinya ada gerakan yang bisa dilakukan dari state tersebut, dalam hal ini, gerakannya adalah 10O4'. Namun, jika node E memiliki atribut (current:L, hold:O, residual: 13), fungsi pembangkit akan mengembalikan false karena tidak ada L maupun O pada Tetrimino lanjutan untuk residual 13.



Gambar 7. State space tree yang dibangkitkan dari Gambar 6 dengan asumsi n adalah 4. Dalam hal ini, next queue yang tampak hanya 2

Dibuat dengan bantuan https://my.visme.co/

E. Solusi Runut Balik

Setelah menentukan semua properti dari algoritma backtracking untuk pencarian solusi 4 wide, langkah-langkah dari solusi bisa ditentukan. Langkah-langkah tersebut adalah:

1) Jika state belum ditinjau, tinjau state beserta atribut-atributnya dan lanjutkan ke tahap 2. Jika state sudah ditinjau, Tinjau anak simpul tersebut yang belum ditinjau. Jika simpul tidak memiliki anak atau semua anaknya sudah ditinjau, lakukan backtrack ke aras atas, kemudian lakukan tahap 1 lagi. Jika tidak ada simpul yang dapat ditinjau lagi, pilih solusi persoalan dengan state terakhir yang memiliki Tetrimino lanjutan terbanyak. Jika tidak ada goal node, pilih simpul terjauh dari akar sebagai goal node dan solusi persoalannya adalah lintasan dari akar ke node tersebut. Setelah solusi persoalan ditemukan, lanjut ke tahap 6.
2) Tandai simpul yang sedang ditinjau dengan indikator sudah ditinjau, kemudian tentukan apakah state tersebut merupakan goal node atau bukan dengan cara memeriksa kedalaman simpul tersebut atau dengan memeriksa apakah atribut current dan hold dari state tersebut bernilai None. Jika state tersebut merupakan goal node, tandai dan tambahkan ke solusi persoalan, kemudian lanjutkan pencarian dengan kembali ke aras atas dan kembali ke tahap 1. Jika bukan goal node, lanjutkan ke tahap 3
3) Dengan menggunakan sebuah tabel, dalam hal ini tabel 2, tentukan apakah ada Tetrimino current atau hold pada daftar Tetrimino lanjutan untuk residual state yang sedang ditinjau; jika tidak ada, bunuh simpul, kemudian lakukan backtrack ke simpul di aras atas dan ulangi dari tahap 1. Jika ada, lanjut ke tahap 4
4) Untuk setiap Tetrimino pada hold dan current yang ada pada Tetrimino lanjutan state yang ditinjau, bangkitkan simpul anak berdasarkan gerakan-gerakan yang menyebabkan *line clear*
5) Ganti state yang sedang ditinjau dengan salah satu state anak, kemudian Kembali ke tahap 1.
6) Laksanakan semua gerakan solusi hingga mencapai state akhir, kemudian ulangi keseluruhan proses dari tahap 1 dengan informasi *next queue*, current, dan hold yang baru.

Agar lebih jelas, tinjau gambar 7. Pada state space tree tersebut, ditemukan 6 goal node. Akan tetapi, goal node yang dipilih adalah salah satu goal dengan residual 1. Dalam hal ini, akan dipilih goal node tersebut dengan lintasan 1O6’-6’I6’-6’L4’-4’J1. Lintasan tersebut dipilih dengan harapan Tetrimino yang dibangkitkan selanjutnya dapat melanjutkan kombo.

F. Alternatif Solusi

Solusi yang telah disebutkan diatas bisa dimodifikasi sehingga lebih dinamis, dalam hal ini, goal node dan fungsi pembangkit dievaluasi ulang setiap kali melakukan gerakan. Hal ini dilakukan karena dengan melakukan suatu gerakan, satu Tetrimino tambahan akan masuk ke *next queue* sehingga seluruh goal node yang telah ditentukan sebelumnya bisa dibangkitkan lagi dan diubah agar dapat menyesuaikan dengan Tetrimino selanjutnya.

IV. KESIMPULAN

Algoritma backtrack dapat digunakan untuk menentukan gerakan terbaik untuk strategi 4 wide dalam permainan Tetris modern. Dengan informasi berupa *next queue*, gerakan selanjutnya dapat dikomputasi hingga tiba di goal node dengan seluruh Tetrimino di *next queue*, *hold queue*, dan *current Tetrimino* habis. Komputasi kemudian diulangi dari awal dengan informasi *next queue* yang baru. Algoritma tersebut dapat dimodifikasi sehingga goal node dan fungsi pembangkit dievaluasi ulang setiap gerakan sehingga lebih dinamis.

PRANALA VIDEO YOUTUBE

Berikut merupakan pranala video Youtube terkait penjelasan beserta demo dari makalah ini:

https://youtu.be/YRXJj4UOZ1c

REFERENSI

[1] Tetris. 2022. About Tetris®. [online] Available at: <https://tetris.com/about-us> [Accessed 20 May 2022].

[2] Pemberton, S., 2022. The Computer as Extended Phenotype. [online] W3.org. Available at: <https://www.w3.org/2011/Talks/01-14-steven-phenotype/> [Accessed 20 May 2022].

[3] Munir, R., 2022. Algoritma Runut-balik (Backtracking) (Bagian 1). [online] Informatika.stei.itb.ac.id. Available at: <https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-backtracking-2021-Bagian1.pdf> [Accessed 20 May 2022].

[4] 2009. 2009 Tetris® Design Guideline. [ebook] Blue Planet Software, Inc. Available at: <https://www.dropbox.com/s/g55gwls0h2muqzn/tetris%20guideline%2

[5] davdav1233 and cosin307, 2020. 4-Wide (4列REN). [online] Four.lol. Available at: <https://four.lol/stacking/4-wide> [Accessed 20 May 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, 22 Mei 2022

Muhammad Naufal Satriandana 13520068