Penggunaan Algortima Greedy dan Algoritma Brute Force untuk Mencari Langkah Terbaik dalam Permainan Tetris

David Owen Adiwiguna / 13519169
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519169@stei.std.itb.ac.id

***Abstrak*—Untuk sebagian dari kita, game bukanlah hal yang bisa lepas begitu saja dari kehidupan. Hampir semua dari kita pasti pernah bermain game, dan mungkin salah satu dari game itu adalah Tetris. Tetris adalah game menyusun balok yang sudah ada sejak tahun 1984. Sampai saat ini, tetris masih banyak dimainkan dan digemari semua kalangan usia. Namun, kebanyakan dari orang yang memainkan tetris hanya bermain berdasarkan intuisi saja. Pada makalah ini, penulis akan mencari tahu bagaimana cara bermain tetris dengan lebih efektif dan tanpa mengandalkan intuisi sepenuhnya, hal itu akan dicapai dengan menggunakan pendekatan algoritma greedy dimana kemungkinan gerakan yang terjadi akan dinilai berdasarkan beberapa parameter yang ada, dan gerakan yang menunjukkan hasil terbaik berdasarkan parameter tersebut akan dipilih.**

Kata Kunci*—*tetris; balok; greedy; algoritma;

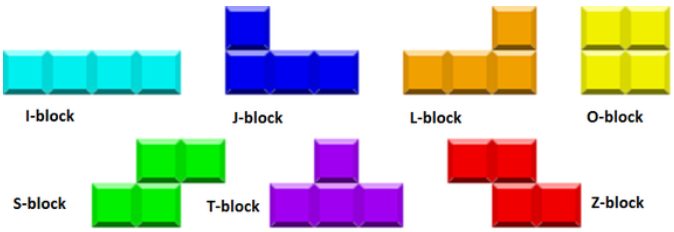
I. PENDAHULUAN

Tetris pertama kali dibuat pada tahun 1984 oleh Alexey Pajitnov, namun sudah ada banyak game tetris yang dimodernisasi tampilannya agar terlihat lebih bagus. Tetris termasuk ke dalam game yang paling sukses sepanjang masa, dimana Tetris berhasil terjual sebanyak 202 juta copy sampai saat ini. Meskipun telah dirilis berulang-ulang kali, pertaturan dan cara bermain Tetris tetaplah sama. Area permainan Tetris berukuran 20 balok tingginya dan 10 balok lebarnya. Tujuan dari permainan ini adalah mencapai skor setinggi mungkin sebelum kalah, atau cukup bertahan hidup lebih lama daripada lawan pemain dalam game mode versus. Pemain akan kalah jika area permainan Tetris terisi penuh sampai atas.

Pada awal permainan, balok-balok Tetris akan jatuh secara perlahan, namun semakin bertambahnya waktu dan naik level, kecepatan balok-balok berjatuhan akan semakin cepat sehingga diperlukan kemampuan yang lebih untuk tidak kalah.

Gambar 1.1 Layar Game Over Tetris
Sumber : https://tetris-full-screen-game.peatix.com

Untuk mencegah terjadinya game over, pemain harus mengurangi baris-baris balok yang ada dengan cara mengisi penuh 1 baris area permainan dengan balok, maka baris itu akan hilang.



Gambar 1.2 Balok yang ada pada permainan Tetris
Sumber : https://www.quora.com/What-are-the-different-blocks-in-Tetris-called-Is-there-a-specific-name-for-each-block

Tetris memiliki 7 jenis balok yang berbeda bentuk dan warnanya, player biasanya dapat melihat beberapa blok yang akan dia dapatkan selanjutnya, dan di beberapa game Tetris, pemain juga dapat meyimpan 1 blok untuk digunakan nanti. Kemunculan blok secara sekilas terlihat acak dan memungkinkan untuk terjadinya ketidak adilan, namun kemunculan block itu ternyata tidak serta-merta secara acak tetapi ada algoritma yang mengaturnya. Blok-blok tersebut kemunculannya tidak diatur satu per satu, tetapi ke-7 blok tersebut digabungkan dalam sebuah *sequence* dan urutan kemunculan blok dalam *sequence* tersebut diacak, lalu kemudian *sequence* tersebut dilanjut dengan *sequence-sequence* yang lainnya, dengan cara inilah semua pemain Tetris dapat mendapat perlakuan yang adil, sehingga kejuaraan Tetris bisa diadakan.

Kejuaraan Tetris diadakan dengan nama "Classic Tetris World Championship**"** kejuaraan ini mulai dilakukan setiap tahunnya sejak tahun 2010. Hadiah yang didapat dalam kejuaraan ini memang tidak sebesar kejuaraan *E-sport* lainnya, tetapi sudah termasuk cukup besar untuk game klasik seperti Tetris yaitu senilai 10.000 US Dollar yang dibagikan dari juara 1-16.

Nama Tetris itu sendiri diambil dari nama kejadian dimana pemain dapat menghilangkan 4 baris di area permainan sekaligus dengan menggunakan 1 blok, hal ini dapat diraih dengan cara menggunakan *i-block*.

Tujuan dari Tetris adalah mengumpulkan poin sebanyak-banyaknya, dan cara untuk mengumpulkan poin dengan jumlah terbanyak adalah dengan melakukan Tetris terus menerus. Dengan 1 kali melakukan Tetris, pemain akan mendapatkan 1200 poin, sementara jika pemain hanya dapat menghilangkan 1 baris di area permainan dalam 1 waktu, pemain hanya akan mendapat 30 poin.

Baru-baru ini, muncul sebuah game Tetris baru bernama *Tetris Effect* yang memiliki mekanik yang memungkinkan pemain untuk menghilangkan 16 baris di area permainan di waktu yang bersamaan.

Gambar 1.3 Logo Tetris Effect
Sumber : https://www.tetriseffect.game/

II. LANDASAN TEORI

A. Algoritma Greedy

Algoritma Greedy merupakan algoritma pencarian solusi paling efektif dengan cara memilih kondisi optimum lokal, yaitu solusi optimum yang bisa didapatkan 1 langkah ke depan, tentu saja cara ini belum pasti mendapatkan solusi yang optimal secara keseluruhan, namun karena mudahnya implementasi algoritma greedy dan hasilnya yang cukup mendekati solusi optimal, algoritma ini banyak digunakan dalam kehidupan sehari-hari.

Algoritma Greedy paling banyak dipakai untuk menyelesaikan masalah optimisasi, karena dinilai cukup baik dalam menyelesaikan masalah tersebut dengan prinsip "take what you can get now!". Algoritma ini berjalan dengan cara melakukan satu demi satu langkah, dan di setiap langkah tersebut dicari solusi yang paling optimalnya. Setelah solusi yang paling optimal untuk langkah tersebut ditemukan, baru kemudian dilanjutkan ke langkah selanjutnya dan dicari lagi solusi yang paling optimal untuk langkah tersebut, begitu seterusnya hingga mendapatnya solusi global.

Algortima Greedy disusun oleh beberapa elemen yang meliputi :

1. Himpunan kandidat, C.
   Himpunan ini berisi elemen-elemen pembentuk solusi. Contohnya adalah Simpul atau sisi di dalam sebuah graf, job, task, koin, benda, karakter, dll
2. Himpunan solusi, S.
   Himpunan ini berisi kandidat-kandidat yang terpilih sebagai solusi persoalan.
3. Fungsi Solusi
   Fungsi yang menentukan apakah solusi sudah didapat dari himpunan kandidat yang ada.
4. Fungsi Seleksi
   Fungsi yang pada setiap langkah memilih kandidat yang paling memungkinkan mencapai solusi optimal.
5. Fungsi Kelayakan
   Fungsi yang memeriksa apakah suatu kandidat yang telah dipilih dapat memberikan solusi yang layak, yaitu kandidat tersebut memenuhi segala syarat yang ada.
6. Fungsi Obyektif
   Fungsi yang memaksimumkan atau meminimumkan solusi.

Skema Umum pada Algoritma Greedy adalah sebagai berikut :

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }
Deklarasi
  x : kandidat
  S : himpunan_solusi

Algoritma:
  S ← {}      { inisialisasi S dengan kosong }
  while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)     { pilih sebuah kandidat dari C}
      C ← C – {x}        { buang x dari C karena sudah dipilih }
      if LAYAK(S ∪ {x}) then      { x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi }
          S ← S ∪ {x}     { masukkan x ke dalam himpunan solusi }
      endif
  endwhile
  {SOLUSI(S) or C = {} }

  if SOLUSI(S) then    { solusi sudah lengkap }
     return S
  else
     write('tidak ada solusi')
  endif
```

Gambar 2.1 Skema Umum Algortima Greedy
Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

Kelebihan dari Algoritma Greedy adalah kecepatannya dan hasilnya yang cukup mendekati solusi paling baik. Algoritma Greedy dapat mendapatkan solusi dengan cepat karena hanya perlu mencari solusi pada satu demi satu langkah, dan setelah langkah tersebut terlewati, semua Himpunan Solusi pada langkah sebelumnya dibuang dan tidak diipertimbangkan lagi.

Namun, meskipun solusi yang didapat akan mendekati solusi paling baik, solusi yang dihasilkan dari Algoritma Greedy masih kalah dengan algoritma-algoritma lain seperti Brute Force, A*, dll.

B. Algoritma Brute Force

Algoritma Brute Force adalah algoritma yang lebih mengandalkan tenaga daripada otak, dimana algoritma tersebut mencari semua kemungkinan solusi yang ada kemudian membandingkannya dan mencari solusi yang paling baik. Algortima ini memiliki prinsip "just do it" dimana algoritma tersebut akan melakukan segalanya untuk mencari solusi terbaik.

Algoritma Brute Force dipastikan akan mendapat solusi terbaik, namun akan memakan waktu yang cukup lama, pertambahan waktu ini akan bertambah secara eksponensial semakin banyak hal yang harus diuji.

Algoritma Brute Force merupakan algoritma yang implementasinya cukup mudah sehingga sering digunakan, cara pencarian solusinya juga jelas dan hampir semua persoalan bisa diselesaikan dengan algoritma Brute Force.

Namun, kelemahan dari Algoritma Brute Force adalah kompleksitasnya yang bertambah secara eksponensial sehingga memerlukan waktu yang lama untuk mencari solusi yang optimal jika n-nya besar. Maka dari itu, algoritma Brute Force lebih cocok untuk digunakan di masalah yang himpunan solusi yang harus diceknya sedikit.

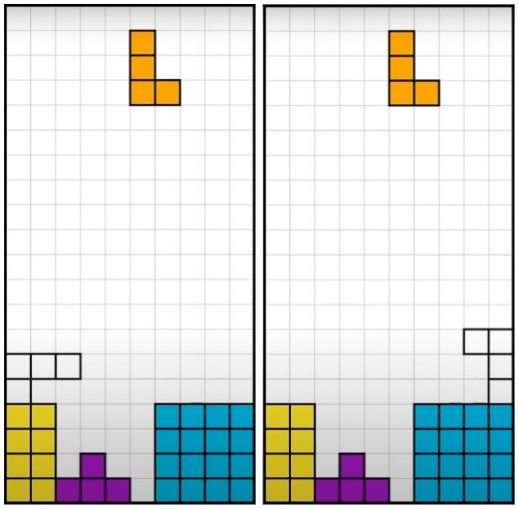
```
procedure CariElemenTerbesar(input a1, a2, ..., an : integer, output maks : integer)
{ Mencari elemen terbesar di antara elemen a1, a2, ..., an.
Elemen terbesar disimpan di dalam maks.
Masukan: a1, a2, ..., an
Luaran: maks
}
Deklarasi
 k : integer

Algoritma:
 maks←a1
 for k←2 to n do
   if ak > maks then
     maks←ak
   endif
 endfor
```

Gambar 2.2 Contoh Algoritma Brute Force untuk Pencarian Angka Terbesar
Sumber :
https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Brute-Force-(2021)-Bag1.pdf

III. IMPLEMENTASI

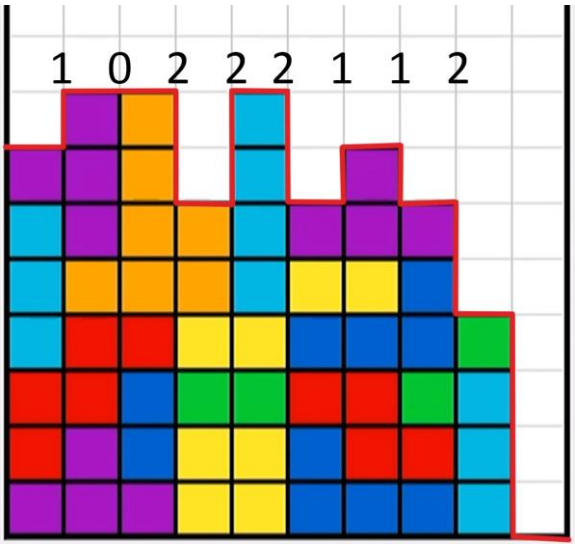
Untuk mencari solusi paling optimal untuk masing-masing langkah yang bisa dilakukan, pertama-tama kita harus mengetahui keseluruhan himpunan solusi yang bisa kita pilih, hal ini bisa kita lakukan dengan menggunakan Algoritma Brute Force, pada kasus permainan Tetris, himpunan solusi yang ada berarti kemungkinan penempatan balok baru ke arena permainan. Pemilihan penggunaan algoritma Brute Force untuk persoalan ini adalah agar dapat mendapatkan semua kemungkinan penempatan balok yang ada tanpa ada yang terlewat, hal ini memungkinkan untuk dilakukan dengan Brute Force karena jumlah penempatan balok yang mungkin sangat terbatas. Algoritma Brute Force akan membaca balok apa yang sedang dipegang oleh pemain dan mengeluarkan semua himpunan solusi yang mungkin terjadi.



Gambar 3.1 Algoritma Brute Force menentukan himpunan solusi yang mungkin
Sumber : Dokumentasi Penulis

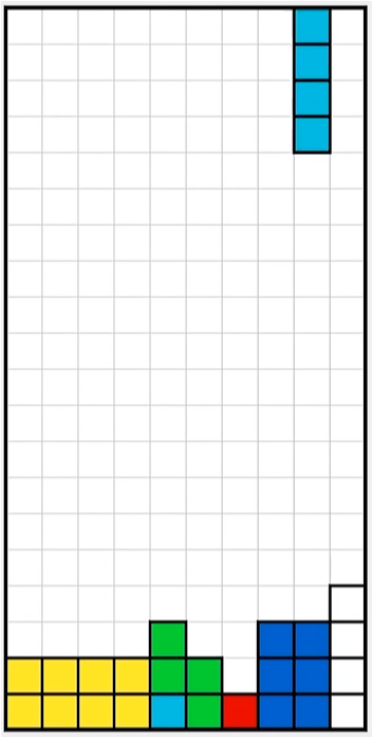
Untuk mengetahui langkah mana yang terbaik, harus ada parameter yang jelas untuk menilai semua langkah yang mungkin, yang kemudian membandingkan semua nilai yang ada dan memilih yang terbaik. Pada sebuah permainan Tetris, hal yang ingin kita lakukan adalah meminimalisir tinggi dari keseluruhan balok, selain itu kita juga ingin membuat balok yang sudah disusun di arena serata mungkin agar memudahkan kita untuk menghilangkan baris di arena permainan dan juga menghindari terbuatnya sebuah lubang di bawah balok yang sudah terpasang. 3 faktor tersebut lah yang akan menjadi penentu himpunan solusi mana yang paling efektif.

Nilai yang menentukan seberapa baik langkah tersebut akan ditentukan dengan rumus Nilai = 3*Jumlah Lubang dibawah balok + 1*Ketinggian + 2**Bumpiness* (Total perbedaan tinggi antara 2 Kolom Balok). Nilai Akhir setelah melakukan langkah tersebut akan dibandingkan dengan Nilai Akhir yang didapat dari kemungkinan peletakan balok yang lain. Jika Nilai Akhir setelah peletakkan balok lebih kecil daripada Nilai Awal sebelum peletakkan balok, berarti keadaan arena permainan setelah peletakkan balok lebih baik daripada sebelumnya.



Gambar 3.2 Cara Menghitung *Bumpiness*
Sumber : Dokumentasi Penulis

Jadi, pendekatan Algoritma Greedy ini adalah Greedy by Nilai, dimana untuk mendapatkan nilai yang optimal (dalam kasus ini berarti nilai yang paling kecil), langkah yang dilakukan harus mengurangi terbentuknya lubang, meminimalkan tinggi dari total balok, dan membuat tumpukan balok menjadi serata mungkin.



Gambar 3.2 Contoh Kasus
Sumber : Dokumentasi Penulis

Gambar di atas menunjukkan langkah paling optimal yang bisa dipilih, langkah tersebut dipilih karena tidak menimbulkan adanya lubang, tidak menambah ketinggian yang signifikan dan juga mengurani *Bumpiness* yang ada.

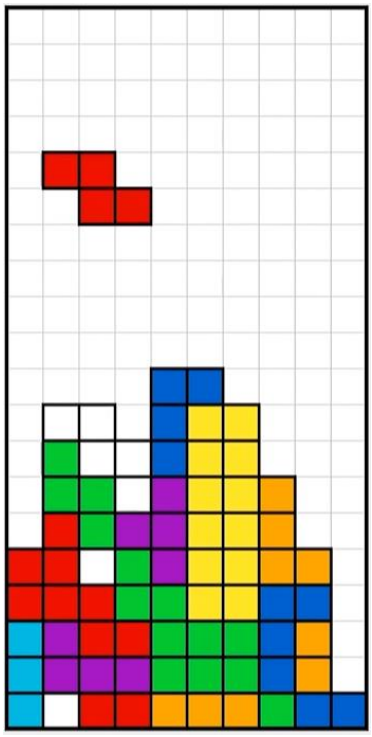
Nilai akhir = 3*0+1*4+2*(1+1+1+2+1) = 16
Nilai awal = 3*0+1*3+2*(1+1+1+2+3) = 19

Bisa dilihat pada hasil perhitungan di atas, Nilai Akhir (nilai setelah melakukan peletakan balok) lebih kecil daripada Nilai Awal, sehingga keadaan akhir dinilai lebih baik daripada keadaan awal, dan juga tidak ada himpunan solusi lain yang berhasil mendapatkan Nilai Akhir yang lebih kecil dari Himpunan Solusi terpilih

Cara ini bisa menghilangkan baris-baris karena dengan meletakan balok pada baris yang kosong, cenderung akan mengurangi *Bumpiness* yang ada.

Namun, kelemahan dari algoritma ini juga bisa dilihat pada gambar di atas, kelemahannya adalah algoritma ini tidak terpaku dengan mendapatkan skor sebanyak mungkin, algoritma hanya terpaku pada Nilai yang seminim mungkin, sehingga algoritma ini hanya terfokus akan cara agar tidak kalah dalam permainan Tetris.

Tetapi, dengan algoritma ini, pemain akan lebih sulit untuk kalah, sehingga dapat mencapai poin yang lebih tinggi seiring berjalannya waktu meskipun pertambahan poin cukup lambat, sehingga Algoritma ini cukup efektif.



Gambar 3.2 Contoh Kasus 2
Sumber : Dokumentasi Penulis

Ada kalanya Algoritma ini terpaksa membuat lubang karena langkah tersebut merupakan langkah paling optimal yang bisa dilakukan

Nilai akhir = 3*3+1*10+2*(4+1+2+1+2+2+4) = 51
Nilai awal = 3*2+1*10+2*(3+1+1+4+1+2+2+4) = 52

Meskipun terpaksa membuat lubang, bisa dilihat bahwa ternyata Nilai Akhir masih lebih kecil daripada Nilai Awal, dan langkah tersebut masih merupakan langkah yang menghasilkan Nilai Akhir paling kecil dibandingkan langkah-langkah lainnya.

IV. KESIMPULAN

Penyelesaian masalah ini dengan Algoritma Greedy menghasilkan hasil yang diharapkan dan cukup efektif, yaitu mendapat poin sebanyak mungkin, namun hal ini dicapai bukan dengan cara mendapatkan Tetris sebanyak mungkin, tetapi hanya dengan cara bertahan hidup dan tidak kalah.

Penyelesaian permainan Tetris secara optimal tidak dapat diselesaikan hanya menggunakan Algoritma dasar seperti Brute Force dan Greedy karena banyaknya pertimbangan yang ada, namun penggunaan kedua algoritma ini cukup untuk mendapatkan hasil yang cukup baik.

V. SARAN

Penyelesaian masalah ini masih bisa dikembangkan dan diperbaiki lebih lagi, terutama dalam bagian Algoritma Greedy. Sebaiknya, Algoritma mempertimbangkan terjadinya Tetris, karena perbedaan poin yang sangat jauh dibandingkan menghilangkan baris satu demi satu.

Selain itu, Algoritma ini belum mempertimbangkan kemungkinan untuk melihat balok-balok selanjutnya dan juga fitur untuk menyimpan balok yang sekarang sedang dipilih. Kedua hal ini dapat meningkatkan poin yang didapat secara cukup signifikan karena algoritma dapat melihat beberapa langkah ke depan, dan juga memiliki 2 pilihan balok yang tersedia, sehingga himpunan solusi yang dapat dipilih menjadi 2 kali lebih banyak.

VI. UCAPAN TERIMAKASIH

Penulis ingin berterima kasih kepada Tuhan Yesus Kristus karena hanya dengan rahmatNya, makalah ini bisa terbuat, penulis jugs ingin berterima kasih kepada keluarga dan teman-teman yang senantiasa menyemangati dalam penyusunan makalah ini. Penulis juga berterima kasih kepada Bapak Rinaldi Munir sebagai Dosen dari kelas Strategi Algoritma K4 yang sudah mengajarkan saya banyak sekali ilmu-ilmu baik itu terkait pelajaran dikelas, maupun pelajaran hidup.

Akhir kata, penulis ingin meminta maaf apabila makalah ini masih memiliki banyak kekurangan dan apabila ada salah-salah kata.

REFERENSI

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Brute-Force-(2021)-Bag1.pdf. Diakses pada tanggal 8 Mei 2021

[2] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Brute-Force-(2021)-Bag2.pdf. Diakses pada tanggal 8 Mei 2021

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf. Diakses pada tanggal 9 Mei 2021

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag2.pdf. Diakses pada tanggal 9 Mei 2021

[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag3.pdf. Diakses pada tanggal 9 Mei 2021

[6] https://chihsienyen.github.io/pdf/Tetris_AI.pdf. Diakses pada tanggal 9 Mei 2021

[7] https://hal.inria.fr/hal-00926213/document. Diakses pada tanggal 9 Mei 2021

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, 11 Mei 2021

David Owen Adiwiguna / 13519169