Penerapan Algoritma Bruteforce dan DFS pada Permainan Hashi (Bridges)

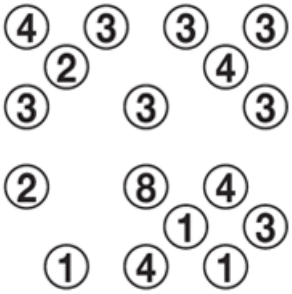
Dewana Gustavus Haraka Otang - 13521173
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13521173@std.stei.itb.ac.id

***Abstrak*—Permainan hashi adalah sebuah permainan logika yang dimainkan di atas sebuah papan. Dalam permainan ini pemain diberikan sejumlah pulau dengan angka di setiap pulau. Pemain akan menghubungkan pulau-pulau tersebut dengan jembatan, dan terdapat beberapa aturan. Aturan-aturan tersebut meliputi membatasi jumlah jembatan yang terhubung dengan sebuah pulau, larangan adanya jembatan yang bersilangan, dan jembatan tidak boleh melewati pulau lain. Tujuan permainan hashi adalah membuat jembatan-jembatan yang sesuai sehingga jumlah jembatan yang terhubung ke sebuah pulau sama dengan angka yang terdapat pada pulau tersebut, serta memastikan semua pulau terhubung melalui jembatan yang telah dibuat.**

Kata Kunci—hashi, bridges, bruteforce, DFS

I. PENDAHULUAN

Hashi adalah permainan logika yang diperkenalkan oleh perusahaan asal jepang yaitu Nikoli. Permainan hashi dimainkan oleh 1 orang pemain dalam sebuah papan. Dalam permainan hashi pemain akan diberikan beberapa pulau dengan angka yang terdapat di setiap pulau.

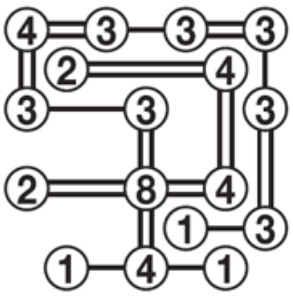


Gambar 1. Contoh keadaan awal permainan hashi

Pemain akan diberikan kesempatan untuk menyambungkan 2 buah pulau yang berada sejajar dalam garis horizontal atau garis vertikal. Dalam pembuatan jembatan ada beberapa restriksi yaitu:

1. Dalam sebuah pasangan pulau, hanya boleh ada minimal 0 jembatan dan maksimal 2 jembatan
2. Tidak boleh ada 2 buah jembatan yang bersilangan (jembatan tidak boleh melewati jembatan lain)
3. Apabila sebuah jembatan menghubungi pulau a dan pulau b, tidak boleh ada pulau c yang berada diantara pulau a dan pulau b (jembatan tidak boleh melewati pulau)

Tujuan akhir dari permainan hashi adalah dengan membuat jembatan sedemikian sehingga jumlah jembatan yang terhubung ke sebuah pulau sama dengan angka yang terdapat pada pulau tersebut, dan semua pulau terkoneksi melalui jembatan yang telah dibuat.



Gambar 2. Contoh keadaan akhir permainan hashi

II. TEORI DASAR

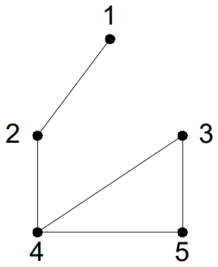
A. Bruteforce

Bruteforce adalah salah satu strategi algoritma yang menyelesaikan masalah menggunakan pendekatan yang lempeng (straight forward). Langkah penyelesaian masalah dengan algoritma bruteforce biasanya caranya jelas dan sangat sederhana.

Langkah penyelesaian masalah dengan algoritma bruteforce tidak terlalu berfokus pada kecepatan pencarian solusi dan akan mencoba segala kemungkinan jawaban sehingga bukan merupakan pilihan strategi yang cocok apabila memperhatikan kecepatan mencari solusi dan sering disebut sebagai algoritma naif.

Implementasi pencarian solusi untuk algoritma bruteforce biasanya mudah karena tidak terlalu memperhatikan kecepatan pencarian solusi. Karena implementasinya yang mudah, hampir seluruh permasalahan dapat diselesaikan dengan strategi bruteforce.

B. Graf



Gambar 3. Contoh graf

Graf adalah struktur data yang digunakan untuk merepresentasikan objek-objek diskrit beserta hubungan antara objek-objek tersebut. Graf umumnya terdiri atas 2 komponen yaitu:

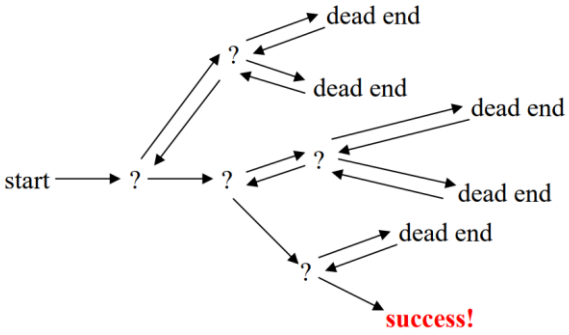
1. Simpul/Node/Vertex

   Simpul biasanya direpresentasikan menggunakan titik yang menyatakan sebuah objek diskrit.

2. Sisi/Edge

   Sisi biasanya direpresentasikan menggunakan garis yang menyatakan hubungan antara 2 buah objek.

C. DFS



Gambar 4. Contoh traversal menggunakan algoritma DFS

Depth First Search (DFS) adalah salah satu algoritma graf yang dapat dipakai untuk melakukan traversal. DFS merupakan proses traversal dengan memakai pendekatan pencarian mendalam. DFS biasanya dipakai untuk melakukan uninformed search yaitu proses pencarian tanpa diberikan informasi tambahan.

Pada algoritma DFS, ketika berada di sebuah simpul, kita akan mengunjungi simpul tetangga yang belum pernah dikunjungi dan proses tersebut akan dilakukan terus menerus sampai sudah tidak ada lagi simpul tetangga yang belum terkunjungi. Ketika berada pada sebuah simpul yang seluruh tetangganya sudah dikunjungi sehingga tidak dapat melanjutkan pencarian, algoritma DFS akan melakukan runut-balik (backtrack) ke simpul sebelumnya. Proses pencarian pada algoritma DFS akan berakhir ketika tidak ada lagi simpul yang dapat dicapai namun belum terkunjungi.

III. IMPLEMENTASI

A. Merepresentasikan permasalahan dan solusi

Agar memudahkan dalam mencari solusi representasi permasalahan dan solusi akan diubah menjadi seperti berikut:

1. Pulau

   Pulau yang diberikan pada awal permainan hashi akan direpresentasikan dalam bentuk tupel berisi tiga elemen yaitu

   $$< x, y >$$

   atau

   $$< x, y, z >$$

   Dimana x adalah posisi pulau dalam sumbu x, y adalah posisi pulau dalam sumbu y, dan z adalah angka yang terdapat pada pulau yang merepresentasikan berapa banyak jembatan yang harus terhubung dengan pulau tersebut.

   Kumpulan pulau yang diberikan pada awal permainan akan disimpan dalam sebuah larik yang berisi pulau.

2. Jembatan

   Jembatan yang akan dibangun akan direpresentasikan dalam bentuk tupel berisi empat elemen yaitu

   $$< x_1, y_1, x_2, y_2 >$$

   Dimana x_1 adalah posisi pulau 1 dalam sumbu x, y_1 adalah posisi pulau 1 dalam sumbu y, x_2 adalah posisi pulau 2 dalam sumbu x, dan y_2 adalah posisi pulau 2 dalam sumbu y.

   Ditambahkan juga restriksi tambahan untuk representasi jembatan agar memudahkan pencarian solusi yaitu nilai $x_1 \leq x_2$, dan nilai $y_1 \leq y_2$.

3. Solusi

   Solusi yang akan diberikan pemain akan direpresentasikan sebagai larik berisi seluruh jembatan yang digunakan sebagai solusi akhir permainan.

B. Pencarian solusi

Pencarian solusi permainan hashi dengan menggunakan strategi bruteforce akan terdiri dari beberapa tahap yaitu:

1. Mencari seluruh kandidat jembatan yang dapat menghubungkan pulau secara horizontal
2. Mencari seluruh kandidat jembatan yang dapat menghubungkan pulau secara vertikal
3. Membangkitkan seluruh kemungkinan kombinasi memilih jembatan yang akan dipakai
4. Validasi solusi untuk setiap kemungkinan kombinasi

C. Mencari seluruh kandidat jembatan horizontal

Untuk mencari seluruh kandidat jembatan horizontal yang mungkin dapat dilakukan dengan cara mengurutkan larik pulau yang diberikan. Pengurutan dilakukan dengan prioritas pertama

diurutkan berdasarkan posisi pada sumbu y diurutkan dari kecil ke besar, lalu apabila terdapat dua buah pulau atau lebih dengan posisi sumbu y yang sama maka prioritas pengurutan kedua adalah diurutkan berdasarkan posisi pada sumbu x diurutkan dari kecil ke besar. Dapat dipastikan tidak ada dua buah pasang pulau berbeda dengan posisi sumbu x dan sumbu y yang sama karena pulau sudah dipastikan tidak dapat tumpang tindih pada awal permainan.

Setelah dilakukan pengurutan, apabila dua buah pulau bersebelahan pada larik yang sudah terurut, maka dapat dipastikan apabila posisi sumbu y kedua pulau adalah sama berarti kedua pulau sejajar secara vertikal dan bisa didapat kandidat jembatan horizontal baru dengan dua buah pasang pulau tersebut. Seluruh kandidat jembatan horizontal dapat dicari dengan cara mengiterasi seluruh indeks pada larik dari 0 sampai (jumlah pulau – 1) dan cek apakah pulau pada indeks i dan indeks i+1 sejajar secara vertikal.

Dengan cara mengiterasi seperti itu kita juga telah memenuhi restriksi tidak boleh ada jembatan yang melewati pulau, karena kita hanya mencari jembatan yang menghubungkan dua buah pulau yang sejajar secara vertikal dan posisinya bersebelahan. Setiap kandidat jembatan horizontal yang ditemukan akan dimasukkan ke dalam sebuah larik jembatan horizontal sebanyak dua kali karena pada permainan hashi jembatan yang menghubungkan sebuah pasang pulau dapat dibangun sebanyak 2 jembatan.

D. Mencari seluruh kandidat jembatan vertikal

Untuk mencari seluruh kandidat jembatan vertikal yang mungkin dapat dilakukan dengan cara yang mirip dengan pencarian kandidat jembatan horizontal. Kita akan mengurutkan larik pulau yang diberikan. Pengurutan dilakukan dengan prioritas pertama diurutkan berdasarkan posisi pada sumbu x diurutkan dari kecil ke besar, lalu apabila terdapat dua buah pulau atau lebih dengan posisi sumbu x yang sama maka prioritas pengurutan kedua adalah diurutkan berdasarkan posisi pada sumbu y diurutkan dari kecil ke besar. Dapat dipastikan tidak ada dua buah pasang pulau berbeda dengan posisi sumbu x dan sumbu y yang sama karena pulau sudah dipastikan tidak dapat tumpang tindih pada awal permainan.

Setelah dilakukan pengurutan, apabila dua buah pulau bersebelahan pada larik yang sudah terurut, maka dapat dipastikan apabila posisi sumbu x kedua pulau adalah sama berarti kedua pulau sejajar secara horizontal dan bisa didapat kandidat jembatan vertikal baru dengan dua buah pasang pulau tersebut. Seluruh kandidat jembatan vertikal dapat dicari dengan cara mengiterasi seluruh indeks pada larik dari 0 sampai (jumlah pulau – 1) dan cek apakah pulau pada indeks i dan indeks i+1 sejajar secara horizontal.

Dengan cara mengiterasi seperti itu kita juga telah memenuhi restriksi tidak boleh ada jembatan yang melewati pulau, karena kita hanya mencari jembatan yang menghubungkan dua buah pulau yang sejajar secara horizontal dan posisinya bersebelahan. Setiap kandidat jembatan vertikal yang ditemukan akan dimasukkan ke dalam sebuah larik jembatan vertikal sebanyak dua kali karena pada permainan hashi jembatan yang menghubungkan sebuah pasang pulau dapat dibangun sebanyak 2 jembatan.

E. Membangkitkan seluruh kombinasi jembatan

Setelah mencari seluruh kandidat jembatan horizontal dan vertikal, kita perlu menentukan jembatan mana saja yang akan kita pakai sebagai kandidat solusi kita. Penentuan jembatan yang dipakai dapat dilakukan dengan strategi bruteforce yaitu dengan cara membangkitkan seluruh kombinasi pemakaian jembatan.

Kombinasi jembatan yang dipakai akan direpresentasikan sebagai dua buah larik yang berisi bilangan bulat 0 atau 1 untuk masing-masing jembatan horizontal dan vertikal, dengan 0 berarti jembatan tersebut tidak dipakai, dan 1 berarti jembatan tersebut dipakai.

F. Validasi solusi untuk setiap kemungkinan kombinasi

Setelah seluruh kombinasi jembatan yang akan dipakai dibangkitkan, perlu dilakukan pengecekan apakah kombinasi jembatan tersebut tidak melanggar restriksi dan memenuhi tujuan akhir permainan atau tidak. Hal-hal yang perlu dilakukan untuk memvalidasi solusi adalah sebagai berikut:

1. Restriksi banyak jembatan yang menghubungkan sebuah pasangan pulau yaitu minimal sebanyak 0 jembatan dan maksimal sebanyak 2 jembatan telah dipastikan tidak akan dilanggar ketika mencari kandidat jembatan secara horizontal dan vertikal.
2. Restriksi jembatan tidak boleh melewati pulau telah dipastikan tidak akan dilanggar ketika mencari kandidat jembatan secara horizontal dan vertikal.
3. Restriksi tidak boleh ada jembatan yang melewati jembatan lain dapat di validasi dengan cara mengiterasi seluruh jembatan vertikal yang dipakai dan akan dilakukan iterasi pada seluruh jembatan horizontal yang dipakai untuk dilakukan pengecekan apakah kedua jembatan tersebut bersilangan atau tidak.
4. Tujuan akhir permainan yaitu jumlah jembatan yang terhubung pada sebuah pulau harus sama dengan angka yang terdapat pada pulau tersebut dapat divalidasi dengan cara membuat penanda nilai untuk setiap pulau dan menambahnya untuk setiap jembatan yang dipakai.
5. Tujuan akhir permainan yaitu seluruh pulau dapat terkoneksi dengan menggunakan jembatan yang dipakai dapat divalidasi dengan merepresentasikan kondisi papan sebagai sebuah graf dengan pulau sebagai simpul dan jembatan sebagai sisi. Apabila seluruh komponen pada graf tidak berarah terhubung, maka seharusnya ketika dilakukan traversal menggunakan DFS seluruh simpul akan terkunjungi.

G. Validasi restriksi tidak ada jembatan yang melewati jembatan lain

Untuk memvalidasi restriksi tidak ada jembatan yang melewati jembatan lain, kita akan mengiterasi seluruh pasangan jembatan horizontal dan vertikal yang dipakai dan memvalidasi apakah kedua jembatan tersebut bersilangan atau tidak. Sebuah pasangan jembatan akan bersilangan apabila arah kedua

jembatan berbeda (satu horizontal dan satu vertikal), dan nilai posisi sumbu x jembatan vertikal berada diantara posisi sumbu x jembatan horizontal, dan nilai posisi sumbu y jembatan horizontal berada diantara posisi sumbu y jembatan vertikal. Secara lebih matematis sebuah pasangan jembatan akan bersilangan apabila memenuhi seluruh kondisi berikut.

$$H_{y_1} = H_{y_2}$$
$$V_{x_1} = V_{x_2}$$
$$H_{x_1} < V_{x_1} < H_{x_2}$$
$$V_{y_1} < H_{y_1} < V_{y_2}$$

Dengan H merupakan jembatan horizontal, dan V adalah jembatan vertikal, dan nilai pada subskrip H dan V merupakan nilai tupel jembatan.

H. Validasi tujuan akhir jumlah jembatan

Untuk melakukan validasi tujuan akhir permainan yaitu apakah jumlah jembatan yang terhubung pada sebuah pulau sama dengan angka yang berada di dalam pulau tersebut atau tidak, dapat dilakukan dengan cara pertama membuat penanda berisi nilai bilangan bulat untuk setiap pulau dengan nilai awal 0.

Kita akan mengiterasi seluruh jembatan yang dipakai baik jembatan vertikal maupun jembatan horizontal dan akan menambah nilai penanda untuk pulau yang berada pada titik awal dan titik akhir jembatan.

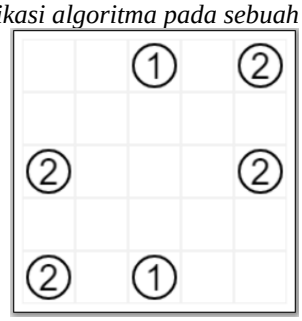
Setelah seluruh jembatan yang dipakai selesai diiterasi, kita akan mengiterasi seluruh pulau yang diberikan dan mengecek apakah nilai penanda sama dengan angka yang berada pada pulau.

I. Validasi tujuan akhir pulau terhubung

Untuk melakukan validasi tujuan akhir permainan yaitu apakah seluruh pulau sudah terhubung atau tidak, dapat dilakukan dengan cara merepresentasikan kondisi pulau dan jembatan sebagai graf. Apabila seluruh pulau sudah terhubung, maka seharusnya apabila dilakukan traversal dari sembarang pulau, maka seluruh pulau yang diberikan akan terkunjungi.

Kita dapat memvalidasi restriksi tersebut dengan cara menjalankan algoritma DFS pada sembarang pulau dan membuat penanda untuk setiap pulau yang sudah dikunjungi. Setelah algoritma DFS selesai, kita perlu mengiterasi Kembali seluruh pulau dan memastikan seluruh pulau memiliki penanda yang menandakan bahwa pulau itu sudah dikunjungi ketika menjalankan algoritma DFS.

J. Contoh aplikasi algoritma pada sebuah kasus



Gambar 5. Kondisi awal permainan sebagai contoh aplikasi algoritma

Berikut adalah contoh kondisi awal permainan yang akan digunakan. Tahap pertama yang akan dilakukan sebelum menjalankan algoritma adalah mengubah representasi pulau menjadi tupel seperti yang telah dijelaskan pada bagian A. Berikut tupel-tupel pulau setelah di representasikan.

Pulau

$< 1, 1, 2 >$
$< 1, 2, 2 >$
$< 2, 1, 1 >$
$< 2, 3, 1 >$
$< 3, 2, 2 >$
$< 3, 3, 2 >$

Setelah representasi pulau diubah menjadi tupel, tahap kedua adalah mencari seluruh kandidat jembatan horizontal dan vertikal. Langkah untuk melakukan pencarian adalah seperti yang sudah dijelaskan pada bagian C dan D.

Hasil pengurutan pada bagian C

$< 1, 1 >$
$< 2, 1 >$
$< 1, 2 >$
$< 3, 2 >$
$< 2, 3 >$
$< 3, 3 >$

Jembatan Horizontal

$< 1, 1, 2, 1 >$
$< 1, 1, 2, 1 >$
$< 1, 2, 3, 2 >$
$< 1, 2, 3, 2 >$
$< 2, 3, 3, 3 >$
$< 2, 3, 3, 3 >$

Hasil pengurutan pada bagian D

$< 1, 1 >$
$< 1, 2 >$
$< 2, 1 >$
$< 2, 3 >$
$< 3, 2 >$
$< 3, 3 >$

Jembatan Vertikal

$< 1, 1, 1, 2 >$
$< 1, 1, 1, 2 >$
$< 2, 1, 2, 3 >$
$< 2, 1, 2, 3 >$
$< 3, 2, 3, 3 >$
$< 3, 2, 3, 3 >$

Setelah seluruh kandidat jembatan horizontal dan vertikal sudah didapatkan, tahap selanjutnya adalah mencari seluruh kombinasi cara pemakaian jembatan.

Pencarian kombinasi cara pemakaian jembatan adalah dengan cara membuat dua buah larik yaitu larik yang berisi seluruh kemungkinan larik biner (elemennya bernilai 0 atau 1) dengan panjang N dan larik yang berisi seluruh kemungkinan larik biner (elemennya bernilai 0 atau 1) dengan panjang M,

dengan N adalah banyak kandidat jembatan horizontal dan M adalah banyak kandidat jembatan vertikal.

Kombinasi Pemakaian Jembatan Horizontal

$< 0, 0, 0, 0, 0, 0 >$
$< 0, 0, 0, 0, 0, 1 >$
$< 0, 0, 0, 0, 1, 0 >$
$< 0, 0, 0, 0, 1, 1 >$
$< 0, 0, 0, 1, 0, 0 >$
..
$< 1, 1, 1, 1, 0, 1 >$
$< 1, 1, 1, 1, 1, 0 >$
$< 1, 1, 1, 1, 1, 1 >$

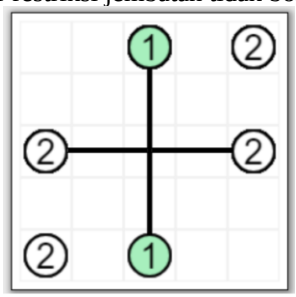
Kombinasi Pemakaian Jembatan Vertikal

$< 0, 0, 0, 0, 0, 0 >$
$< 0, 0, 0, 0, 0, 1 >$
$< 0, 0, 0, 0, 1, 0 >$
$< 0, 0, 0, 0, 1, 1 >$
$< 0, 0, 0, 1, 0, 0 >$
..
$< 1, 1, 1, 1, 0, 1 >$
$< 1, 1, 1, 1, 1, 0 >$
$< 1, 1, 1, 1, 1, 1 >$

Setelah seluruh kombinasi jembatan yang akan dipakai telah ditemukan, tahap selanjutnya adalah mencoba seluruh kemungkinan cara pemakaian jembatan dan memvalidasi apakah kumpulan jembatan yang dipakai merupakan solusi permainan.

Berikut adalah beberapa contoh solusi yang melanggar restriksi atau tujuan akhir permainan dan juga solusi yang memenuhi seluruh restriksi dan tujuan akhir permainan:

1. Melanggar restriksi jembatan tidak boleh bersilangan



Gambar 6. Contoh kombinasi pemakaian jembatan yang melanggar restriksi jembatan tidak boleh bersilangan

Pemakaian Jembatan Horizontal

$< 0, 0, 1, 0, 0, 0 >$

Pemakaian Jembatan Vertikal

$< 0, 0, 1, 0, 0, 0 >$

Pada kasus kombinasi jembatan tersebut terdapat jembatan <1,2,3,2> dan <2,1,2,3>. Seperti yang sudah dijelaskan pada bagian G, kita dapat mengecek apakah dua buah jembatan bersilangan atau tidak dengan menggunakan rumus berikut.

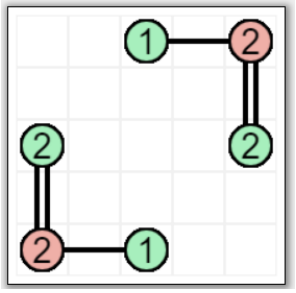
$$H_{y_1} = H_{y_2}$$
$$V_{x_1} = V_{x_2}$$
$$H_{x_1} < V_{x_1} < H_{x_2}$$
$$V_{y_1} < H_{y_1} < V_{y_2}$$

Apabila nilai yang dimiliki jembatan dimasukkan.

$$H_{y_1} = H_{y_2}\ (2 = 2, benar)$$
$$V_{x_1} = V_{x_2}\ (2 = 2, benar)$$
$$H_{x_1} < V_{x_1} < H_{x_2}\ (1 < 2 < 3, benar)$$
$$V_{y_1} < H_{y_1} < V_{y_2}\ (1 < 2 < 3, benar)$$

Karena semua restriksi bernilai benar, maka sudah dipastikan bahwa kedua jembatan bersilangan, sehingga jembatan-jembatan yang dipakai tidak dapat menjadi solusi.

2. Melanggar restriksi jumlah jembatan sebuah pulau



Gambar 7. Contoh kombinasi pemakaian jembatan yang melanggar restriksi jumlah jembatan sebuah pulau

Pemakaian Jembatan Horizontal

$< 0, 0, 0, 0, 0, 0 >$

Pemakaian Jembatan Vertikal

$< 1, 1, 1, 0, 1, 1 >$

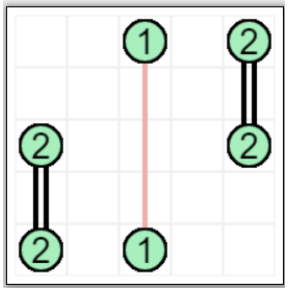
Untuk memvalidasi restriksi jumlah jembatan sebuah pulau, dapat dilakukan dengan cara membuat penanda bilangan bulat untuk setiap pulau dengan nilai awal 0, lalu kita akan mengiterasi seluruh jembatan yang dipakai, dan menambah nilai penanda pada pulau yang menjadi titik awal dan titik akhir jembatan. Berikut adalah kondisi penanda untuk setiap pulau setelah dilakukan perhitungan.

Jumlah Penanda Pulau

$< 1,1,2 >:\ 3$
$< 1,2,2 >:\ 2$
$< 2,1,1 >:\ 1$
$< 2,3,1 >:\ 1$
$< 3,2,2 >:\ 2$
$< 3,3,2 >:\ 3$

Karena pulau <1,1,2> memiliki 3 jembatan yang terhubung, dan seharusnya hanya 2 jembatan yang terhubung, maka jembatan-jembatan yang dipakai tidak dapat menjadi solusi.

3. Melanggar restriksi keterhubungan pulau



Gambar 8. Contoh kombinasi pemakaian jembatan yang melanggar restriksi keterhubungan pulau

Pemakaian Jembatan Horizontal

$< 1, 0, 0, 0, 1, 0 >$

Pemakaian Jembatan Vertikal

$< 1, 1, 0, 0, 1, 1 >$

Untuk memvalidasi restriksi keterhubungan pulau, dapat dilakukan dengan merepresentasi kondisi permainan menjadi sebuah graf lalu melakukan traversal dan memberikan penanda untuk setiap pulau yang sudah dikunjungi.

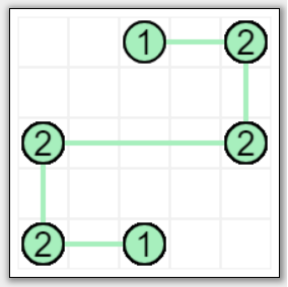
Berikut adalah tahapan traversal graf dengan menggunakan algoritma DFS di mulai dari pulau $< 2,1 >$.

Iterasi	Simpul ekspan	Simpul hidup	Dikunjungi					
			< 1,1 >	< 1,2 >	< 2,1 >	< 2,3 >	< 3,2 >	< 3,3 >
1	< 2,1 >	{< 2,3 >}	0	0	1	0	0	0
2	< 2,3 >	{}	0	0	1	1	0	0

Karena terdapat pulau yang pada akhir algoritma traversal memiliki nilai 0 pada larik dikunjungi, maka terdapat pulau yang tidak dapat kunjungi melalui pulau $< 2,1 >$ yang mengartikan jembatan-jembatan yang dipakai tidak membuat seluruh pulau terhubung.

Karena jembatan-jembatan yang dipakai tidak mampu menghubungkan seluruh pulau, maka jembatan-jembatan yang dipakai tidak dapat menjadi solusi.

4. Solusi yang memenuhi seluruh restriksi



Gambar 9. Contoh kombinasi pemakaian jembatan yang memenuhi seluruh restriksi

Pemakaian Jembatan Horizontal

<1, 0, 1, 0, 1, 0>

Pemakaian Jembatan Vertikal

<1, 0, 0, 0, 1, 0>

Untuk memvalidasi apakah jembatan-jembatan yang dipakai merupakan solusi yang valid kita akan memvalidasi restriksi-restriksi yang diberikan dan memvalidasi tujuan akhir permainan.

Berikut adalah tahapan untuk memvalidasi jembatan yang dipakai memenuhi seluruh restriksi dan tujuan akhir permainan.

1. Memeriksa seluruh pasangan jembatan yang dipakai, tidak ada pasangan jembatan yang bersilangan. Hal ini dapat dilakukan menggunakan ketentuan yang diberikan pada bagian G. Untuk jembatan yang dipakai pada solusi ini dipastikan tidak ada pasangan jembatan yang bersilangan, namun tidak diuraikan karena terlalu banyak perhitungan.
2. Memeriksa tujuan akhir bahwa jumlah jembatan yang terhubung pada sebuah pulau sama dengan angka yang terdapat pada pulau tersebut. Berikut adalah jumlah penanda untuk setiap pulau pada solusi ini.

Jumlah Penanda Pulau

$< 1,1,2 >: 2$
$< 1,2,2 >: 2$
$< 2,1,1 >: 1$
$< 2,3,1 >: 1$
$< 3,2,2 >: 2$
$< 3,3,2 >: 2$

Karena nilai penanda pada seluruh pulau memiliki nilai yang sama dengan angka yang berada dalam pulau, maka tujuan akhir jumlah jembatan yang terhubung kepada sebuah pulau terpenuhi.

3. Memeriksa tujuan akhir keterhubungan pulau. Berikut adalah hasil tabel setelah melakukan traversal menggunakan algoritma DFS dimulai dari pulau $< 2,1 >$.

Iterasi	Simpul ekspan	Simpul hidup	Dikunjungi					
			< 1,1 >	< 1,2 >	< 2,1 >	< 2,3 >	< 3,2 >	< 3,3 >
1	< 2,1 >	{< 1,1 >}	0	0	1	0	0	0
2	< 1,1 >	{< 1,2 >}	1	0	1	0	0	0
3	< 1,2 >	{< 3,2 >}	1	1	1	0	0	0
4	< 3,2 >	{< 3,3 >}	1	1	1	0	1	0
5	< 3,3 >	{< 2,3 >}	1	1	1	0	1	1
6	< 2,3 >	{}	1	1	1	1	1	1

Karena pada akhir traversal seluruh pulau telah dikunjungi, maka jembatan-jembatan yang dipakai sudah dipastikan menghubungkan seluruh pulau.

Karena setelah memeriksa seluruh restriksi dan tujuan akhir dan seluruh proses validasi mengatakan bahwa solusi valid, maka kombinasi jembatan yang dipakai merupakan solusi yang valid. Berikut representasi lebih formal untuk solusi yang dibuat.

Solusi

<1,1,2,1>
<1,2,3,2>
<2,3,3,3>
<1,1,1,2>
<3,2,3,3>

K. Kompleksitas Waktu Algoritma

Berikut adalah uraian kompleksitas waktu untuk setiap tahap dalam algoritma:

- Mengurutkan pulau pada untuk dipakai di bagian C dan D $O(n^2)$ atau $O(n \log(n))$ tergantung teknik pengurutan

Asumsi teknik yang digunakan adalah $O(n^2)$

- Mencari seluruh kandidat jembatan horizontal dan vertikal

$$O(n)$$

- Membangkitkan seluruh kombinasi pemakaian jembatan

$$O(2^H + 2^V)$$

- Iterasi seluruh kemungkinan kombinasi pemakaian jembatan

$$O(2^H * 2^V)$$
$$= O(2^{H+V})$$

- Validasi restriksi jembatan bersilangan

$$O(H * V)$$

- Validasi tujuan akhir jumlah jembatan

$$O(H + V + n)$$

- Validasi tujuan akhir seluruh pulau terhubung

$$O(H + V + n)$$

- Kompleksitas keseluruhan algoritma

$$O(n^2) + O(n) + O(2^H + 2^V) + O(2^{H+V}) * (O(H * V) + O(H + V + n) + O(H + V + n))$$
$$= O(n^2 + 2^{H+V} * (H * V + n))$$

Dengan n = jumlah pulau, H = jumlah kandidat jembatan horizontal, dan V = jumlah kandidat jembatan vertikal.

IV. KESIMPULAN

Penyelesaian untuk permainan hashi dapat dicari menggunakan kombinasi strategi bruteforce dan DFS. Pencarian solusi menggunakan algoritma bruteforce mempunyai kompleksitas waktu yang sangat besar yaitu $O(n^2 + 2^{H+V} * (H * V + n))$ untuk permainan hashi. Solusi akhir yang ditemukan pada permainan hashi dapat dimodelkan menjadi graf dan dapat dilakukan traversal menggunakan algoritma DFS. Karena kompleksitas waktu masih terlalu besar dan pemilihan solusi masih terlalu naif, maka masih terdapat banyak ruang untuk mengoptimisasi penyelesaian dengan algoritma bruteforce.

V. UCAPAN TERIMA KASIH

Pertama-tama, puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa karena berkat rahmat-Nya penulis dapat menyelesaikan makalah ini dengan baik. Penulis juga berterima kasih kepada orang tua, serta teman-teman yang selalu memberikan semangat dan dukungan kepada penulis sehingga makalah ini dapat terselesaikan. Penulis juga tak lupa berterima kasih kepada Bapak Dr. Ir. Rinaldi Munir, M.T. selaku dosen mata kuliah strategi algoritma yang telah memberikan banyak ilmu dan motivasi dalam kegiatan perkuliahan. Terakhir, penulis memohon maaf apabila dalam penulisan makalah ini terdapat kesalahan baik disengaja maupun tidak disengaja. Penulis berharap makalah ini dapat bermanfaat bagi banyak orang.

VI. REFERENSI

[1] https://www.nikoli.co.jp/en/puzzles/hashiwokakero/, diakses tanggal 20 mei 2023

[2] www.kakuro-online.com/hashi, diakses tanggal 20 mei 2023

[3] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2021-2022/Algoritma-Brute-Force-(2022)-Bag1.pdf, diakses tanggal 21 mei 2023

[4] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag1.pdf, diakses tanggal 21 mei 2023

[5] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/BFS-DFS-2021-Bag2.pdf, diakses tanggal 21 mei 2023

[6] https://www.geeksforgeeks.org/backtracking-to-find-all-subsets/, diakses tanggal 22 mei 2023

VII. 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 2023

Dewana Gustavus Haraka Otang 13521173