Penyelesaian Permainan Menyusun Balok dengan Algoritma *Greedy*

Faiz Muhammad Muflich / 13517093
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika ITB
Jl. Ganesha 10 Bandung 40132, Indonesia
Faizmuh26@gmail.com

***Abstract*—Banyak sekali permainan puzzle yang menggunakan balok sebagai media objek permainannya. Salah satu permainan yang cukup dikenal banyak orang yaitu permainan menyusun balok dengan beberapa informasi yang diberikan dan harus dapat menyusun balok-balok tersebut sesuai dengan informasi yang diberikan. Pada makalah ini akan dijelaskan mengenai penerapan salah satu jenis stategi algoritma yaitu *greedy* sebagai salah satu cara untuk menyelesaikan permainan tersebut.**

Keywords—puzzle balok; greedy; informasi; brute force;

I. PENDAHULUAN

Terdapat sebuah permainan puzzle yang cukup terkenal yaitu puzzle menyusun balok-balok berukuran 1x1x1 di atas suatu grid berukuran tertentu, misal n x m. Sehingga dalam setiap grid (i,j) terdapat $h_{i,j}$ balok tersusun di atasnya. Untuk setiap penyusunan terdapat aturan tertentu. Namun, informasi $h_{i,j}$ tersebut tidak diketahui sehingga pemain hanya diberi informasi mengenai grid tersebut dari sudut pandang depan, sudut pandang kiri, dan sudut pandang atas saja.

Dari sudut pandang depan akan diberikan informasi sebanyak m buah (karena kolom grid berjumlah m). Untuk setiap kolom akan diberikan informasi mengenai tinggi maksimum balok yang dapat terlihat dari posisi tersebut. Maka nantinya akan ada h_1, h_2, …, h_m nilai yang diketahui oleh pemain.

Dari sudut pandang kiri akan diberikan informasi sebanyak n buah (karena baris grid berjumlah n). Untuk setiap baris akan diberikan informasi mengenai tinggi maksimum balok yang dapat terlihat dari posisi tersebut. Maka nantinya akan ada h_1, h_2, …, h_n nilai yang diketahui oleh pemain.

Dari sudut pandang atas akan diberikan informasi berupa matrik berukuran n x m. Untuk setiap elemen matriks tersebut $a_{i,j}$ bernilai 0 atau 1. Jika $a_{i,j}$ bernilai 1 maka pada cell tersebut terdapat balok dengan ketinggian $h_{i,j} > 0$. Sedangkan untuk $a_{i,j}$ bernilai 0 maka pada cell tersebut tidak terdapat balok atau dengan kata lain $h_{i,j} = 0$.

Tugas dari pemain adalah menyusun balok-balok tersebut sehingga informasi yang diberikan oleh permainan tersebut tidak kontradiksi dengan hasil penyusunan pemain.

Dari permainan ini kemudian timbul permasalahan apakah ada secara pasti algoritma yang dapat menyelesaikan permainan tersebut tanpa harus mencobanya secara satu-persatu secara bruteforce? Karena jika permainan tersebut dimainkan dengan cara mencoba satu persatu maka jelas permainan tersebut menjadi sukar untuk dimainkan. Sehingga penulis tertarik untuk membahas penyelesaian permasalahan ini dengan pendekatan algoritma *greedy.*

II. DASAR TEORI

A. *Algoritma Brute Force*

Algoritma Brute Force merupakan algoritma untuk mencari solusi dari suatu permasalahan dengan menggunakan pendekatan yang sederhana dan paling mudah unutk diimplementasikan. Lebih dikenal dengan istilah lempang atau straight forward, sehingga algoritma Brute Force dapat mencari solusi dari permasalahan tersebut dengan lengkap, pasti kebenarannya dan sederhana. Namun, kompleksitas dari algoritma brute force bisa sangat besar. Sehingga sebaiknya penggunaan algoritma brute force tidak untuk *constraint* yang besar.

Penyelesaian algoritma Brute Force biasanya didasarkan pada pernyataan masalah. Beberapa contoh kasusnya adalah permasalahan *searching.* Baik untuk mencari elemen terbesar atau elemen terkecil. Kemudian mencari indeks sebuah nilai pada array, mencari elemen terkecil ke k, dan permutasi.

Algoritma Brute Force memiliki beberapa karakteristik yang sangat khas dan mudah untuk dikenali. Yang pertama, algoritma ini bukanlah algoritma yang cerdas dan mangkus atau efisien, karena algoritma ini membutuhkan kompleksitas waktu yang sangat besar dalam penyelesaiannya jika dibandingkan dengan algoritma lainnya, sehingga waktu yang dibutuhkan juga berbanding lurus dengan kompleksitas waktu penyelesaiannya. Yang kedua, algoritma brute force biasa digunakan sebagai pembanding keefisienan algoritma lainnya. Karena algoritma brute force biasa dianggap sebagai basis dari suatu algoritma sehingga mudah sekali untuk dikembangkan algoritma lain yang kemudian dibandingkan dengan algoritma brute force teresebut dalam urusan kompleksitasnya. Hal ini lumrah dilakukan di perlombaan-perlombaan semacam *competitive programming.*

Secara sistematis, cara kerja dari algoritma brute force atau yang biasa disebut *exhaustive search* adalah :

1. Enumerasi adalah mengenumerasikan list dari setiap kemungkinan solusi yang ada.
2. Evaluasi adalah menyeleksi setiap kemungkinan solusi satu persatu kemudian membandingkan solusi terbaik yang ditemukan dan menyimpannya.

3. Bila pencarian solusi berakhir, solusi terbaik ditemukan.

B. Algoritma Greedy

Algoritma greedy adalah salah satu strategi pencarian solusi permasalahan yang sering digunakan untuk menangani kasus-kasus optimasi yang dalam pencarian solusinya selalu menghasilkan nilai solusi lokal yang maksimum ataupun minimum dari sebuah permasalahan yang diberikan.

Dalam penggunaan algoritma *greedy* ini dikenal dua terminologi dasar yang mendasari penerapan algoritma ini yaitu solusi layak dan Solusi optimum. Solusi layak adalah solusi yang membatasi nilai sehingga nilai yang bersangkutan memenuhi syarat tertentu yang diberikan. Sedangkan solusi optimum adalah solusi sebenarnya dari permasalahan yang diberikan, solusi optimum didapat dari membandingkan semua solusi layak yang ada. Algoritma ini memiliki ciri khas yang menjadikan namanya *greedy* yaitu untuk setiap langkah mengambil langkah yang paling optimal, tanpa mempertimbangkan konsekuensi pengambilan langkah tersebut di kemudian waktu.

Algoritma greedy merupakan salah satu algoritma yang dalam pencarian solusi terbaiknya dipercayakan pada iterasi langkah satu persatu dan diharapakan dalam iterasi tersebut setiap langkah mendapati hasil yang terbaik. Setiap langkah harus merupakan solusi lokal yang terbaik dan sebagai konsekuensinya tidak dapat diubah ketika langkah selanjutnya telah dijalankan. Oleh karena itu, karena solusi diambil dari langkah perlangkah yang menghasilkan solusi lokal, solusi global yang terjadi ada kalanya tidak menjamin solusi global yang terbaik. Maka dari itu dalam penggunaan algoritma *greedy* perlu dipastikan lagi apakah algoritma yang dibuat terbukti dapat menyelesaikan permasalahan tersebut.

Prinsi dari algoritma *greedy* ada beberap elemen utama yang membentuknya yaitu:

1. Himpunan kandidat. Himpunan ini berisi semua kemungkinan solusi-solusi lokal yang dapat membangun solusi global. Pada setiap langkah, sebuah kandidat diambil dari himpunan ini untuk dilakukan pengecekan solusi lokal kemudian dipindahkan ke dalam himpunan solusi yang membangun solusi global dari permasalahan yang terkait.
2. Himpunan solusi. Himpunan ini berisi hasil penerapan strategi greedy yang ditawarkan pada setiap langkah. Hasil tersebut yang kemudian kita kenal sebagai optimum lokal. Setiap elemen himpunan ini dapat dikatakan sebagai pilihan solusi yang terbaik pada saat pilihan tersebut diambil dan dimasukkan dari himpunan kandidat ke dalam himpunan solusi. Himpunan solusi ini yang nantinya akan membangun solusi global terkait.
3. Fungsi seleksi. Fungsi seleksi adalah fungsi yang digunakan pada setiap langkah pencarian solusi untuk memilih elemen dari himpunan kandidat yang mempunyai nilai paling optimum sehingga mempunyai peluang untuk membentuk solusi global yang paling baik.
4. Fungsi kelayakan. Fungsi kelayakan adalah fungsi yang membatasi himpunan solusi. Hal ini karena fungsi kelayakan memeriksa kelayakan dari elemen himpunan kandidat yang telah dipilih oleh fungsi seleksi. Hal ini diperlukan karena dalam algoritma greedy, selalu terdapat syarat yang tidak boleh dilanggar dalam pengambilan langkah solusi. Fungsi kelayakan yang bertanggung jawab dalam membatasi kandidat sehingga tidak menyalahi batasaan yang telah diberikan.
5. Fungsi obyektif. Fungsi obyektif merupakan fungsi yang mengeluarkan solusi global. Fungsi ini akan menyeleksi solusi yang paling optimal dari himpunan solusi yang kemudian menjadi jawaban dari permasalahan tersebut.

Kelebihan dari algoritma greedy adalah kecepatan yang dimilikinya terkait kompleksitas waktunya yang cepat. Dan juga pendekatan yang mudah dipahami oleh bahasa manusia. Algoritma greedy juga merupakan algoritma yang jika terbukti kebeneran solusinya maka pasti mudah sekali untuk dipahami cara kerja dari algoritma tersebut dalam mencari solusi. Selain itu, algoritma greedy tidak memakan banyak memory untuk kebanyakan kasus, dan algoritma yang jauh lebih efektif dari pada algoritma lempeng brute force.

Akan tetapi algoritma greedy terkadang sulit untuk didesain dan jika sudah ditemukan pendekatannya harus diuji terlebih dahulu sehingga dapat dipastikan apakah algoritma greedy yang dibuat dapat menjamin solusi global dari permasalahan yang diberikan. Meskipun demikian untuk persoalan yang tidak dapat diselesaikan dengan algoritma greedy, penggunaan algoritma ini akan mendekatkan pembuat dalam pencarian solusi yang lebih optimal karena dengan menerapkan algoritma greedy dapat diketahui beberapa batasan yang harus diatasi dan bagaimana solusi lain yang dapat dibuat sehingga permasalahan dapat diselesaikan dengan lebih tepat.

III. Strategi Penyelesaian Permaian Menyusun Balok

Sebelum menyusun strategi dalam menyelesaikan permasalahan tersebut sebelumnya diberikan terlebih dahulu contoh cara memainkan permainan ini.

Misalkan terdapat grid berukuran 3 x 7. Diketahui dari tampak depan tinggi balok yang dapat terlihat adalah

2 3 0 0 2 0 1

Artinya pada kolom ke-1 jika dilihat dari depan tinggi balok tertinggi yang terlihat adalah 2. Karena balok berukuran 1x1x1 maka pada kolom 1 terdapat tumpukan balok paling tinggi adalah 2 buah balok pada paling tidak satu cell di kolom tersebut.

Pada kolom ke-2 terdapat tumpukan balok tertinggi sejumlah 3 buah pada paling tidak satu cell di kolom kedua. Pada kolom ke-3 tidak ada tumpukan balok yang terlihat. Pada kolom 4 tidak terdapat tumpukan balok. Pada kolom ke-5 terdapat tumpukan balok tertinggi yang terlihat yaitu 2 buah pada paling tidak satu cell di kolom ke-5. Berikut seterusnya.

Kemudian diketahui informasi tampak kiri tinggi balok yang dapat terlihat adalah

2 1 3

Artinya pada baris pertama jika dilihat dari tampak kiri balok tertinggi yang terlihat sebanyak 2 buah tumpukan. Baris ke-2 balok tertinggi yang terlihat sebanyak 1 buah tumpukan

dan baris ke-3 tumpukan balok tertinggi yang terlihat adalah sebanyak 3 buah tumpukan balok.

Kemudian informasi terakhir yang diterima yaitu dari tampak atas adalah

1 0 0 0 1 0 0

0 0 0 0 0 0 1

1 1 0 0 0 0 0

Artinya pada cell (1,1) terdapat tumpukan balok, pada cell (1,2) tidak terdapat balok sama sekali. Pada cell (1,3) tidak terdapat tumpukan balok. Begitu seterunya hingga pada cell (3, 7) tidak terdapat tumpukan balok. Atau secara mudahnya 1 berarti terdapat tumpukan balok dan 0 berarti tidak ada tumpukan balok sama sekali.

Kemudian dari informasi yang diberikan pemain harus dapat menyusun balok-balok berukuran 1x1x1 pada grid 3 x 7 (untuk kasus ini) sehingga hasil penyusunan balok-balok tersebut tidak menyalahi informasi-informasi yang telah diberikan.

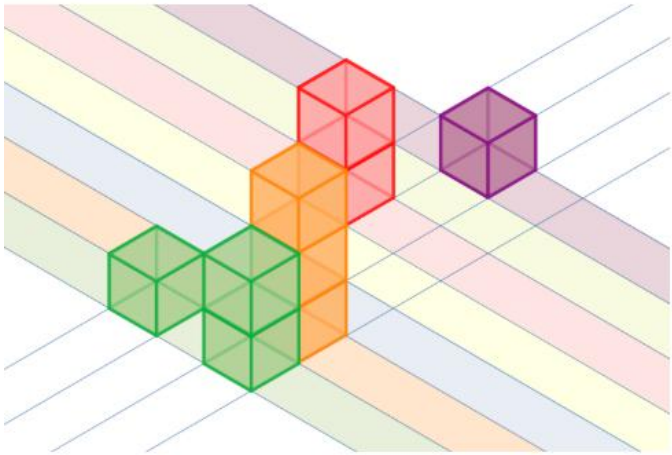
Salah satu solusi yang mungkin untuk kasus di atas adalah dengan menyusun balok-balok pada grid sehingga tersusun seperti berikut

1 0 0 0 2 0 0

0 0 0 0 0 0 1

2 3 0 0 0 0 0

Maksud dari gambar di atas adalah untuk setiap nilai $h_{i,j}$ pada cell (i,j) tersebut merepresentasikan jumlah tumpukan balok. Sebagai contoh pada cell (3,2) terdapat tumpukan 3 balok.



Ilustrasi1.*(sumber : codeforces)*

Ilustrasi1 menggambarkan solusi permasalahan yang diminta di atas.

Solusi dari persoalan yang diminta tidak selalu unik. Mungkin saja terdapat lebih dari satu solusi yang mungkin. Maka jika dimainkan langsung begitu saja akan terasa susah kemudian akan disusun strategi penyelesaiannya dengan algorimta brutefroce dan greedy.

A. Stategi Penyelesaian dengan Brute Force

Dengan contoh kasus yang sama akan diimplementasikan algoritma bruteforce untuk memecahkan masalah ini.

Diketahui

Tampak depan : 2 3 0 0 2 0 1

Tampak kiri : 2 1 3

Tampak atas :

1 0 0 0 1 0 0

0 0 0 0 0 0 1

1 1 0 0 0 0 0

Dengan algoritma brute force solusi dapat tercapai dengan mengimplementasikan satu persatu percobaan pada matriks yang diberikan.

Langkah pertama dimulai dari tampak depan. Untuk setiap cell pertama yang terdapat tumpukan baloknya dalam setia kolom atur tinggi balok sebanyak tinggi balok tertinggi yang dapat terlihat dari tampak depan kolom tersebut. Sehingga Grid yang sebelumnya

1 0 0 0 1 0 0

0 0 0 0 0 0 1

1 1 0 0 0 0 0

Akan berubah menjadi

2 0 0 0 2 0 0

0 0 0 0 0 0 1

1 3 0 0 0 0 0

Setelah itu kemudian matriks tersebut kita periksa apakah sudah memenuhi informasi yang diberikan atau masih menyalahi. Setelah dilakukan pengecekan ternyata matriks yang terjadi tidak menyalahi informasi yang diberikan sehingga solusi ditemukan.

Akan diberikan contoh kasus lain sebagai berikut :

Diketahui :

Ukuran grid : 4 x 5

Tampak depan : 3 5 2 0 4

Tampak kiri : 4 2 5 4

Tampak atas :

0 0 0 0 1

1 0 1 0 0

0 1 0 0 0

1 1 1 0 0

Solusi :

Langkah pertama dimulai dari tampak depan. Untuk setiap cell pertama yang terdapat tumpukan baloknya dalam setia kolom atur tinggi balok sebanyak tinggi balok tertinggi yang dapat terlihat dari tampak depan kolom tersebut. Sehingga Grid yang sebelumnya

0 0 0 0 1

1 0 1 0 0

0 1 0 0 0

1 1 1 0 0

Akan berubah menjadi

0 0 0 0 4

3 0 2 0 0

0 5 0 0 0

1 1 1 0 0

Setelah langkah pertama susunan balok sudah memenuhi jika dilihat dari tampak depan, namun jika dilihat dari tampak kiri masih salah sehingga dilakukan langkah berikutnya yaitu untuk setiap baris jika ditemukan balok yang mempunyai tinggi lebih dari tampak kiri yang seharusnya maka, geser balok tersebut ke baris lain dengan kolom yang sama. Dan jika pada baris tersebut belum terdapat balok dengan tinggi seperti yang terlihat dari tampak kiri maka atur cell pertama yang terdapat baloknya setinggi informasi tersebut. Jika penempatan tinggi balok baru menyalahi informasi tampak depan, maka geser balok tersebut ke kolom lainnya.

Maka grid akan berubah menjadi

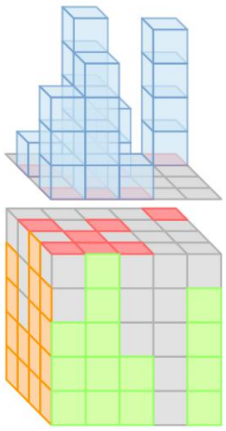
0 0 0 0 4

1 0 2 0 0

0 5 0 0 0

3 4 1 0 0

Hal tersebut terus diulang jika grid masih menyalahi informasi tampak depan, tampak kiri, dan tampak atas yang telah diberikan sampai penyusunan tidak menyalahi informasi yang diberikan.



Ilustrasi2 (*sumber : codeforces0*

Ilustrasi2 menggambarkan solusi yang berhasil ditemukan dari strategi brute force yang telah diterapkan.

B. Strategi Penyelesaian dengan Algoritma Greedy

Setelah dilakukan implementasi dengan menggunakan algoritma bruteforce. Maka kemudian terdapat pola yang bisa tersadari sehingga pendekatan dengan algoritma greedy jauh lebih menjamin tercapainya solusi optimal yang mungkin tanpa harus mencoba menyusun balok tersebut satu persatu.

Algoritma greedy yang dapat digunakan untuk menyelesaikan permasalahan ini adalah sebagai berikut :

Jika diketahui grid berukuran 4 x 5 dengan informasi tambahan :

Tampak depan : 3 5 2 0 4

Tampak kiri : 4 2 5 4

Tampak atas :

0 0 0 0 1

1 0 1 0 0

0 1 0 0 0

1 1 1 0 0

Catat a_i sebagai nilai dari Tampak depan kolom ke-i. Catat b_i sebagai nilai dari tampak kiri baris ke-i. Kemudian lakukan iterasi pada matriks tampak atas. Jika nilai dari cell pada baris i dan kolom j tersebut bukan 0, ubah nilai tersebut menjadi nilai minimum antara b_i dengan a_j. Maka dengan demikian hasil dari penyusunana balok tidak akan menyalahi informasi yang diberikan.

Solusi :

	a	3	5	2	0	4
b						
4		0	0	0	0	1
2		1	0	1	0	0
5		0	1	0	0	0
4		1	1	1	0	0

Akan berubah menjadi

	a	3	5	2	0	4
b						
4		0	0	0	0	4
2		2	0	2	0	0
5		0	5	0	0	0
4		3	4	2	0	0

Perhatikan bahwa hasil tersebut tidak menyalahi baik dari tampak depan, tampak samping, maupun tampak atas.

Maka dengan demikian algoritma greedy yang sudah disebutkan dapat digunakan untuk menyusun persoalan puzzle menyusun balok.Adapun demikian, algoritma ini dapat dibuktikan mengapa selalu memberikan jawaban yang optimal. Hal ini sangat jelas dapat diterangkan bahwa untuk setiap cell, tinggi maksimum yang dapat dimilikinya adalah nilai minimum antara a_i dengan b_i. Karena jika cell tersebut diatur tingginya lebih dari minimum antara a_j atau b_i maka cell tersebut sudah menyalahi batasan yang diberikan oleh salah satu di antara a_j atau b_i tersebut.

IV. Simpulan

Algoritma greedy terbukti sangat ampuh dalam menyelesaikan permainan menyusun balok-balok dalam suatu grid yang diberikan batasan informasi tertentu. Sebagai perbandingan jika dibandingkan dengan pendekatan brute force pemain akan kerepotan dalam menyusun balok-balok tersebut. Hal ini karena pemain harus mempertimbangkan tampak depan, tampak samping satu persatu apakah menyalahi atau tidak. Sedangkan dengan adanya strategi greedy yang diberikan pemain dapat menyusun balok-balok tersebut dengan cepat sekali bahkan dapat dilakukannya tanpa berpikir berat cukup menghitung nilai terkecil saja antara informasi tampak depan dengan tampak kiri untuk setiap cell.

Ucapan Syukur

Pertama, penulis ingin mengucapkan syukur yang teramat sangat kepada Allah SWT atas rahmatNya sehingga makalah ini dapat terselesaikan dengan baik. Penulis juga berterimakasih kepada Bapak Rinaldi Munir selaku dosen mata kuliah IF2211 Strategi Algoritma sekaligus pemantik semangat penulis dalam menulis makalah ini.

References

[1] Munir, Rinaldi. 2018. *Strategi Algoritma.* Bandung: Institut Teknologi Bandung.

[2] https://codeforces.com/contest/1153/problem/B diakses 25 April 2019 ,pukul 02.00.

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Algoritma-Greedy-(2019).pdf diakses pada 25 April 2019, pukul 03.00.

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, 26 April 2019

Faiz Muhammad Muflich / 13517093