# BAB III
# ANALISIS DAN PERANCANGAN

## 3.1 Analisis

Subbab analisis akan mencakup analisis kebutuhan sistem, analisis proses penyelesaian Hashi, analisis fungsional dan analisis non-fungsional.

### 3.1.1 Analisis Masalah

Analisis masalah merupakan proses identifikasi serta evaluasi mengenai masalah-masalah yang timbul dari pengembangan sistem, baik mengenai aturan serta teknik-teknik yang digunakan dalam pengembangan sistem.

Adapun kebutuhan yang harus dipenuhi serta aturan-aturan yang harus dipenuhi dalam menjalankan *game puzzle* tersebut adalah sebagai berikut:

a. Aturan-aturan yang terdapat pada *game puzzle hashi*:
   1) Jembatan harus vertikal atau horizontal (tidak ada yang diagonal atau berlekuk).
   2) Jembatan tidak boleh saling berpotongan.
   3) Hanya boleh terdapat maksimum 2 buah jembatan untuk satu rute (antara 2 pulau yang sama).
   4) Solusi *Hashi* adalah semua pulau terhubung satu sama lain, tidak ada pulau yang terisolir dan semua pulau mempunyai jembatan sejumlah angka yang dituliskan padanya.
   5) Pengecekan arah jembatan dimulai dari atas, bawah, kiri, dan kanan.
   6) Banyak arah suatu pulau adalah pulau tetangga yang sisa bobotnya > 0 dan tidak terpotong oleh jembatan lainnya.
b. Teknik-teknik yang dilakukan untuk menyelesaikan *game puzzle Hahsi*:
   1) Teknik *Just Enough Neighbours*

      Dalam teknik ini setiap pulau dapat dikatakan memiliki bobot asli jika pulau tersebut belum memiliki hubungan dengan pulau manapun.

      a) Mulai pencarian dari pulau yang berada pada posisi a1.

b) Untuk pulau yang mempunyai bobot asli 1 dan hanya punya 1 arah, dengan minimal bobot tetangga sebesar 1, maka bangun 1 jembatan ke arah tersebut.
c) Untuk pulau yang mempunyai bobot asli 2 dan hanya punya 1 arah, dengan minimal bobot tetangga sebesar 2, maka bangun 2 jembatan ke arah tersebut.
d) Untuk pulau yang mempunyai bobot asli 4 dan hanya punya 2 arah, dengan minimal bobot tetangga sebesar 2, maka bangun masing-masing 2 jembatan ke masing-masing arah.
e) Untuk pulau yang mempunyai bobot asli 6 dan hanya punya 3 arah, dengan minimal bobot tetangga sebesar 2, maka bangun masing-masing 2 jembatan ke masing-masing arah.
f) Untuk pulau yang mempunyai bobot asli 8 dan hanya punya 4 arah, dengan minimal bobot tetangga sebesar 2, maka bangun masing-masing 2 jembatan ke masing-masing arah.
g) Setelah ditemukan pulau yang sesuai dengan aturan pada teknik *Just Enough Neighbour* maka pencari akan kembali dimulai dari point b) untuk pulau berikutnya.
h) Pencarian akan berhenti jika tidak ditemukan lagi pulau-pulau yang sesuai dengan aturan yang dimiliki oleh *Just Enough Neighbour*.

2) Teknik *One Unsolved Neightbour*
a) Mulai pencarian dari pulau yang berada pada posisi a1 dengan menjadikan pulau-pulau yang belum terselesaikan oleh teknik sebelumnya sebagai inputan.
b) Untuk pulau yang punya sisa bobot 1 dan hanya punya 1 arah, maka bangun 1 jembatan ke arah tersebut. Dikatakan sisa bobot 1 , artinya pulau bisa jadi berbobot lebih dari 1, tetapi pada saat itu sisa jembatan yang bisa dibangun hanya 1, karena jembatan lain sudah terhubung dengan pulau lain.

c) Untuk pulau yang punya sisa bobot 2 dan hanya punya 1 arah, maka bangun 2 jembatan ke arah tersebut. Dikatakan sisa bobot 2, artinya pulau juga bisa jadi punya bobot asli lebih besar dari 2, tetapi pada saat itu sisa bobot hanya tinggal 2.
d) Setelah ditemukan pulau yang sesuai dengan aturan pada teknik *One Unsolved Neightbour* maka pencari akan kembali dimulai dari point b) untuk pulau berikutnya.
e) Pencarian akan berhenti jika tidak ditemukan lagi pulau-pulau yang sesuai dengan aturan yang dimiliki oleh *One Unsolved Neightbour.*

3) Teknik *Few Neighbour*

Dalam teknik ini setiap pulau dapat dikatakan memiliki bobot asli jika pulau tersebut belum memiliki hubungan dengan pulau manapun.

a) Mulai pencarian dari pulau yang berada pada posisi a1 dengan menjadikan pulau-pulau yang belum terselesaikan oleh teknik sebelumnya sebagai inputan.
b) Setiap pulau yang punya bobot asli sebesar 3, dan punya 2 arah, boleh membangun 1 jembatan ke masing-masing arah.
c) Setiap pulau yang punya bobot asli sebesar 5, dan hanya punya 3 arah, boleh membangun 1 jembatan ke masing-masing arah.
d) Setiap pulau yang punya bobot asli sebesar 7, dan hanya punya 4 arah, boleh membangun 1 jembatan ke masing-masing arah.
e) Setelah ditemukan pulau yang sesuai dengan aturan pada teknik *Few Neighbours* maka pencari akan kembali dimulai dari point b) untuk pulau berikutnya.
f) Pencarian akan berhenti jika tidak ditemukan lagi pulau-pulau yang sesuai dengan aturan yang dimiliki oleh *Few Neighbours.*

4) Teknik *Left Overs*

Dalam teknik ini setiap pulau dapat dikatakan memiliki bobot asli jika pulau tersebut belum memiliki hubungan dengan pulau manapun.

a) Mulai pencarian dari pulau yang berada pada posisi a1 dengan menjadikan pulau-pulau yang belum terselesaikan oleh teknik sebelumnya sebagai inputan.
b) Untuk setiap pulau X yang punya bobot asli sebesar n dan hanya punya arah n dengan (n-1) tetangga yang mempunyai sisa bobot atau bobot asli sebesar 1, maka bangun 1 jembatan dari pulau X ke tetangganya yang tidak mempunyai sisa bobot atau bobot asli 1.
c) Setelah ditemukan pulau yang sesuai dengan aturan pada teknik *Left Overs* maka pencari akan kembali dimulai dari point b untuk pulau berikutnya.
d) Pencarian akan berhenti jika tidak ditemukan lagi pulau-pulau yang sesuai dengan aturan yang dimiliki oleh *Left Overs.*

5) Teknik *Isolation*

Dalam teknik ini setiap pulau dapat dikatakan memiliki bobot asli jika pulau tersebut belum memiliki hubungan dengan pulau manapun.

a) Mulai pencarian dari pulau yang berada pada posisi a1 dengan menjadikan pulau-pulau yang belum terselesaikan oleh teknik sebelumnya sebagai inputan
b) Untuk pulau yang memiliki bobot asli 1 dan mempunyai tetangga dengan bobot asli 1, maka kedua pulau tersebut tidak boleh saling berhubungan, maka bangun 1 jembatan ke pulau lainnya yang tidak memiliki bobot asli 1.
c) Untuk pulau yang memiliki bobot asli 2 dan mempunyai tetangga dengan bobot asli 2, maka kedua pulau tersebut tidak boleh saling berhubungan, maka bangun 2 jembatan ke pulau lainnya yang tidak memiliki bobot asli 2.
d) Setelah ditemukan pulau yang sesuai dengan aturan pada teknik *Isolation* maka pencari akan kembali dimulai dari point b untuk pulau berikutnya.

e) Pencarian akan berhenti jika tidak ditemukan lagi pulau-pulau yang sesuai dengan aturan yang dimiliki oleh *Isolation.*

Jika semua teknik sudah dijalakan tetapi masih ada pulau yang belum terselesaikan maka pencarian akan mulai kembali dari point a) pada teknik *Just enough Neighbour* hingga tidak ada lagi perubahan bobot pulau dengan menggunakan teknik *Solving Hashi*.

6) Teknik *Breath First Search (BFS)*

Metode pencarian BFS akan mengembangkan ruang dan keadaan secara melebar. Dari keadaan awal *Puzzle Hashi,* algoritma BFS akan mengembangkan semua kemungkinan arah jembatan yang dapat dibangun pada suatu pulau.

a) Mulai pencarian dari pulau yang berada pada posisi a1 dengan menjadikan pulau-pulau yang belum terselesaikan oleh teknik *solving Hashi* sebagai inputan.
b) Kembangkan setiap kemungkinan yang ada pada setiap node hingga solusi *hashi* ditemukan, jika sudah tidak ada lagi node yang dapat dikembangkan sedangkan soal masih belum selesai berarti soal memang tidak memiliki jawaban.

7) Teknik *Deepth First Search (DFS)*

Dari keadaan awal *Puzzle Hashi,* algoritma DFS akan mengembangkan hanya 1 kemungkinan arah jembatan yang dapat dibangun pada suatu pulau menjadi keadaan baru. Selanjutnya, algoritma DFS akan memajukan pencarian ke pulau berikutnya dan mengembangkan 1 kemungkinan arah jembatan pada pulau tersebut tanpa melanggar aturan permainan *Hashi.*

a) Mulai pencarian dari pulau yang berada pada posisi a1 dengan menjadikan pulau-pulau yang belum terselesaikan oleh teknik *solving Hashi* sebagai inputan.

b) Kembangkan setiap kemungkinan yang ada pada setiap node hingga solusi *hashi* ditemukan, jika sudah tidak ada lagi node yang dapat dikembangkan sedangkan soal masih belum selesai berarti soal memang tidak memiliki jawaban.

### 3.1.2 Analisis Proses Penyelesaian Hashi

Pembahasan proses penyelesaian *game Puzzle Hashi* akan mencakup beberapa hal berikut:

1. Teknik *Solving Hashi* dalam mencari solusi *Hashi*, dan
2. Teknik Pencarian *Depth First Search* (DFS) dalam mencari solusi *Hashi*.
3. Teknik Pencarian *Breath First Search* (BFS) dalam mencari solusi *Hashi*.

#### 3.1.2.1 Teknik *Solving Hashi* dalam Mencari Solusi *Hashi*

Sebagaimana yang dijelaskan pada bab 2 bahwa teknik *Solving Hashi* yang digunakan dalam mencari solusi *Hashi*, mencakup beberapa teknik berikut:

1. Teknik *Just Enough Neighbours*
2. Teknik *One Unsolved Neighbours*
3. Teknik *Few Neighbours*
4. Teknik *Leftovers*
5. Teknik *Isolation*

Semua teknik ini akan diimplementasikan secara berurutan untuk mencari jawaban dari soal *Hashi*.

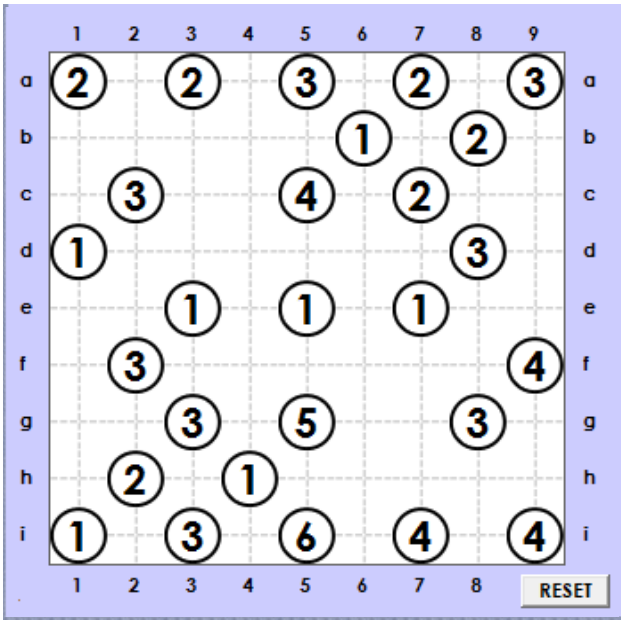
##### 3.1.2.1.1 Teknik *Just Enough Neighbours*

Perlu diingat bahwa, aturan menggambar jembatan antar pulau dalam *puzzle Hashi* harus mengikuti aturan di bawah ini:

1. Jembatan harus vertikal atau horizontal (tidak ada yang diagonal atau berlekuk)
2. Jembatan tidak boleh saling bertimpaan
3. Hanya boleh terdapat maksimum 2 buah jembatan untuk satu rute (antara 2 pulau yang sama)

4. Solusi *Hashi* adalah semua pulau terhubung satu sama lain, tidak ada pulau yang terisolir dan semua pulau mempunyai jembatan sejumlah angka yang dituliskan padanya.

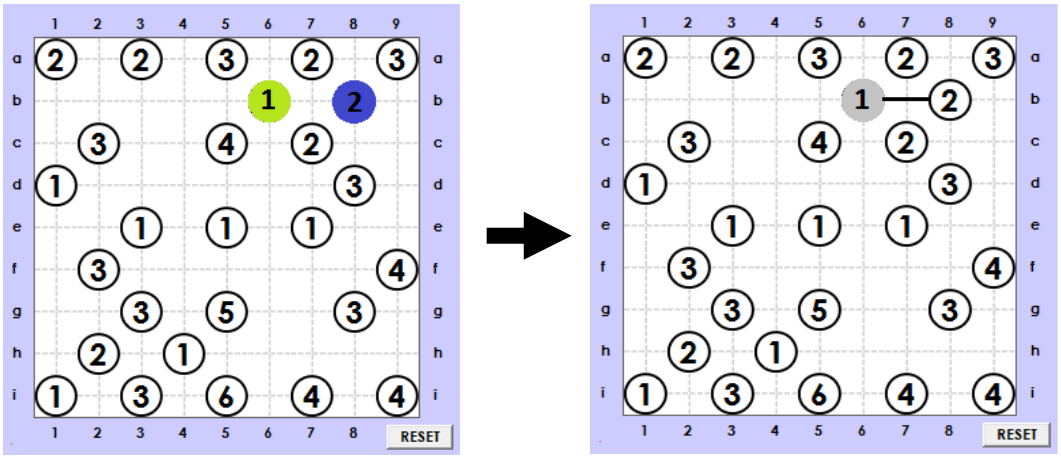
Misalkan, contoh soal *puzzle Hashi* yang akan diselesaikan terlihat pada gambar 3.1 berikut.




**Gambar 3.1 Contoh Soal *Hashi* yang akan Diselesaikan**

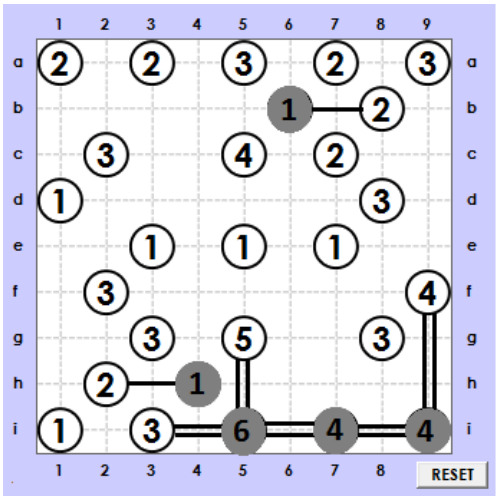
Untuk contoh kasus pada gambar 3.1, perhatikan pulau pada b6. Ukuran pulau adalah 1, sehingga pulau tersebut membutuhkan 1 jembatan yang terhubung padanya. Akan tetapi pulau ini hanya mempunyai 1 tetangga, yaitu berada pada posisi sebelah kanan (pulau b8). Oleh karena pulau b6 hanya memiliki bobot 1 dan hanya memiliki 1 tetangga maka satu-satunya cara untuk menyelesaikan pulau b6 adalah membangun 1 jembatan ke pulau b8.

Ini adalah teknik "*Just Enough Neighbours*". Setelah diselesaikan, pulau b6 diberi warna abu-abu untuk memberi tanda bahwa pulau tersebut sudah diselesaikan dan tidak perlu diperhitungkan lagi.




**Gambar 3.2 Pulau b6 diselesaikan dengan Teknik "*Just Enough Neighbours*"**

Setelah pulau b6 terselesaikan, maka ulangi proses pengecekan yang sama terhadap semua pulau yang belum terselesaikan. Bila ada pulau yang mempunyai kasus yang sama, maka selesaikan dengan teknik ini. Hasil penggambaran jembatan untuk teknik ini seperti terlihat pada gambar 3.3 berikut.

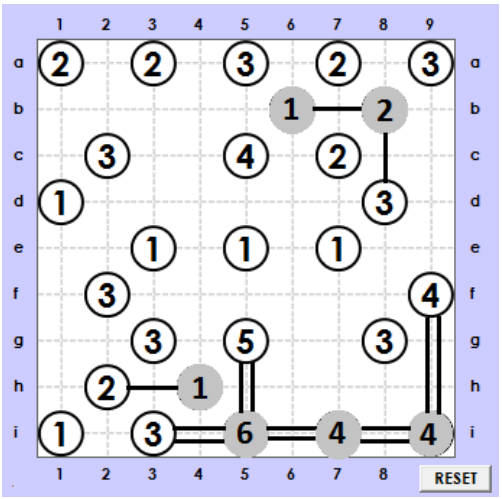



**Gambar 3.3 Hasil Penelusuran dengan Teknik "*Just Enough Neighbours*"**

### 3.1.2.1.2 Teknik *One Unsolved Neighbour*

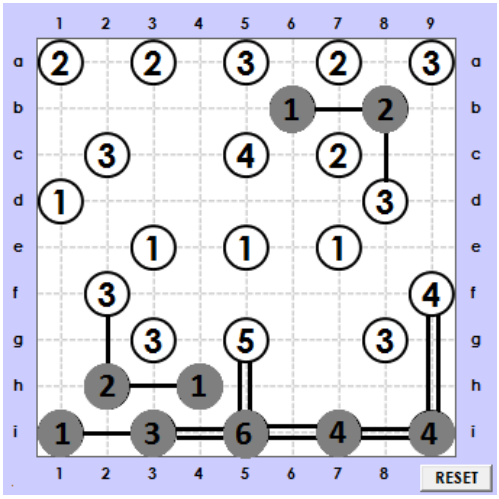
Teknik ini adalah teknik yang hampir sama dengan teknik "*Just Enough Neighbours*". Perhatikan pulau b8 yang mempunyai ukuran sebesar satu dan hanya mempunyai satu tetangga di sebelah bawah yaitu pulau d8. Pulau b8 tidak

mempunyai tetangga yang lain. Oleh karena itu, hubungkan pulau b8 ke d8 seperti terlihat pada gambar 3.4 berikut.




**Gambar 3.4 Pulau h4 diselesaikan Teknik "*One Unsolved Neighbour*"**

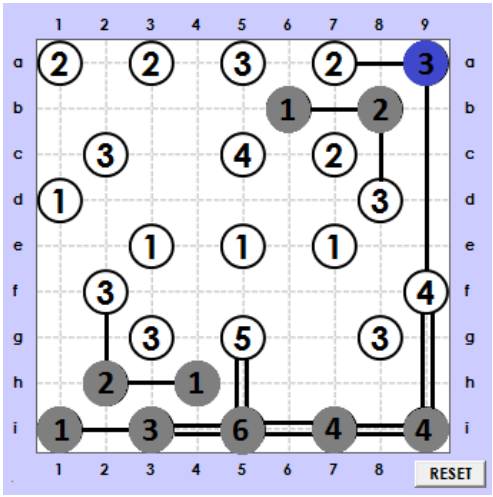
Setelah pulau b8 terselesaikan, maka ulangi proses pengecekan yang sama terhadap semua pulau yang belum terselesaikan. Bila ada pulau yang mempunyai kasus yang sama, maka selesaikan dengan teknik ini. Hasil penggambaran jembatan untuk teknik ini seperti terlihat pada gambar 3.4 berikut




**Gambar 3.5 Hasil Penelusuran dengan Teknik "*One Unsolved Neighbour*"**

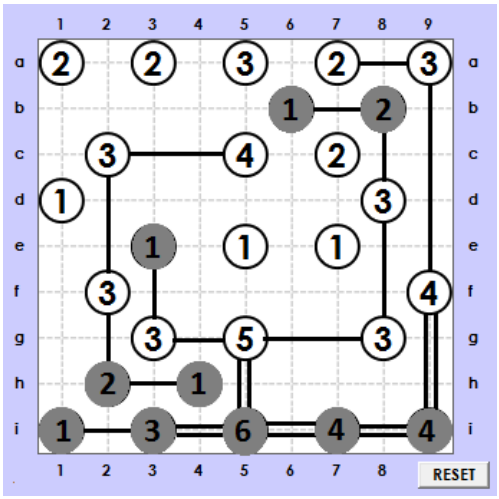
### 3.1.2.1.3 Teknik *Few Neighbours*

Teknik "*Few Neighbours*" bekerja pada dasar aturan bahwa tidak boleh terdapat lebih dari 2 jembatan untuk 1 rute yang sama. Perhatikan pulau a9 yang mempunyai ukuran pulau sebesar 3 dan hanya mempunyai 2 tetangga. Oleh karena hanya boleh maksimum terdapat 2 buah jembatan untuk setiap rute, maka untuk kasus ini harus ada minimal 1 jembatan yang mengarah ke arah yang lain. Pulau a7 hanya berukuran 2. Oleh karena itu, 1 jembatan pulau a9 mengarah ke pulau a7 dan 1 jembatan lainnya mengarah ke pulau f9.




**Gambar 3.6 Pulau a9 diselesaikan dengan Teknik "*Few Neighbours*"**

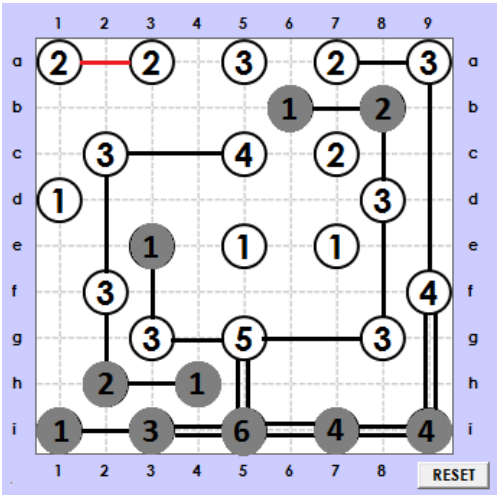
Dengan menggunakan kombinasi teknik "*Just Enough Neighbours*", "*One Unsolved Neighbour*" dan "*Few Neighbours*", soal *Hashi* dapat diselesaikan hingga mencapai keadaan seperti terlihat pada gambar 3.7.




**Gambar 3.7 Penyelesaian Akhir dengan Menggunakan Kombinasi 3 Teknik *Neighbours***

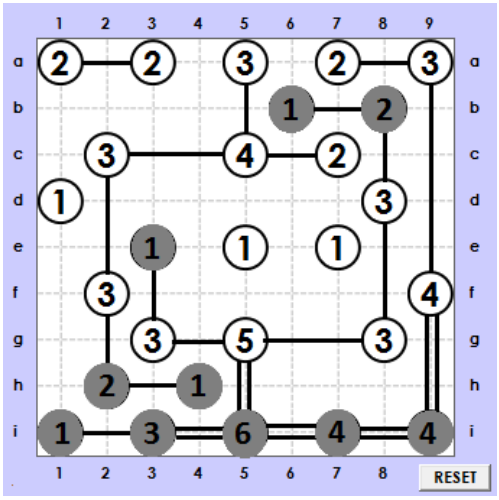
### 3.1.2.1.4 Teknik *Leftovers*

Perhatikan pulau a1 pada gambar 3.7. Pulau tersebut membutuhkan 2 tambahan jembatan dan mempunyai 2 tetangga. Salah satunya adalah pulau d1 yang mempunyai ukuran pulau sebesar 1 dan oleh karenanya hanya dapat mempunyai 1 jembatan. Dengan demikian, 1 jembatan dari pulau a1 harus mengarah ke tetangganya yang lain yaitu pulau a3. Penyelesaiannya dapat dilihat pada gambar 3.8 berikut.




**Gambar 3.8 Pulau a1 dihubungkan dengan Teknik "*LeftOvers*"**

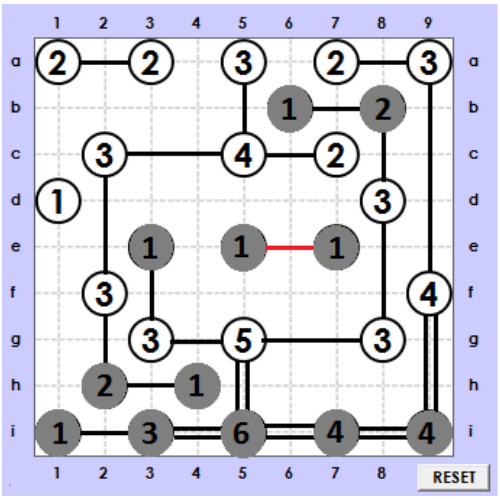
Dengan menggunakan kombinasi teknik "*Just Enough Neighbours*", "*One Unsolved Neighbour*", "*Few Neighbours*" dan "*LeftOvers".* soal *Hashi* dapat diselesaikan hingga mencapai keadaan seperti terlihat pada gambar 3.9.




**Gambar 3.9 Penyelesaian Akhir dengan Menggunakan Kombinasi 4 Teknik *Neighbours***

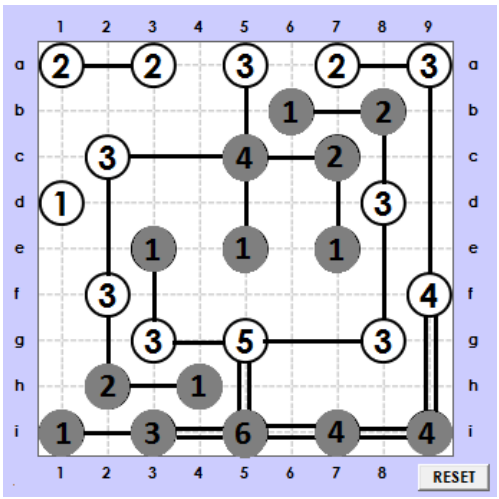
#### 3.1.2.1.5 Teknik *Isolation*

Teknik ini menggunakan dasar aturan bahwa tidak boleh ada pulau yang terisolir (tidak terhubung ke pulau lainnya). Perhatikan gambar *puzzle Hashi* pada gambar 3.7. *Puzzle* hampir selesai, tetapi kita tidak tahu pulau c5 harus mengarah ke mana. Kuncinya adalah pulau e5 yang memiliki ukuran pulau sebesar 1. Pulau ini hanya boleh memiliki 1 jembatan. Bagaimana bila jembatan dari pulau e5 mengarah ke kanan? Amati gambar 3.10 berikut.



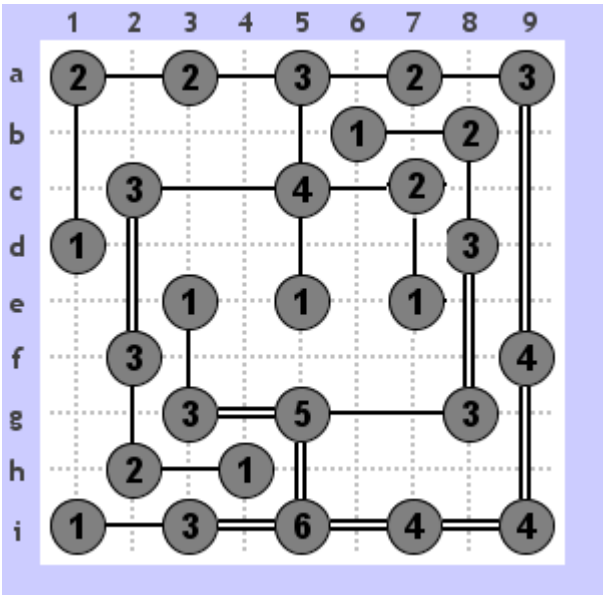

**Gambar 3.10 Pulau e5 dihubungkan dengan pulau e7**

Jika pulau e5 dihubungkan dengan pulau e7, maka kedua pulau ini akan memotong sirkuit yang pendek, saling melengkapi dan terisolir dari pulau lainnya. Dengan demikian, pulau e5 tidak boleh mengarah ke kanan dan pasti mengarah ke atas (pulau c5).




**Gambar 3.11 Pulau c5 dihubungkan dengan pulau e5**

Dari gambar 3.11, masih tersisa beberapa pulau yang belum terselesaikan. Ulangi proses pencarian mulai dari teknik *Just Enough Neighbour* sampai dengan teknik *Isolation* hingga ditemukan solusi seperti pada gambar 3.12 berikut.




**Gambar 3.12 Solusi *Hashi* Terpecahkan**

### 3.1.2.2 Teknik Pencarian BFS atau DFS dalam Mencari Solusi *Hashi*

Apabila setelah menggunakan teknik *solving Hashi* dan tenyata *Hashi* belum terselesaikan atau pun jika *user* ingin menggunakan teknik pencarian DFS dan BFS tanpa mencoba menyelesaikan soal *Hashi* tersebut, maka hal itu sangat dimungkinkan untuk dilakukan.

#### 3.1.2.2.1 Teknik Pencarian BFS

Metode pencarian BFS akan mengembangkan ruang dan keadaan secara melebar. Dari keadaan awal *puzzle Hashi*, algoritma BFS akan mengembangkan semua kemungkinan arah jembatan yang dapat dibangun pada suatu pulau. Oleh karena, penelusuran dimulai dari pulau paling kiri atas, maka penelusuran kemungkinan arah jembatan bisa dibatasi hanya pada arah kanan dan bawah, untuk setiap pulau. Dari pulau paling kiri atas, semua kemungkinan jembatan dikembangkan. Kemudian berlanjut ke pulau-pulau atau *node-node* berikutnya. Algoritma ini hanya akan berjalan jika masih ditemukan pulau-pulau yang belum terselesaikan oleh teknik *solving Hashi*.

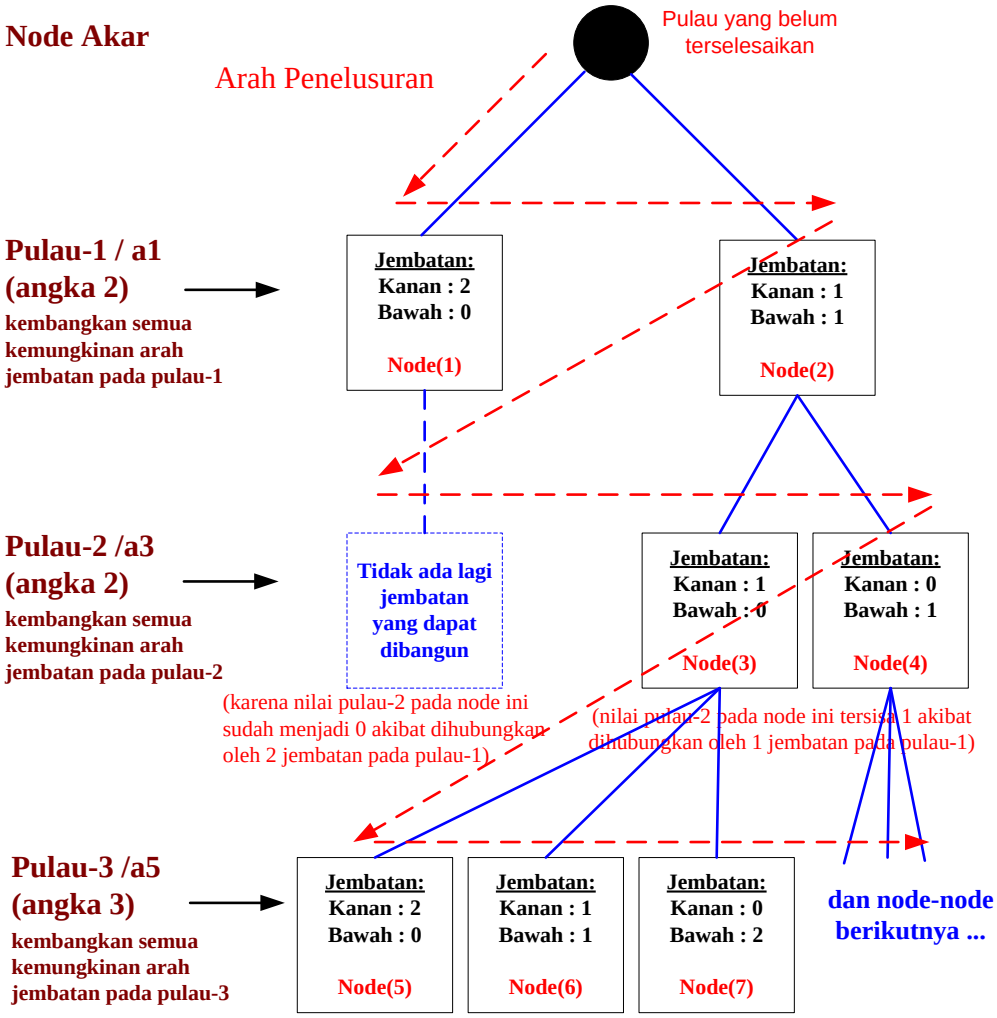
Pengembangan arah jembatan tidak boleh menyalahi aturan Hashi. Dengan demikian, semua kemungkinan arah jembatan di dalam *node-node* BFS adalah

jembatan yang *valid* dan tidak menyalahi aturan. Proses pengembangan ini berlanjut hingga ditemukan solusi *puzzle*.

Secara ringkas, algoritma proses BFS dapat dijabarkan sebagai berikut:

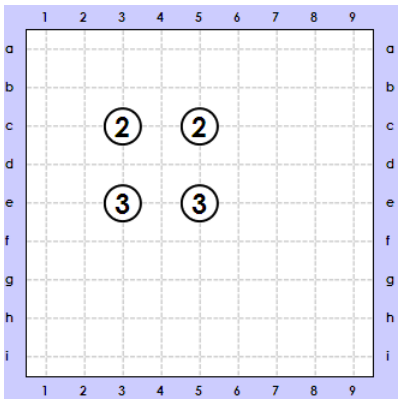
1. Penelusuran dimulai pada pulau-1.
2. Kembangkan semua kemungkinan arah jembatan ke kanan dan ke bawah dari pulau-1. Tempatkan semua kemungkinan arah jembatan *valid* ini ke dalam *node-node* baru pada level-1, misalkan Node(1) dan Node(2).
3. Penelusuran dilanjutkan pada Node(1). Kembangkan lagi semua kemungkinan arah jembatan ke kanan dan ke bawah dari pulau-2. Tempatkan semua kemungkinan arah jembatan *valid* ini ke dalam *node-node* baru pada level-2, misalkan Node(3) dan Node(4).
4. Penelusuran yang sama dilanjutkan pada Node(2) dan seterusnya, hingga didapatkan solusi.

Sebagai contoh, untuk soal *Hashi* pada gambar 3.1, maka penelusuran solusi dengan pohon BFS dapat dilihat pada gambar 3.13 berikut.



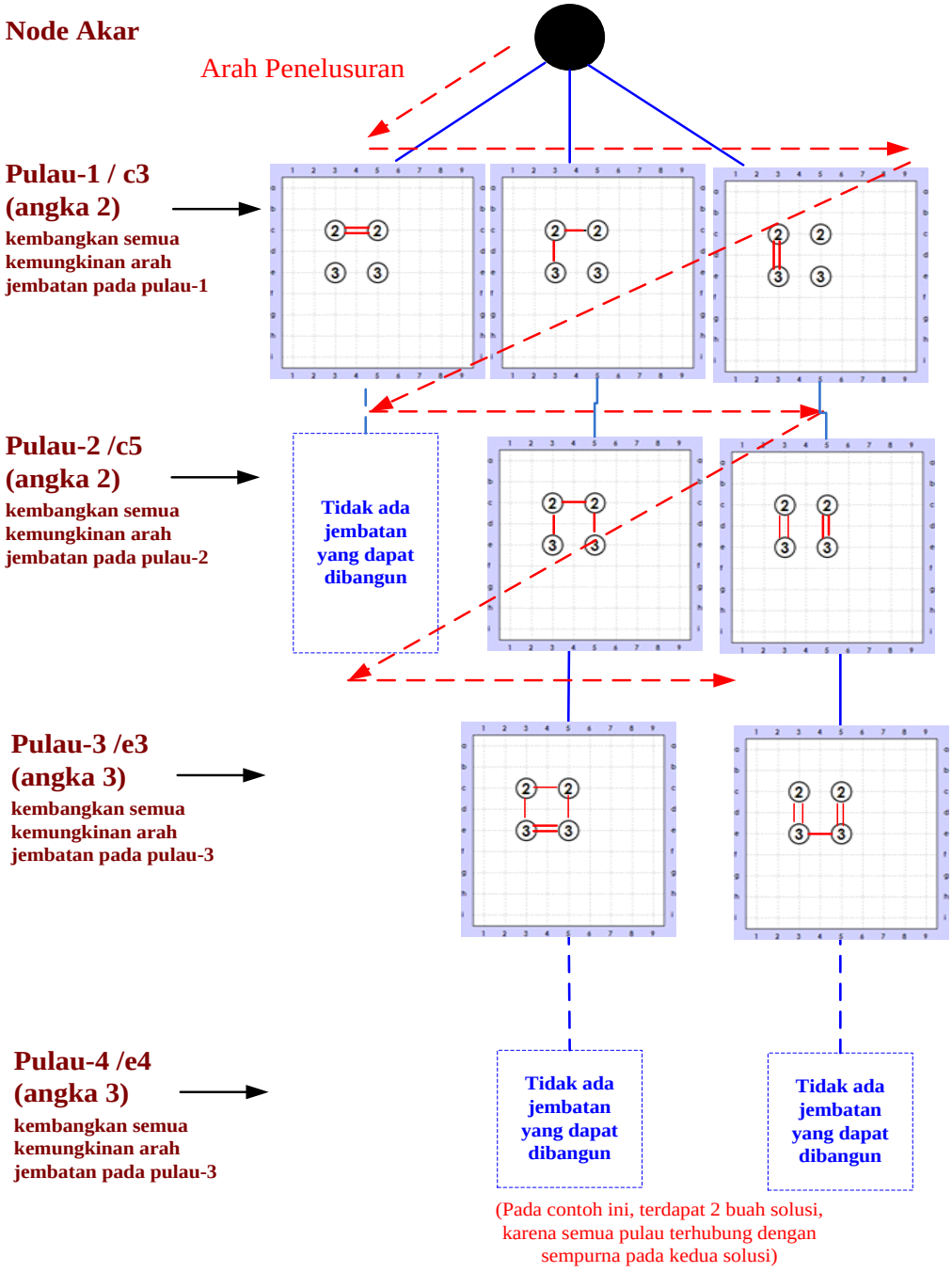

**Gambar 3.13 Penerapan BFS dalam Mencari Solusi *Hashi***

Sebagai contoh sederhana, misalkan soal *Hashi* yang mau diselesaikan, seperti terlihat pada gambar berikut.



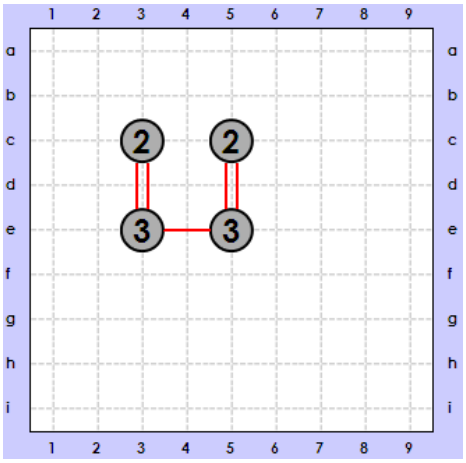

**Gambar 3.14 Contoh Soal Hashi**

Penelusuran dengan metode BFS pada soal di atas adalah sebagai berikut:



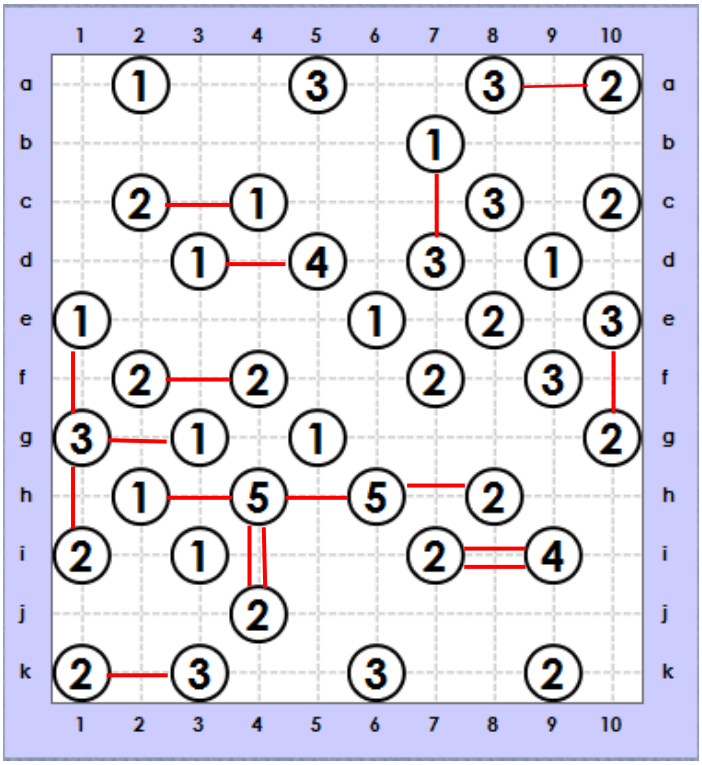

**Gambar 3.15 Penelusuran Metode BFS**

Solusi Hashi yang ditemukan dapat dilihat pada gambar berikut.




**Gambar 3.16 Solusi Hashi**

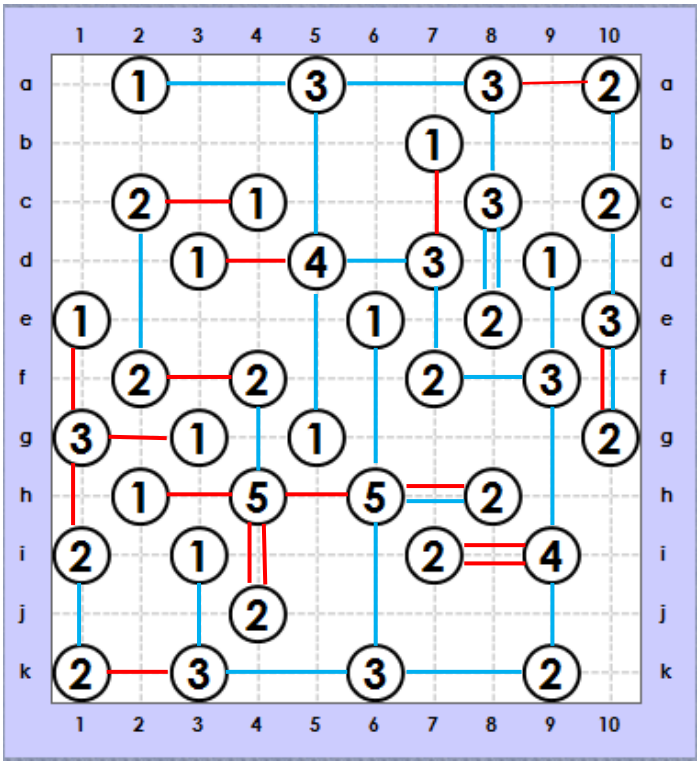
Berdasarkan metode pencarian diatas maka kita dapat lihat algoritma BFS hanya akan berjalan jika teknik *solving Hashi* gagal dalam menyelesaikan soal-soal yang diberikan sehingga algoritma BFS digunakan untuk membantu menyelesaikan soal-soal tersebut, untuk lebih jelasnya bisa kita lihat pada contoh soal dibawah ini yang diselesaikan dengan teknik *solving Hashi* ditambah dengan algoritma BFS




**Gambar 3.17 Solusi Hashi dengan teknik solving hashi**

Berdasarkan hasil pencarian yang telah dilakukan di atas dengan menggunakan teknik *Solving Hashi* maka dapat kita lihat bahwa teknik tersebut tidak dapat menyelesaikan soal *Hashi* yang diberikan. Berdasarkan hasil solusi tersebut pula maka kita dapat tentukan state awal dari soal *Hashi* berdasarkan pulau-pulau yang belum terselesaikan tersebut yang nantinya akan dijadikan sebagai kondisi awal pada algoritma BFS, berikut state awal untuk soal pada gambar 3.17:

1. Pulau yang menjadi inputan (A2, A5, A8, A10, C2, C8, C10, D5, D7, D9, E6, E8, E10, F2, F4, F7, F9, G5, G10, H4, H6, H8, I1, I3, I9, K1, K3, K6, K9).
2. Pencarian selalu dimulai pada pulau yang berada pada posisi kiri atas (A1).
3. Pencarian dibatasi hanya ke arah kanan dan bawah.
4. Pulau yang sudah terselesaikan dengan *Solving Hashi* tidak akan berubah.




**Gambar 3.18 Solusi Hashi dengan BFS**

Bedasarkan inputan diatas tadi maka soal dapat diselesaikan dengan menggunakan algoritma BFS, dan pulau-pulau yang sudah terselesaikan dengan menggunakan teknik *Solving Hashi* tidak mengalami perubahan.

### 3.1.2.2.2 Teknik Pencarian DFS dalam Mencari Solusi *Hashi*

Dari keadaan awal *puzzle Hashi*, algoritma DFS akan mengembangkan hanya 1 kemungkinan arah jembatan yang dapat dibangun pada suatu pulau menjadi keadaan baru. Selanjutnya, algoritma DFS akan memajukan pencarian ke pulau berikutnya dan mengembangkan 1 kemungkinan arah jembatan pada pulau tersebut tanpa melanggar aturan permainan *Hashi*. Algoritma ini hanya akan berjalan jika masih ditemukan pulau-pulau yang belum terselesaikan oleh teknik *solving Hashi*.

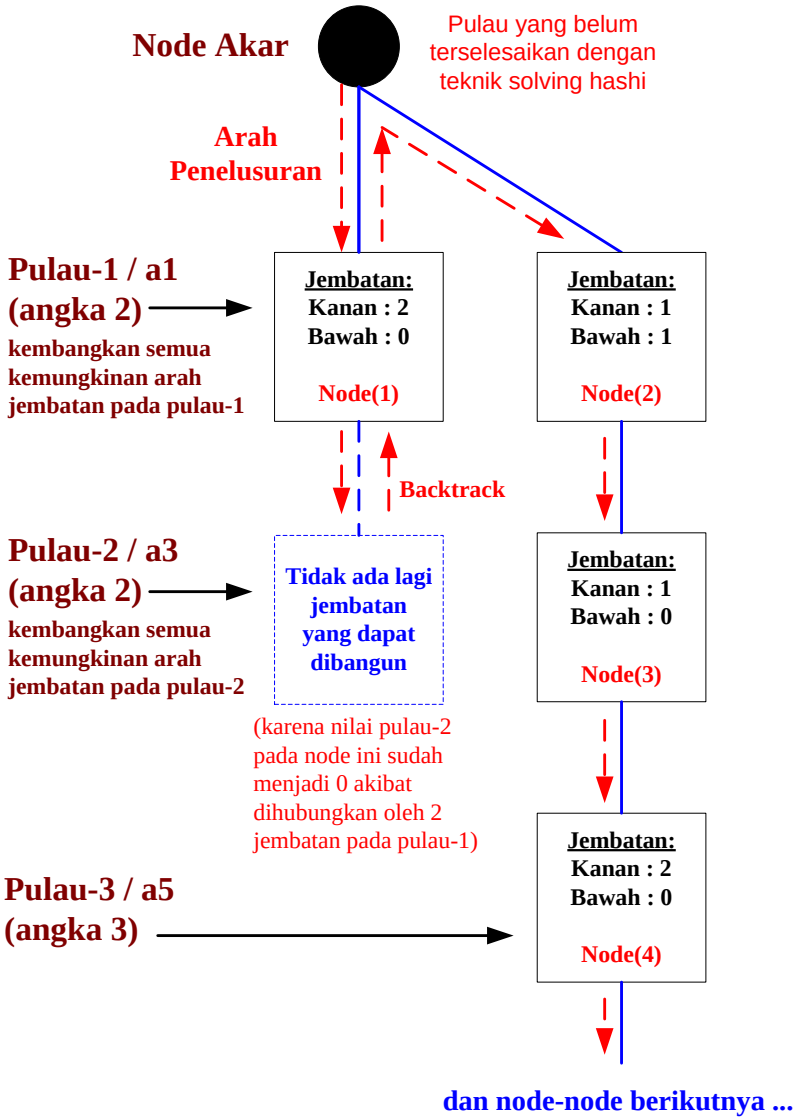
Bila tidak ada jembatan yang dapat dikembangkan pada suatu jalur karena semuanya melanggar aturan *Hashi*, maka metode DFS akan kembali ke jalur

sebelumnya dan mengambil arah jembatan yang lain. Keadaan ini berlanjut hingga jembatan pada pulau terakhir dikembangkan.

Secara ringkas, algoritma proses DFS dapat dijabarkan sebagai berikut:

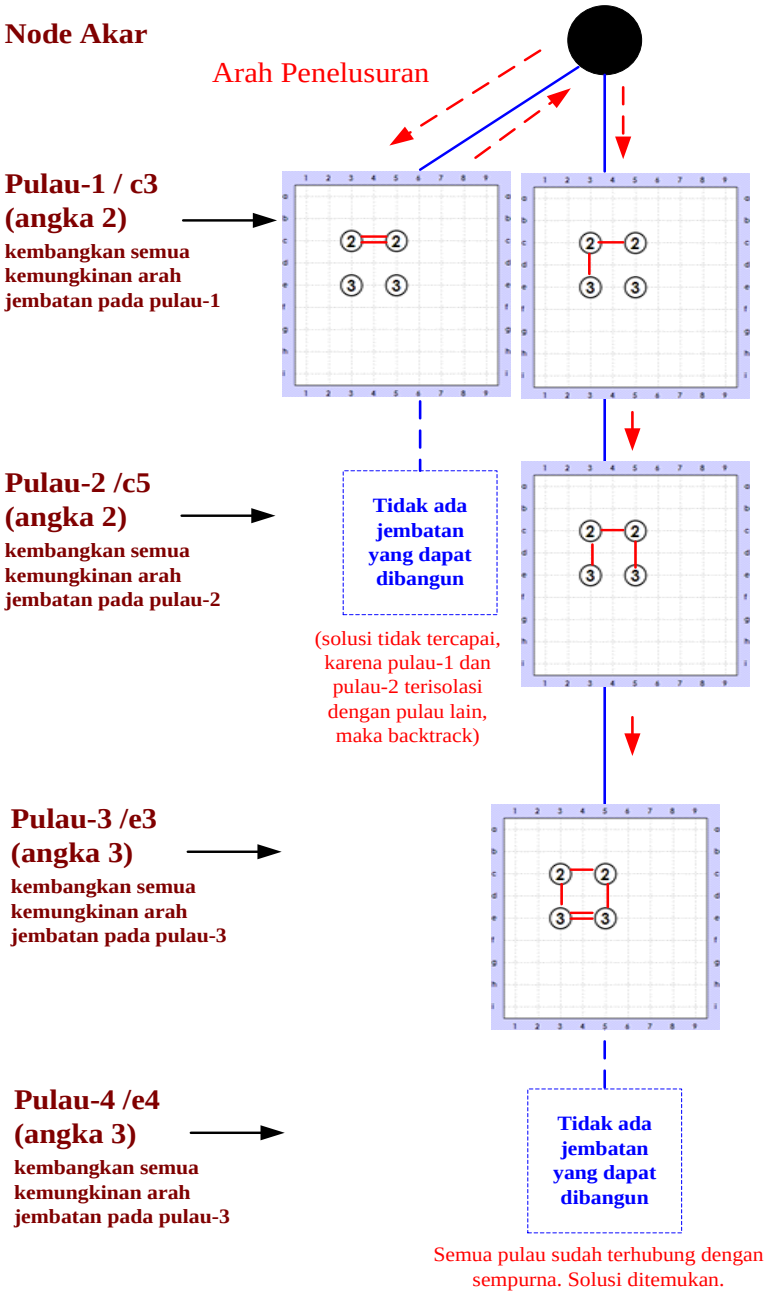
1. Penelusuran dimulai pada pulau-1.
2. Kembangkan satu kemungkinan arah jembatan ke kanan atau ke bawah dari pulau-1. Tempatkan kemungkinan arah jembatan *valid* ini ke dalam satu *node* baru pada level-1, misalkan Node(1).
3. Penelusuran dilanjutkan pada Node(1). Kembangkan lagi satu kemungkinan arah jembatan ke kanan atau ke bawah dari pulau-2. Tempatkan kemungkinan arah jembatan *valid* ini ke dalam *node* baru pada level-2, misalkan Node(2).
4. Teruskan penelusuran yang sama pada *node* berikutnya, secara mendalam pada level-level berikutnya.
5. Apabila terdapat suatu *node* yang tidak dapat membangun jembatan karena menyalahi aturan Hashi, maka penelusuran kembali (*backtrack*) ke *node* induk dan bangun lagi jembatan lain yang berbeda dengan *node* anak yang sudah ada. Teruskan penelusuran ke *node* anak yang baru.
6. Penelusuran yang sama dilanjutkan pada *node-node* berikutnya, hingga didapatkan solusi.

Sebagai contoh, misalkan contoh soal *Hashi* yang akan diselesaikan seperti terlihat pada gambar 3.1, maka penelusuran solusi dengan pohon DFS dapat dilihat pada gambar 3.19 berikut.



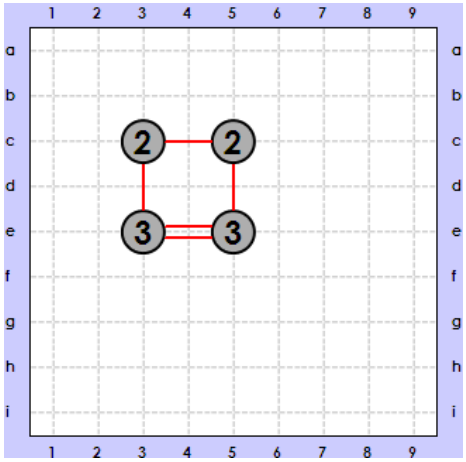

**Gambar 3.19 Penerapan DFS dalam Mencari Solusi *Hashi***

Teknik DFS ini akan mencoba semua kemungkinan hingga ditemukan solusi *Hashi*. Sebagai contoh sederhana, misalkan soal *Hashi* yang mau diselesaikan, seperti terlihat pada gambar 3.20, maka penelusuran dengan metode DFS adalah sebagai berikut:



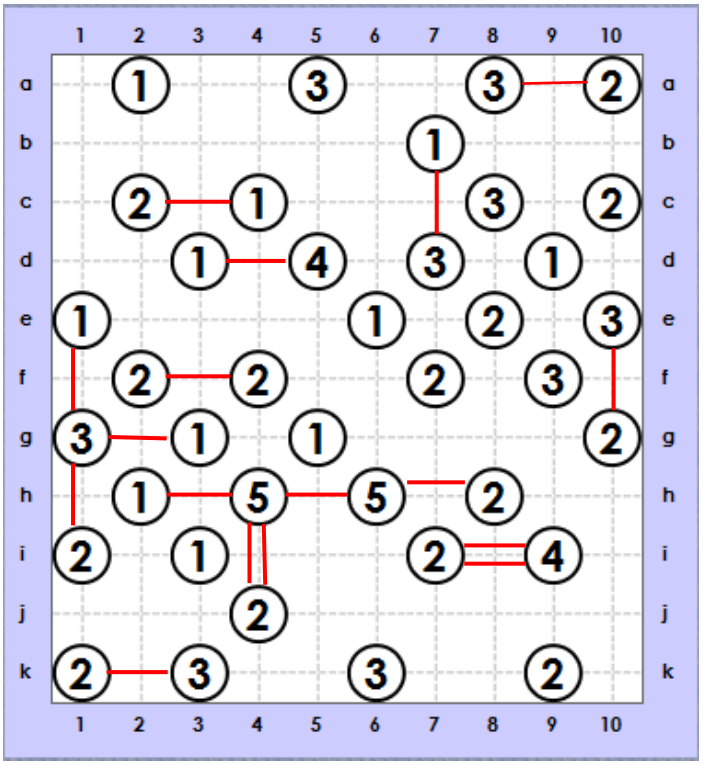

**Gambar 3.20 Penelusuran Metode DFS**

Solusi Hashi yang ditemukan dapat dilihat pada gambar berikut.




**Gambar 3.21 Solusi Hashi dari Penelusuran DFS**

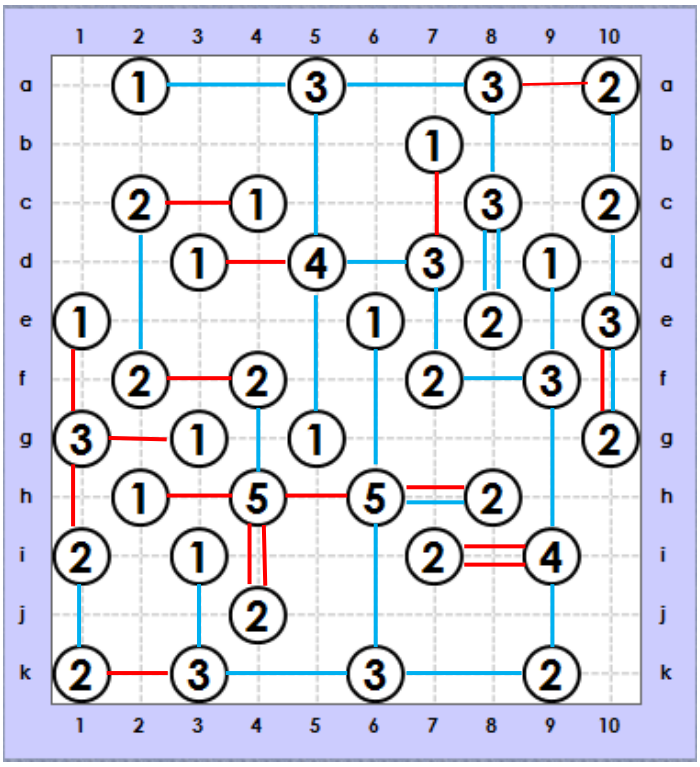
Berdasarkan metode pencarian diatas maka kita dapat lihat algoritma DFS hanya akan berjalan jika teknik *solving Hashi* gagal dalam menyelesaikan soal-soal yang diberikan sehingga algoritma DFS digunakan untuk membantu menyelesaikan soal-soal tersebut, untuk lebih jelasnya bisa kita lihat pada contoh soal dibawah ini yang diselesaikan dengan teknik *solving Hashi* ditambah dengan algoritma DFS




**Gambar 3.22 Solusi Hashi dengan teknik solving hashi**

Berdasarkan hasil pencarian yang telah dilakukan di atas dengan menggunakan teknik *Solving Hashi* maka dapat kita lihat bahwa teknik tersebut tidak dapat menyelesaikan soal *Hashi* yang diberikan. Berdasarkan hasil solusi tersebut pula maka kita dapat tentukan state awal dari soal *Hashi* berdasarkan pulau-pulau yang belum terselesaikan tersebut yang nantinya akan dijadikan sebagai kondisi awal pada algoritma DFS, berikut state awal untuk soal pada gambar 3.22:

1. Pulau yang menjadi inputan (A2, A5, A8, A10, C2, C8, C10, D5, D7, D9, E6, E8, E10, F2, F4, F7, F9, G5, G10, H4, H6, H8, I1, I3, I9, K1, K3, K6, K9).
2. Pencarian selalu dimulai pada pulau yang berada pada posisi kiri atas (A1).
3. Pencarian dibatasi hanya ke arah kanan dan bawah.
4. Pulau yang sudah terselesaikan dengan *Solving Hashi* tidak akan berubah.




**Gambar 3.23 Solusi Hashi dengan DFS**

Bedasarkan inputan diatas tadi maka soal dapat diselesaikan dengan menggunakan algoritma BFS, dan pulau-pulau yang sudah terselesaikan dengan menggunakan teknik *Solving Hashi* tidak mengalami perubahan.

### 3.1.3 Analisis Non-Fungsional

Analisis non-fungsional merupakan proses indentifikasi dan evaluasi mengenai kebutuhan sistem maupun pengguna, sehingga diharapkan game yang dibangun nanti dapat berjalan dengan baik.

Adapun kebutuhan *hardware* dan *software* yang harus dipenuhi untuk menjalankan program game *puzzle Hashiwokakere* tersebut adalah:

Spesifikasi *hardware* yang direkomendasikan untuk menjalankan aplikasi penyelesaian game *puzzle Hashiwokakero* (*Hashi*) adalah sebagai berikut :

1) Spesifikasi yang dibutuhkan untuk menjalankan game pada sisi *User:*

a) Prosesor *Pentium* IV 2.6 GHz dan Motherboard Intel.

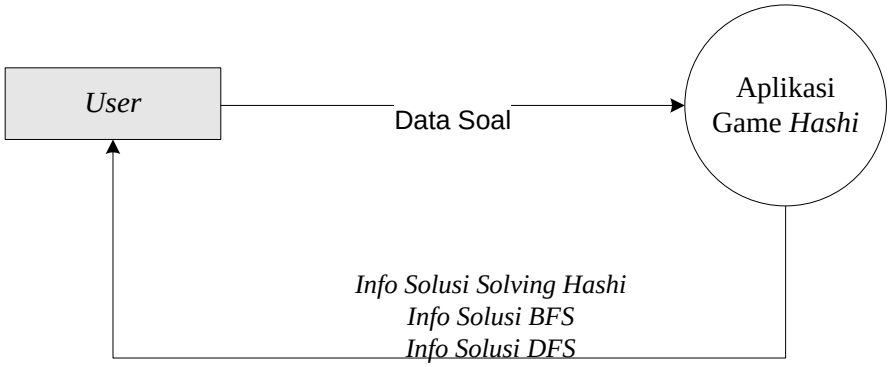
b) Memori (RAM) sebesar 512 Mb.
c) *Harddisk* 2 *GigaByte*.
d) Monitor SVGA yang mendukung resolusi minimum 1024 x768.
e) *Keyboard* dan *Mouse*.

c. Sedangkan spesifikasi *software* yang dibutuhkan untuk menjalankan perangkat lunak ini adalah sebagai berikut :
   1. Sistem operasi Microsoft Windows 98 *Second Edition* / Windows NT / Windows 2000 / Windows XP.
   2. Microsoft Visual Basic 6.

### 3.1.4 Analisis Fungsional

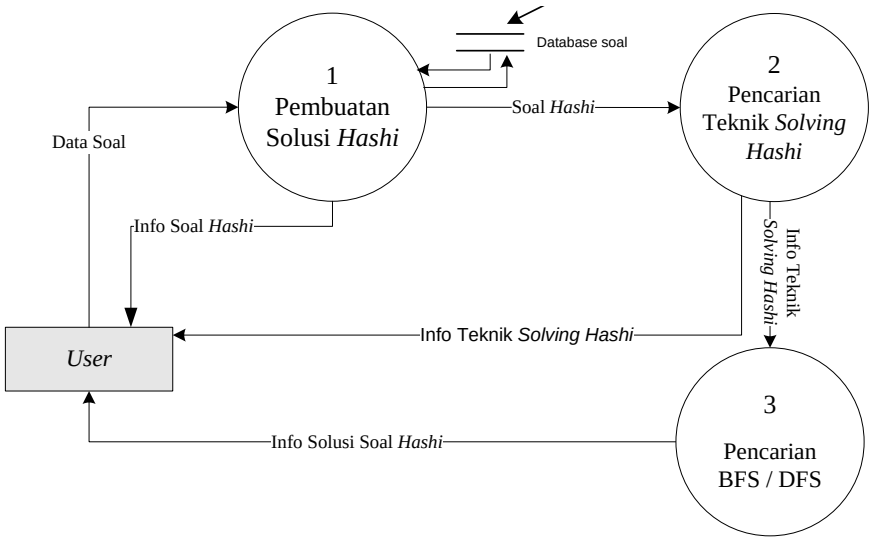
Analisis fungsional akan membahas perancangan sistem di dalam aplikasi. Perancangan Sistem adalah penggambaran, perancangan dan pembuatan sketsa. Pada tahap ini, *tool* yang digunakan untuk menganalisis dan merancang sistem adalah *Data Flow Diagram* (DFD). DFD digunakan untuk menggambarkan hubungan sistem dengan lingkungan. Proses yang terjadi di dalam suatu sistem dan bagaimana masukan ditransformasikan menjadi keluaran akan dirancang dalam bentuk DFD sebagai berikut.

Diagram konteks dari proses pencarian soal *Hashi* ini dapat dilihat pada gambar 3.24. Pada gambar tersebut, yang menjadi entitas adalah *user* atau pengguna aplikasi.




**Gambar 3.24 Diagram Konteks Proses Game *Hashi***

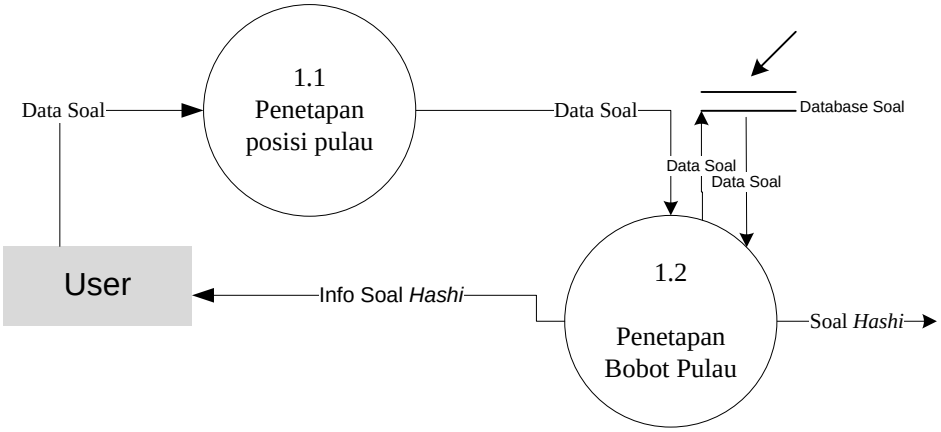
Diagram level 0 dari proses pencarian solusi *Hashi* ini dapat dirincikan pada gambar 3.25 berikut.




**Gambar 3.25 Diagram Level 0 Proses Game *Hashi***

Dari *diagram level* 0 pada gambar 3.25, dapat dirincikan lebih lanjut proses-proses yang terdapat di dalamnya sebagai berikut:

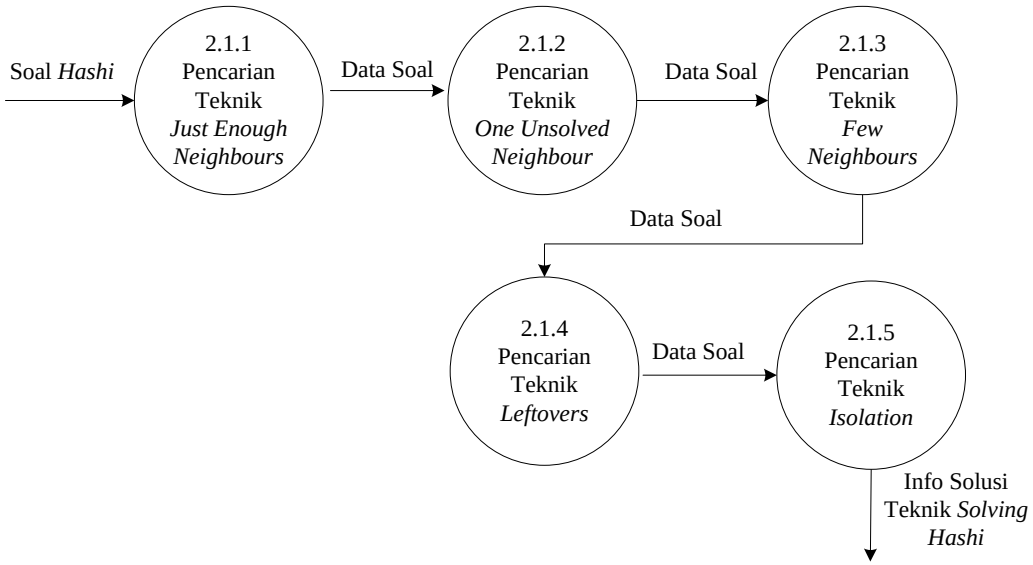
1. Proses 1.0 : Proses Pembuatan Soal Hashi.
   *Data Flow Diagram* (DFD) dari proses 1.0 dapat dilihat pada gambar 3.26.




**Gambar 3.26 DFD Level 1 - Proses Pembuatan Soal *Hashi***

2. Proses 2.0 : Proses Pencarian Teknik *Solving Hashi*

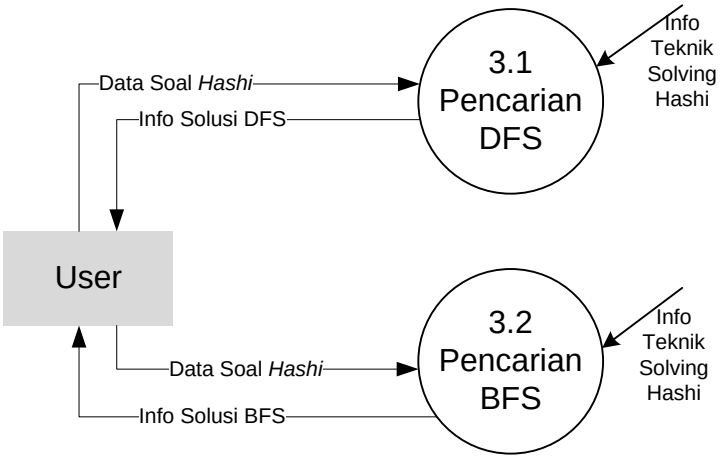
   *Data Flow Diagram* (DFD) dari proses 2.0 dapat dilihat pada gambar 3.27.



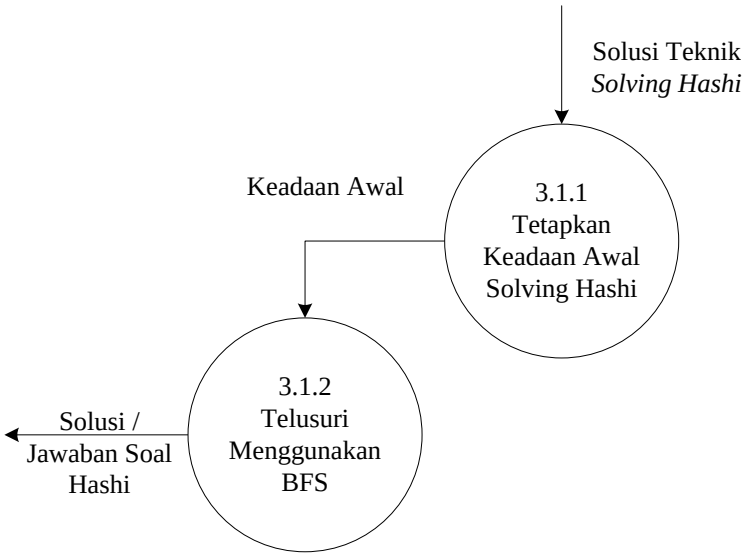

**Gambar 3.27 DFD Level 2 - Proses Pencarian Teknik *Solving Hashi***

3. Proses 2.0 : Proses Pencarian BFS atau DFS

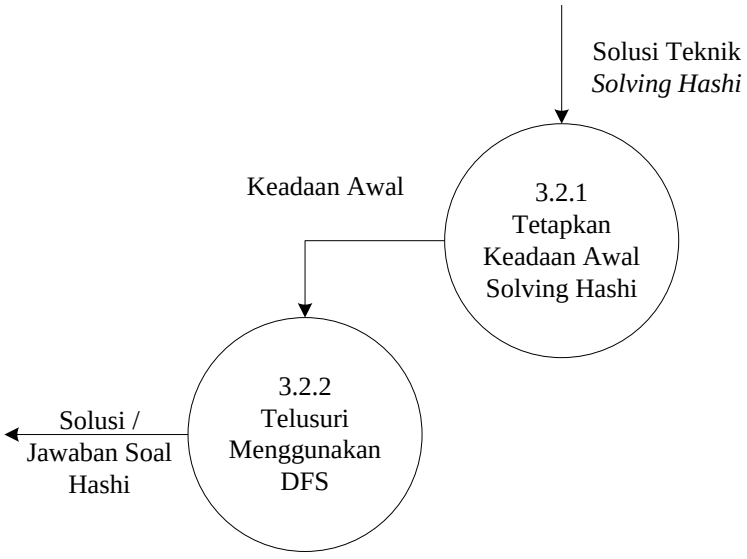
   *Data Flow Diagram* (DFD) dari proses 3.0 dapat dilihat pada gambar 3.28.




**Gambar 3.28 DFD Level 3 - Proses Pencarian BFS/DFS**




**Gambar 3.29 DFD Level 3 - Proses Pencarian BFS**




**Gambar 3.30 DFD Level 3 – Proses Pencarian DFS**

Untuk lebih jelasnya mengenai alur data pada masing-masing proses dapat dilihat pada keterangan berikut di bawah ini.

**Tabel 3.1 Spesifikasi Proses**

**Spesifikasi proses :**

| | |
|---|---|
| Nomor | 1 |
| Nama | Proses Pembuatan Soal Hashi. |
| Masukan | Data Soal (berupa posisi pulau dan bobot pulau) |
| Keluaran | Soal Hashi. |
| Uraian | Proses ini berfungsi untuk membuat soal Hashi, dengan menentukan posisi dan besar bobot pulau. |
| Logika Proses | 1. Pengguna memasukan nilai baris dan nilai kolom.<br>2. Jumlah baris minimal 9 dan jumlah maksimal yang bisa di *input* 20.<br>3. Jumlah kolom minimal 9 dan jumlah maksimal yang bisa di *input* 20.<br>4. Tentukan posisi pulau pada papan permainan *Hashi*.<br>5. Tentukan bobot pulau. |
| | |
| Nomor | 2 |
| Nama | Proses Pencarian Teknik *Solving Hashi*. |
| Masukan | Soal Hashi. |
| Keluaran | Solusi Teknik *Solving Hashi*. |
| Uraian | Proses ini berfungsi untuk mencari dan membangun jembatan yang sudah pasti merupakan solusi, dengan menggunakan teknik *Solving Hashi*. |

| | |
|---|---|
| | Proses ini akan memperingan kerja teknik BFS dan teknik DFS. |
| Logika Proses | 1. Pengguna memilih teknik apa yang akan digunakan.<br>2. Jika *user* melakukan pencarian tanpa memasukan soal maka akan muncul *message* kalau soal *hashi* belum ada.<br>3. Jika *user* melakukan pencarian dan soal tidak menemukan jawaban yang benar maka akan muncul *message* kalau soal *hashi* tidak memiliki jawaban.<br>4. Jika *user* melakukan pencarian dan soal memiliki jawaban yang benar maka akan muncul *message* kalau soal *hashi* memiliki jawaban |
| | |
| Nomor | 3 |
| Nama | Proses Pencarian BFS / DFS. |
| Masukan | Solusi Teknik *Solving Hashi*. |
| Keluaran | Solusi / Jawaban Soal *Hashi*. |
| Uraian | Proses ini berfungsi untuk mencari dan membangun sisa jembatan yang belum terselesaikan oleh teknik *Solving Hashi*, dengan menggunakan teknik BFS (pencarian melebar pertama) dan teknik DFS (pencarian mendalam pertama). |
| Logika Proses | 1. Pengguna dapat memilih teknik pencarian BFS atau DFS jika |

| | |
|---|---|
| | dengan menggunakan teknik *Solving Hashi* jawaban masih belum ditemukan.<br>2. Jika *user* menggunakan teknik BFS maka pencarian akan dilakukan dengan mengkombinasikan *solving hashi* dengan teknik BFS itu sendiri.<br>3. Jika *user* menggunakan teknik DFS maka pencarian akan dilakukan dengan mengkombinasikan *solving hashi* dengan teknik DFS itu sendiri. |
| | |
| Nomor | 2.1.1 |
| Nama | Pencarian Teknik *Just Enough Neighbours* |
| Masukan | Soal *Hashi* |
| Keluaran | Solusi Teknik *Just Enough Neighbours* |
| Uraian | Proses ini berfungsi untuk mencari dan membangun jembatan yang sudah pasti merupakan solusi, dengan menggunakan teknik *Just Enough Neighbours* |
| Logika proses | 1. Proses ini berjalan otomatis ketika *user* memilih untuk melakukan pencarian dengan cara manual<br>2. Jika dengan menggunakan teknik *Just Enough Neighbours* dan solusi belum ditemukan maka proses |

| | |
|---|---|
| | pencarian akan dilanjutkan menggunakan teknik yang lain<br>3. Jika semua teknik telah digunakan dan solusi belum ditemukan makan proses akan dilanjutkan dengan proses pencarian dengan menggunakan metode BFS/DFS |
| | |
| Nomor | 2.1.2 |
| Nama | Pencarian teknik *One Unsolved Neighbours* |
| Masukan | Soal *Hashi* |
| Keluaran | Solusi teknik *One Unsolved Neighbours* |
| Uraian | Proses ini berfungsi untuk mencari dan membangun jembatan yang sudah pasti merupakan solusi, dengan menggunakan teknik *One Unsolved Neighbours* |
| Logika proses | 1. Proses ini berjalan otomatis ketika *user* memilih untuk melakukan pencarian dengan cara manual<br>2. Jika dengan menggunakan teknik *One Unsolved Neighbours* dan solusi belum ditemukan maka proses pencarian akan dilanjutkan menggunakan teknik yang lain<br>3. Jika semua teknik telah digunakan dan solusi belum ditemukan makan proses akan dilanjutkan dengan |

| | |
|---|---|
| | proses pencarian dengan menggunakan metode BFS/DFS |
| | |
| Nomor | 2.1.3 |
| Nama | Pencarian teknik *Few Neighbours* |
| Masukan | Soal *Hashi* |
| Keluaran | Solusi teknik *Few Neighbours* |
| Uraian | Proses ini berfungsi untuk mencari dan membangun jembatan yang sudah pasti merupakan solusi, dengan menggunakan teknik *Few Neighbours* |
| Logika proses | 1. Proses ini berjalan otomatis ketika *user* memilih untuk melakukan pencarian dengan cara manual<br>2. Jika dengan menggunakan teknik *Few Neighbours* dan solusi belum ditemukan maka proses pencarian akan dilanjutkan menggunakan teknik yang lain<br>3. Jika semua teknik telah digunakan dan solusi belum ditemukan makan proses akan dilanjutkan dengan proses pencarian dengan menggunakan metode BFS/DFS |
| | |
| Nomor | 2.1.4 |
| Nama | Pencarian teknik *Leftovers* |
| Masukan | Soal *Hashi* |
| Keluaran | Solusi teknik *Leftovers* |
| Uraian | Proses ini berfungsi untuk mencari dan |

| | |
|---|---|
| | membangun jembatan yang sudah pasti merupakan solusi, dengan menggunakan teknik *Leftovers* |
| Logika proses | 1. Proses ini berjalan otomatis ketika *user* memilih untuk melakukan pencarian dengan cara manual<br>2. Jika dengan menggunakan teknik *Leftovers* dan solusi belum ditemukan maka proses pencarian akan dilanjutkan menggunakan teknik yang lain<br>3. Jika semua teknik telah digunakan dan solusi belum ditemukan makan proses akan dilanjutkan dengan proses pencarian dengan menggunakan metode BFS/DFS |
| | |
| Nomor | 2.1.5 |
| Nama | Pencarian teknik |
| Masukan | Soal *Hashi* |
| Keluaran | Solusi teknik *Isolation* |
| Uraian | Proses ini berfungsi untuk mencari dan membangun jembatan yang sudah pasti merupakan solusi, dengan menggunakan teknik *Isolation* |
| Logika proses | 1. Proses ini berjalan otomatis ketika *user* memilih untuk melakukan pencarian dengan cara manual<br>2. Jika dengan menggunakan teknik *Isolation* dan solusi belum |

| | |
|---|---|
| | ditemukan maka proses pencarian akan dilanjutkan menggunakan teknik yang lain<br>3. Jika semua teknik telah digunakan dan solusi belum ditemukan makan proses akan dilanjutkan dengan proses pencarian dengan menggunakan metode BFS/DFS |
| | |
| Nomor | 3.1 |
| Nama | Pencarian menggunakan teknik DFS |
| Masukan | Solusi teknik *Solving Hashi* |
| Keluaran | Solusi Soal *Hashi* |
| Uraian | Proses ini berfungsi untuk mencari dan membangun sisa jembatan yang belum terselesaikan oleh teknik *Solving Hashi*, dengan menggunakan teknik DFS (pencarian mendalam pertama). |
| Logika proses | 1. *User* memilih teknik apa yang akan digunakan.<br>2. Jika *user* memilih teknik DFS maka pencarian DFS akan dilakukan jika solusi pada teknik *Solving Hashi* belum mendapatkan jawaban yang benar.<br>3. Jika solusi ditemukan sebelum teknik DFS dijalankan maka teknik DFS dapat diabaikan. |
| | |
| Nomor | 3.2 |

| | |
|---|---|
| Nama | Pencarian menggunakan teknik BFS |
| Masukan | Solusi teknik *Solving Hashi* |
| Keluaran | Solusi Soal *Hashi* |
| Uraian | Proses ini berfungsi untuk mencari dan membangun sisa jembatan yang belum terselesaikan oleh teknik *Solving Hashi*, dengan menggunakan teknik BFS (pencarian melebar pertama). |
| Logika proses | 1. *User* memilih teknik apa yang akan digunakan.<br>2. Jika *user* memilih teknik BFS maka pencarian BFS akan dilakukan jika solusi pada teknik *Solving Hashi* belum mendapatkan jawaban yang benar.<br>3. Jika solusi ditemukan sebelum teknik BFS dijalankan maka teknik BFS dapat diabaikan. |
| | |
| Nomor | 3.1.1 |
| Nama | Tetapkan teknik *Solving Hashi* sebagai keadaan awal |
| Masukan | Solusi teknik *Solving Hashi* |
| Keluaran | Solusi Soal *Hashi* |
| Uraian | Pada proses ini semua pulau yang belum terhubung akan disimpan untuk dilakukan pencarian kembali |
| Logika proses | 1. Semua pulau yang belum terhubung akan di*lock* untuk dicari kembali |

| | |
|---|---|
| | kemungkinan yang mungkin terlewat pada saat proses *Solving Hashi.*<br>2. Jika masih ada pulau yang belum terselesaikan maka pencarian akan dilakukan pada pulau-pulau yang belum terselesaikan tersebut.<br>3. Jika susunan pulau tidak terurut maka pencarian akan dimulai dari pulau A1 (pulau dengan posisi pertama) |
| | |
| Nomor | 3.1.2 |
| Nama | Telusuri semua kemungkinan arah jembatan secara DFS |
| Masukan | Keadaan awal pulau yang belum terselesaikan oleh teknik *Solving Hashi* |
| Keluaran | Solusi soal *Hashi* |
| Uraian | Semua kemungkinan yang ada dan terlewatkan oleh teknik *Solving Hashi* akan ditelusuri kembali. |
| Logika proses | 1. Setiap pulau yang terdapat pada keadaan awal akan dilakukan pencarian kembali secara DFS.<br>2. Jika pada proses pencarian *solving hashi* masih belum menemukan jawaban maka pencarian dengan teknik DFS akan dilakukan dengan mengembangkan kemungkinan dari pulau-pulau yang belum |

| | |
|---|---|
| | terselesaian. |
| | |
| Nomor | 3.2 |
| Nama | Pencarian menggunakan teknik BFS |
| Masukan | Solusi teknik *Solving Hashi* |
| Keluaran | Solusi Soal *Hashi* |
| Uraian | Proses ini berfungsi untuk mencari dan membangun sisa jembatan yang belum terselesaikan oleh teknik *Solving Hashi*, dengan menggunakan teknik BFS (pencarian melebar pertama). |
| Logika proses | 1. *User* memilih teknik apa yang akan digunakan.<br>2. Jika *user* memilih teknik BFS maka pencarian BFS akan dilakukan jika solusi pada teknik *Solving Hashi* belum mendapatkan jawaban yang benar.<br>3. Jika solusi ditemukan sebelum teknik BFS dijalankan maka teknik BFS dapat diabaikan. |
| | |
| Nomor | 3.2.1 |
| Nama | Tetapkan teknik *Solving Hashi* sebagai keadaan awal |
| Masukan | Solusi teknik *Solving Hashi* |
| Keluaran | Solusi Soal *Hashi* |
| Uraian | Pada proses ini semua pulau yang belum terhubung akan disimpan untuk dilakukan pencarian kembali |

| | |
|---|---|
| Logika proses | 1. Semua pulau yang belum terhubung akan di*lock* untuk dicari kembali kemungkinan yang mungkin terlewat pada saat proses *Solving Hashi.*<br>2. Jika masih ada pulau yang belum terselesaikan maka pencarian akan dilakukan pada pulau-pulau yang belum terselesaikan tersebut.<br>3. Jika susunan pulau tidak terurut maka pencarian akan dimulai dari pulau A1 (pulau dengan posisi pertama) |
| | |
| Nomor | 3.2.2 |
| Nama | Telusuri semua kemungkinan arah jembatan secara BFS |
| Masukan | Keadaan awal pulau yang belum terselesaikan oleh teknik *Solving Hashi* |
| Keluaran | Solusi soal *Hashi* |
| Uraian | Semua kemungkinan yang ada dan terlewatkan oleh teknik *Solving Hashi* akan ditelusuri kembali. |
| Logika proses | 1. Setiap pulau yang terdapat pada keadaan awal akan dilakukan pencarian kembali secara BFS.<br>2. Jika pada proses pencarian *solving hashi* masih belum menemukan jawaban maka pencarian dengan teknik BFS akan dilakukan dengan |

| | |
|---|---|
| | mengembangkan kemungkinan dari pulau-pulau yang belum terselesaian. |

### 3.1.5 Perancangan

Aplikasi *game puzzle Hashi* yang dilengkapi dengan solusi ini dirancang dengan menggunakan bahasa pemograman *Microsoft Visual Basic* 6.

#### 3.1.5.1 Perancangan Kamus Data

Dengan menggunakan *Microsoft Access* 2003 untuk menyimpan soal *Hashi*. Di dalam *database* yang digunakan, terdapat satu buah tabel, yaitu: ”Soal*Hashi*”. Tabel tersebut berfungsi untuk menyimpan nama, ukuran, level soal, posisi dan bobot pulau *Hashi*.

Struktur tabel *header* dapat dilihat pada tabel 3.2 berikut.

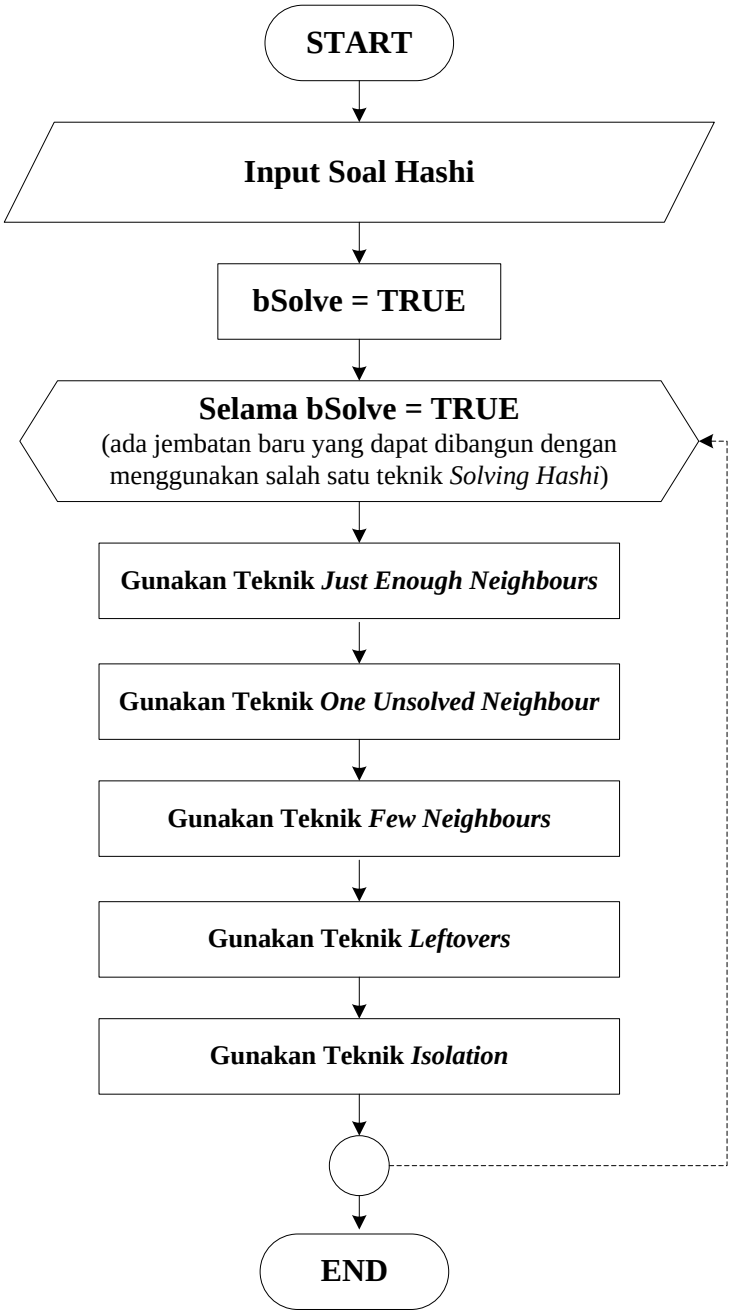
**Tabel 3.2 Struktur Tabel ”Soal*Hashi*”**

| No. | Field Name | Data Type | Field Size |
|---|---|---|---|
| **1.** | **NamaSoal** | **Text** | **20** |
| 2. | JlhBaris | Number | Default Value |
| 3. | JlhKolom | Number | Default Value |
| 4. | Level | Number | Default Value |
| 5. | Posisi Pulau | Number | Default Value |
| 6. | Bobot Pulau | Number | Default Value |

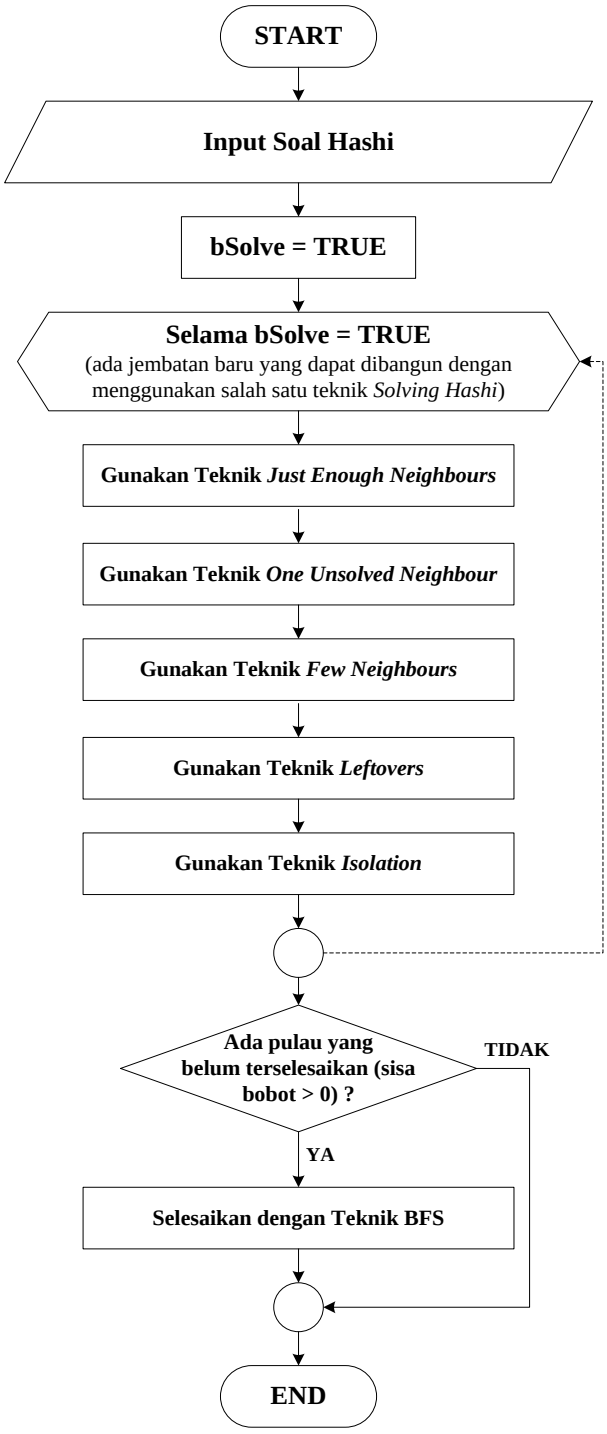
*Field* ”NamaSoal” menyimpan nama dari soal *Hashi*. *Field* ”JlhBaris” dan ”JlhKolom” menyimpan ukuran *Hashi*. Bila ukuran *Hashi* adalah 12 x 10, maka *field* ”JlhKolom” bernilai 12 dan *field* ”JlhBaris” bernilai 10. Field ”Level” menyimpan level kerumitan soal *Hashi* yang di-*input* berdasarkan buku *Hashi* yang ditulis ”Saito Yamamoto”. Level soal ”Sangat mudah” ditandai dengan nilai level 1, ”Mudah” mempunyai nilai level 2, ”Sedang” – mempunyai nilai level 3, ”Rumit” – mempunyai nilai level 4 dan ”Sangat rumit” – mempunyai nilai level 5.

### 3.1.5.2 Flow Chart

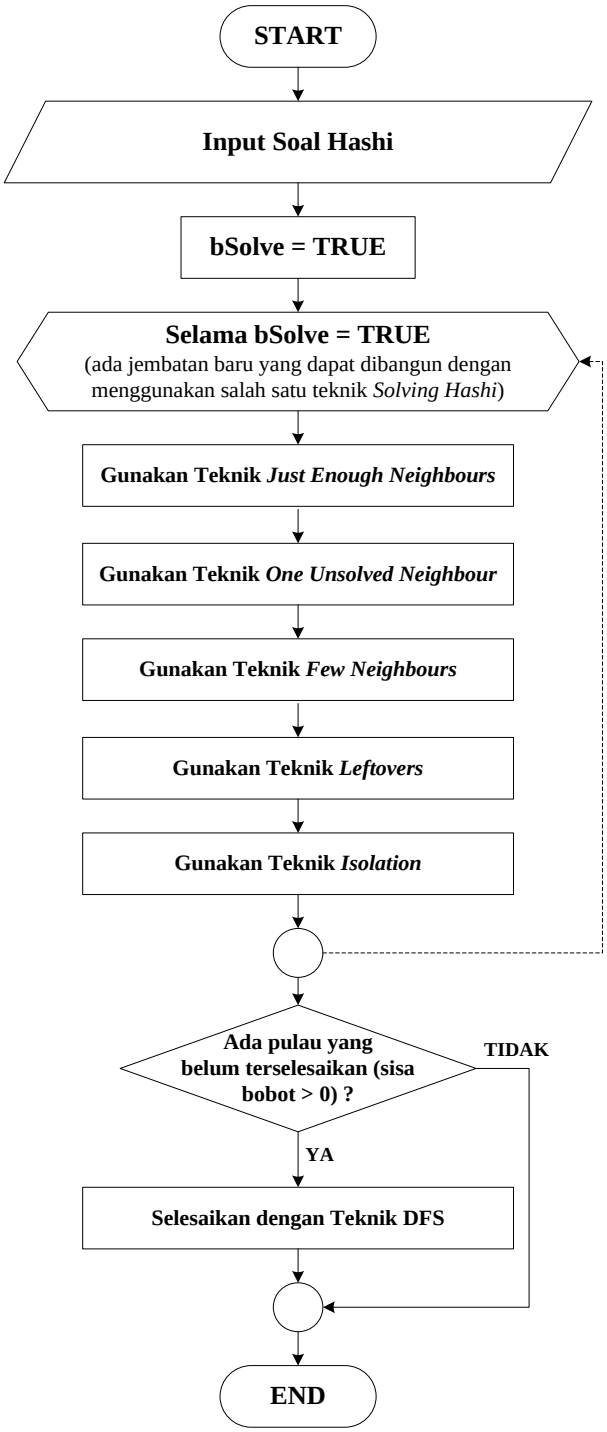
Secara umum, skema pencarian solusi dengan menggunakan teknik *Solving Hashi* dan teknik BFS/DFS dapat digambarkan dalam bentuk *flowchart*, seperti terlihat pada gambar berikut.




**Gambar 3.31 Flowchart Pencarian Jawaban Soal Hashi Manual**




**Gambar 3.32 Flowchart Pencarian Jawaban Soal Hashi Menggunakan Teknik BFS**




**Gambar 3.33 Flowchart Pencarian Jawaban Soal Hashi Menggunakan Teknik DFS**

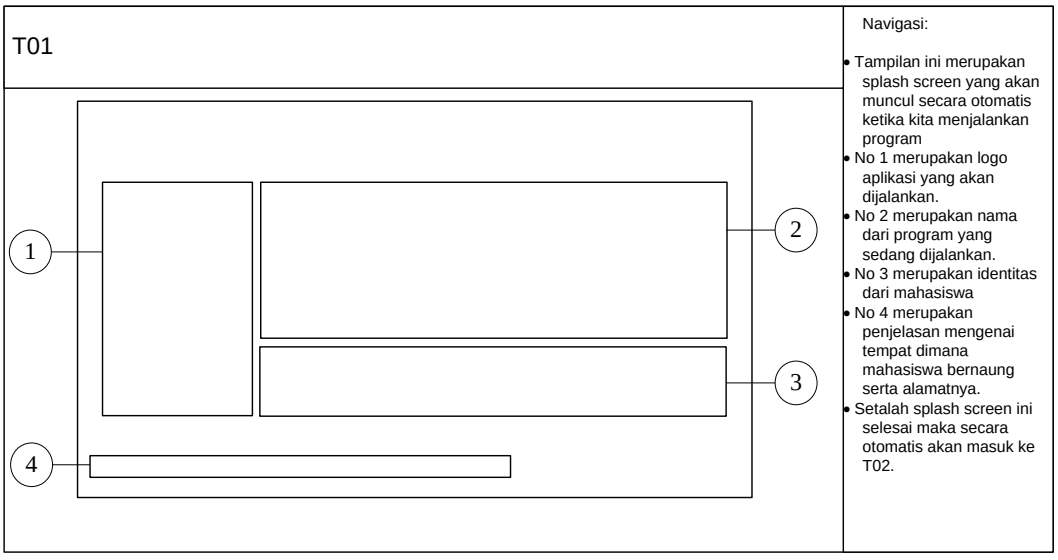
### 3.1.5.3 Perancangan Antar Muka

Aplikasi mempunyai beberapa tampilan, diantaranya adalah:

1. *Form Splash Screen*
2. *Form* Utama
3. *Form* Pulau
4. *Form* Simpan Soal
5. *Form* Buka Soal
6. *Form About*
7. *Form* Manual

#### 3.1.5.3.1 *Form Splash Screen*

Tampilan ini berfungsi sebagai *form* pembuka yang berisi nama dan logo aplikasi, nama dan NIM pembuat aplikasi serta nama universitas dan nama program studi. Perancangan tampilan *form* ini dapat dilihat pada gambar 3.34.




**Gambar 3.34 Rancangan *Form Splash Screen***

Keterangan gambar:

1 . logo aplikasi

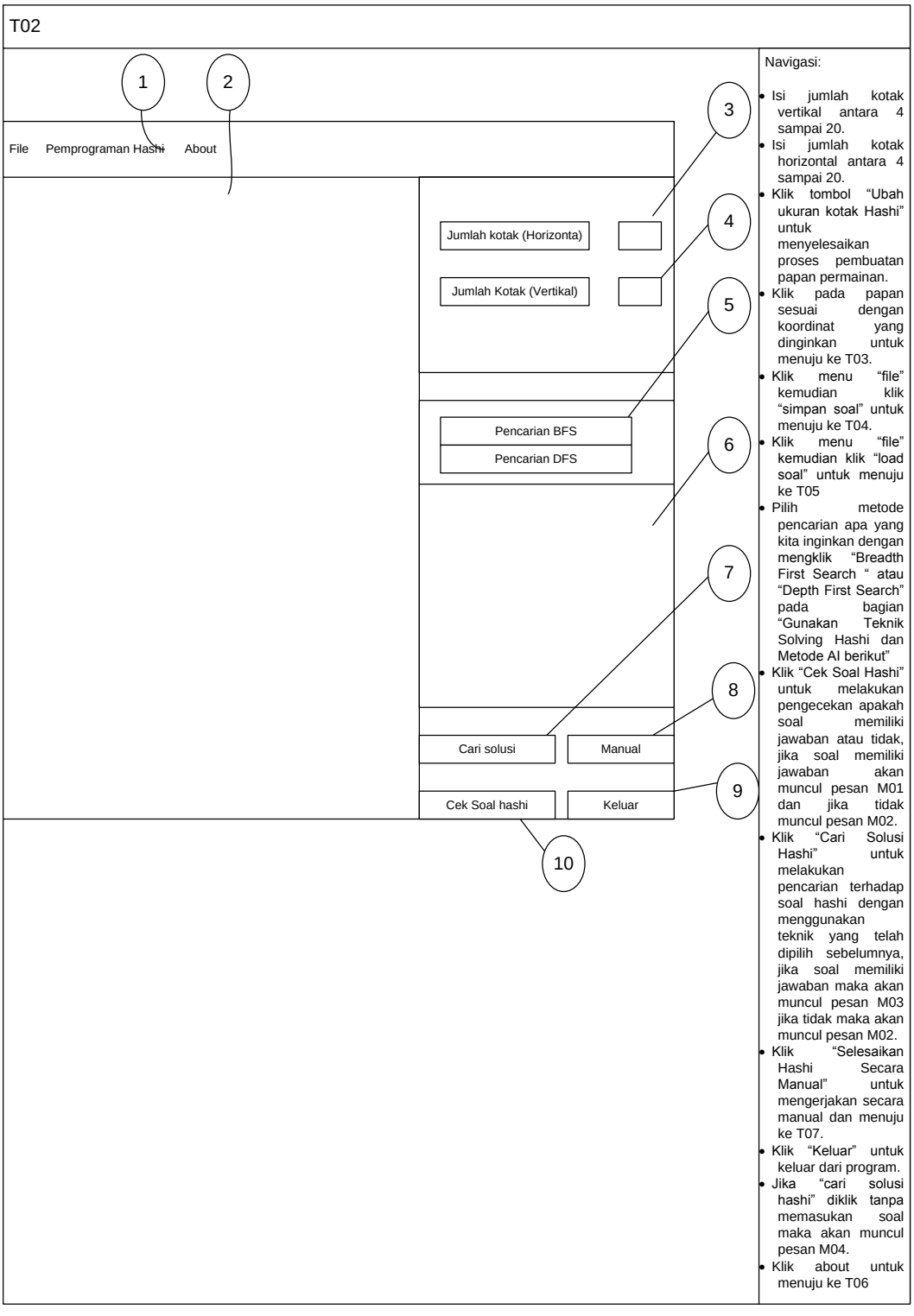
2 . nama aplikasi

3 . identitas mahasiswa

4 . nama sekolah tinggi, kota dan tahun pembuatan

#### 3.1.5.3.2 *Form* Utama

*Form* Utama berfungsi untuk membuat soal *Hashi* dan mempunyai sejumlah menu untuk menjalankan fungsi utama program. Pada *form* ini, *user* dapat memilih ukuran *Hashi* (dari 9x9 sampai 20x20) dan memilih metode pencarian yang akan digunakan, apakah BFS atau DFS. Sedangkan teknik *Solving Hashi* tetap akan dijalankan terlebih dahulu sebelum pencarian dengan metode BFS atau DFS. Perancangan tampilan *form* ini dapat dilihat pada gambar 3.35.




**Gambar 3.35 Rancangan *Form* Utama**

Keterangan gambar:

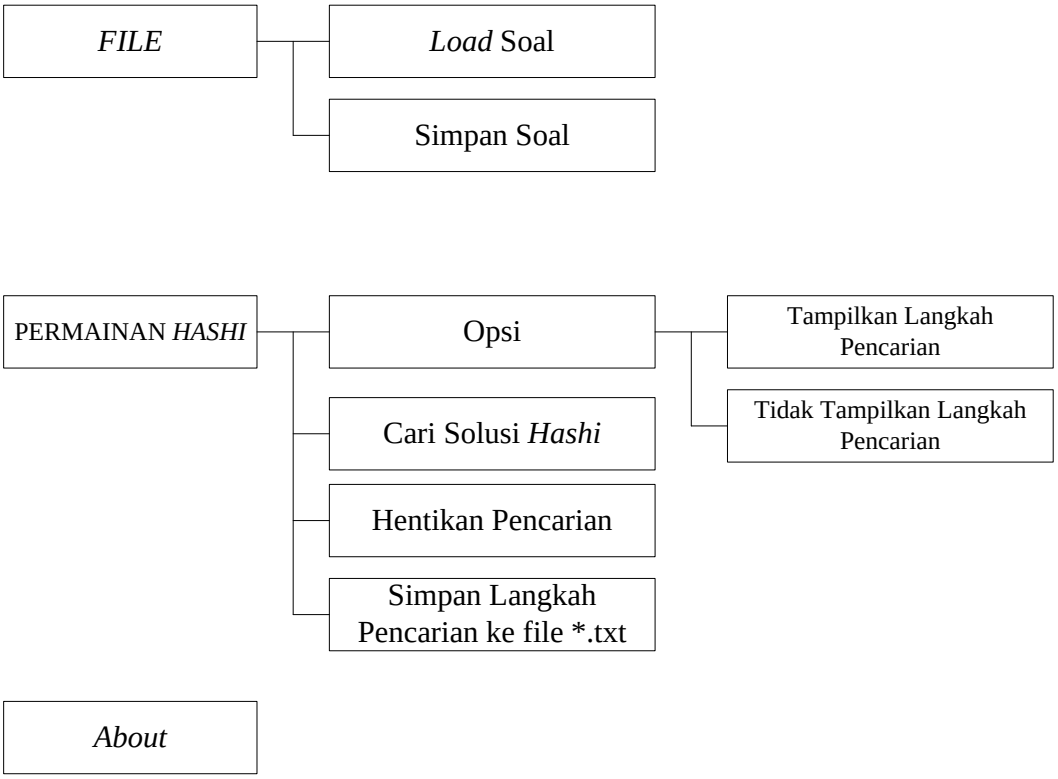
1 . menu aplikasi, yang memiliki rancangan sebagai berikut:

a. Menu *File*, terdiri atas beberapa submenu berikut:
    1) submenu "*Load* Soal", akan membuka *form* Buka Soal.
    2) submenu "Simpan Soal", akan membuka *form* Simpan Soal.

b. Menu Permainan, terdiri atas beberapa submenu berikut:
    1) submenu "Opsi" - "Tampilkan Langkah Pencarian", berfungsi sebagai pilihan untuk menampilkan langkah pencarian.
    2) Submenu "Opsi" – "Tidak Tampilkan Langkah Pencarian", berfungsi sebagai pilihan untuk tidak menampilkan langkah pencarian. Bila langkah pencarian tidak ditampilkan maka proses pencarian akan menjadi lebih cepat.
    3) submenu "Cari Solusi *Hashi*", berfungsi untuk memulai proses pencarian solusi *Hashi*.
    4) submenu "Hentikan Pencarian", untuk menghentikan proses pencarian solusi yang sedang berlangsung.
    5) submenu "Simpan Langkah Pencarian", berfungsi untuk menyimpan langkah pencarian ke *file* *.txt.

c. Menu *About*, berfungsi untuk menampilkan *form About*.

Rancangan struktur menu dapat dilihat pada gambar 3.36 berikut.




**Gambar 3.36 Rancangan Struktur Menu**

2 . daerah tampilan pulau yang dirancang oleh *user*.

3 . *updown*, untuk memilih jumlah baris ukuran *Hashi*.

4 . *updown*, untuk memilih jumlah kolom ukuran *Hashi*.

5 . *optionbutton*, untuk memilih opsi penggunaan metode BFS atau DFS.

6 . *textbox*, untuk menampilkan langkah pencarian.

7 . tombol "Cari", untuk memulai pencarian solusi.

8 . tombol "Selesaikan", untuk membuka *form* Penyelesaian Manual.

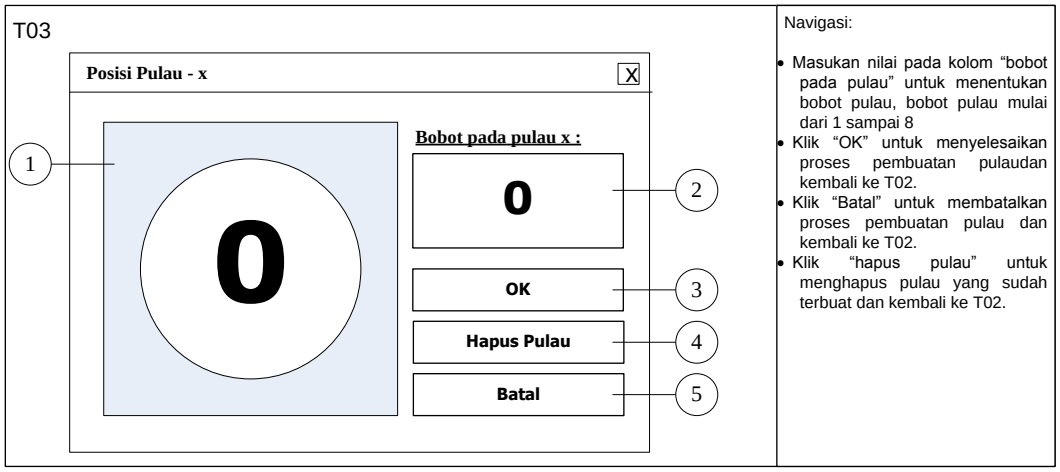
9 . tombol "Cek Soal Hashi", untuk memeriksa apakah soal Hashi mempunyai jawaban atau tidak.

10 . tombol "Keluar", untuk menutup *form*.

#### 3.1.5.3.3 *Form* Pulau

*Form* ini berfungsi untuk membuat pulau baru, mengubah bobot pulau atau menghapus pulau yang telah dirancang sebelumnya. *Form* ini muncul ketika

*user* melakukan klik *mouse* pada daerah perancangan kotak Hashi pada *form* Utama. Perancangan tampilan *form* dapat dilihat pada gambar 3.37.




**Gambar 3.37 Rancangan *Form* Pulau**

Keterangan gambar:

1 . *picturebox*, berisi gambar pulau beserta bobotnya.

2 . *textbox*, untuk memasukkan bobot pulau.

3 . tombol "OK", untuk membentuk pulau dan kembali ke *form* Utama.

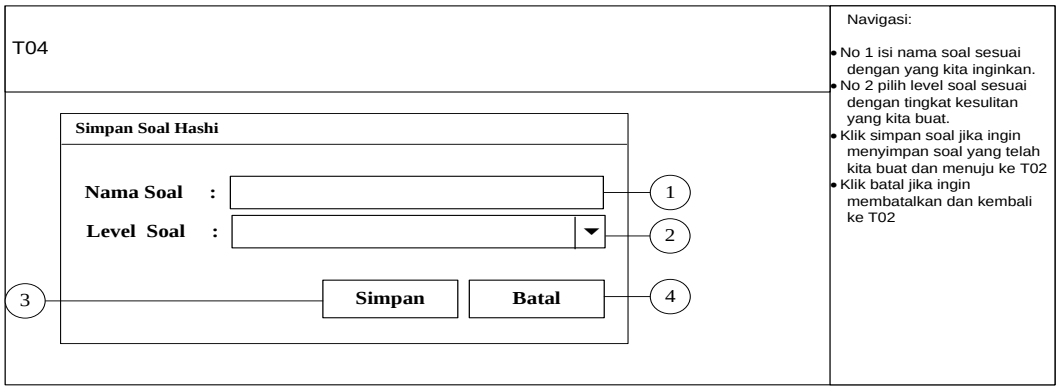
4 . tombol "Hapus Pulau", untuk menghapus pulau dan kembali ke *form* Utama.

5 . tombol "Batal", untuk kembali ke *form* Utama.

6 . tombol "Batal", untuk membatalkan proses pencarian.

### 3.1.5.3.4 *Form* Simpan Soal

*Form* ini berfungsi untuk menyimpan soal *Hashi*. Perancangan tampilan *form* dapat dilihat pada gambar 3.38.




**Gambar 3.38 Rancangan *Form* Simpan Soal**

Keterangan gambar:

1 . *textbox*, untuk memasukkan nama soal.

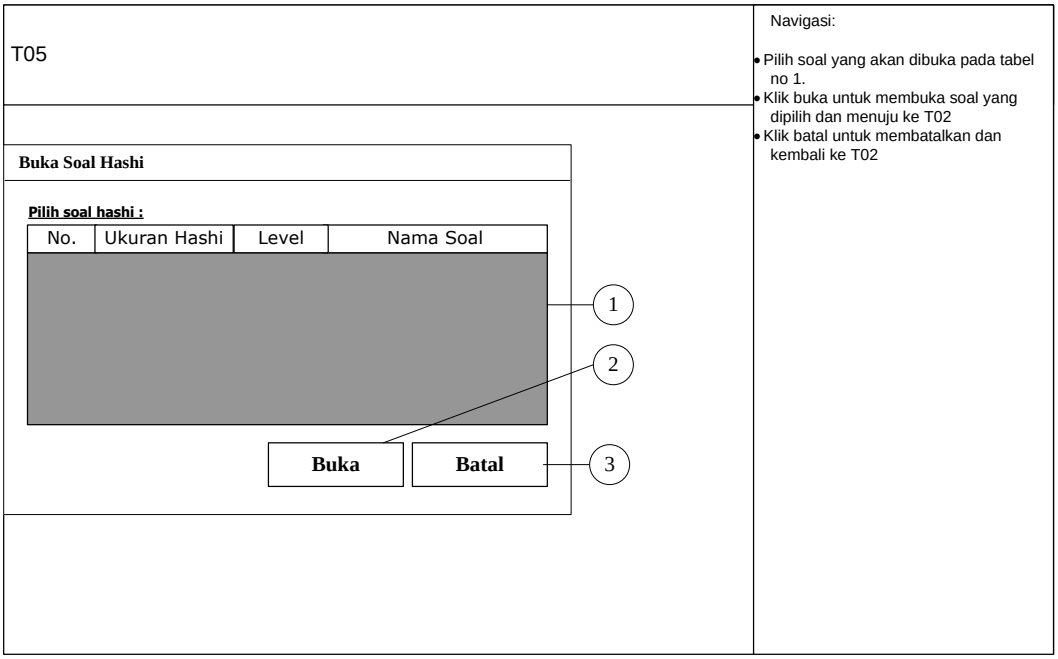
2 . *combobox*, untuk memilih level soal.

3 . tombol “Simpan”, untuk menyimpan soal.

4 . tombol “Batal”, untuk membatalkan proses penyimpanan.

### 3.1.5.3.5 *Form* Buka Soal

*Form* ini berfungsi untuk membuka soal *Hashi*. Perancangan tampilan *form* dapat dilihat pada gambar 3.39.




**Gambar 3.39 Rancangan *Form* Buka Soal**

Keterangan gambar:

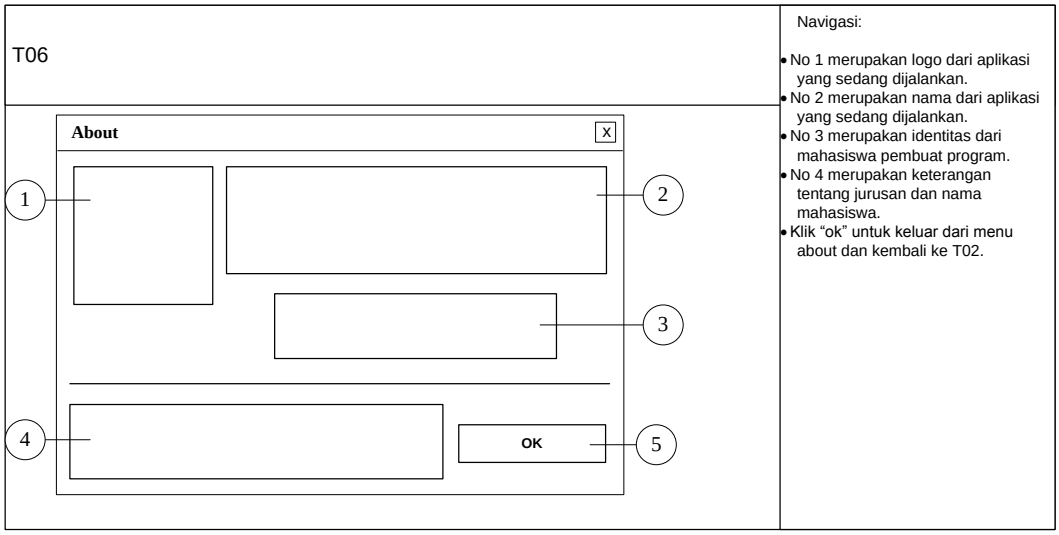
1 . tabel, untuk menampilkan ukuran *Hashi*, level dan nama soal.

2 . tombol “Buka”, untuk membuka soal yang terpilih.

3 . tombol “Batal”, untuk menutup *form*.

### 3.1.5.3.6 *Form About*

*Form About* berfungsi untuk menampilkan identitas mahasiswa dan pembuat aplikasi. Perancangan tampilan *form* dapat dilihat pada gambar 3.40.




**Gambar 3.40 Rancangan *Form About***

Keterangan gambar:

1 . logo aplikasi.

2 . nama aplikasi.

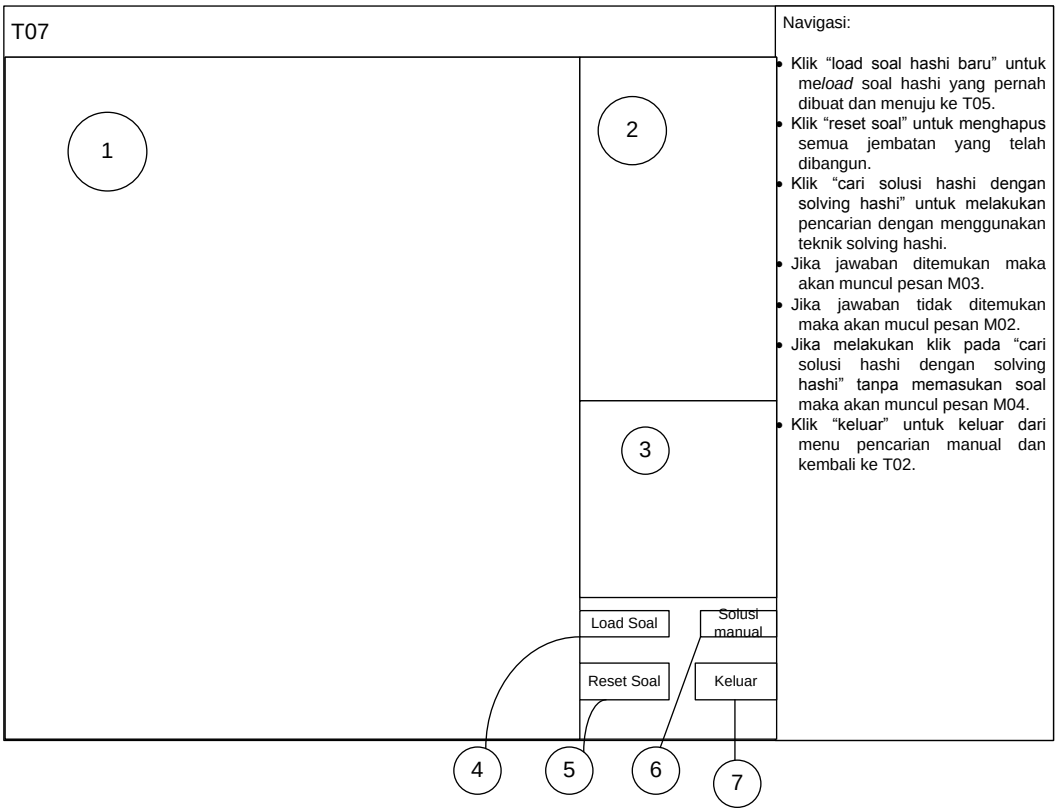
3 . nama dan NIM pembuat aplikasi.

4 . jurusan, program studi dan nama mahasiswa.

5 . tombol untuk menutup *form.*

### 3.1.5.3.7 *Form* Manual

*Form* manual berfungsi untuk menyelesaikan soal *Hashi* secara manual dimana pada *form* ini dijelaskan aturan-aturan yang harus diikuti dalam mengerjakan soal *Hashi.* Selain itu pada form manual ini kita juga dapat menggunakan teknik *solving hashi* yang dapat dipilih jika *user* merasa kesulitan dalam mengerjakan soal *Hashi* seperti yang terlihat pada gambar 3.41.



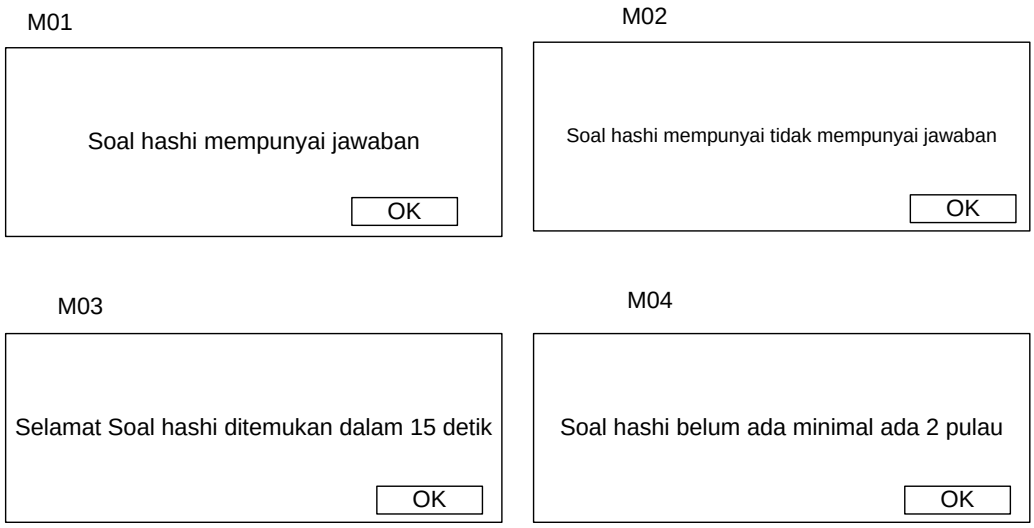

**Gambar 3.41 Rancangan *Form* Manual**

Keterangan gambar:

1. Papan permainan *Hashi.*
2. Aturan permainan *Hashi.*
3. Tampilan teknik-teknik yang digunakan dalam proses pencarian solusi.
4. Tombol “*Load* Soal” , untuk membuka soal.
5. Tombol “*Reset* Soal”, untuk menghapus semua jembatan yang telah dibangun.
6. Tombol “Solusi *Hashi*”, untuk melakukan pencarian dengan menggunakan teknik *Solving Hashi*.
7. Tobol “Keluar”, untuk keluar dari tampilan *Form* Manual.

#### 3.1.5.3.8 Rancangan tampilan pesan

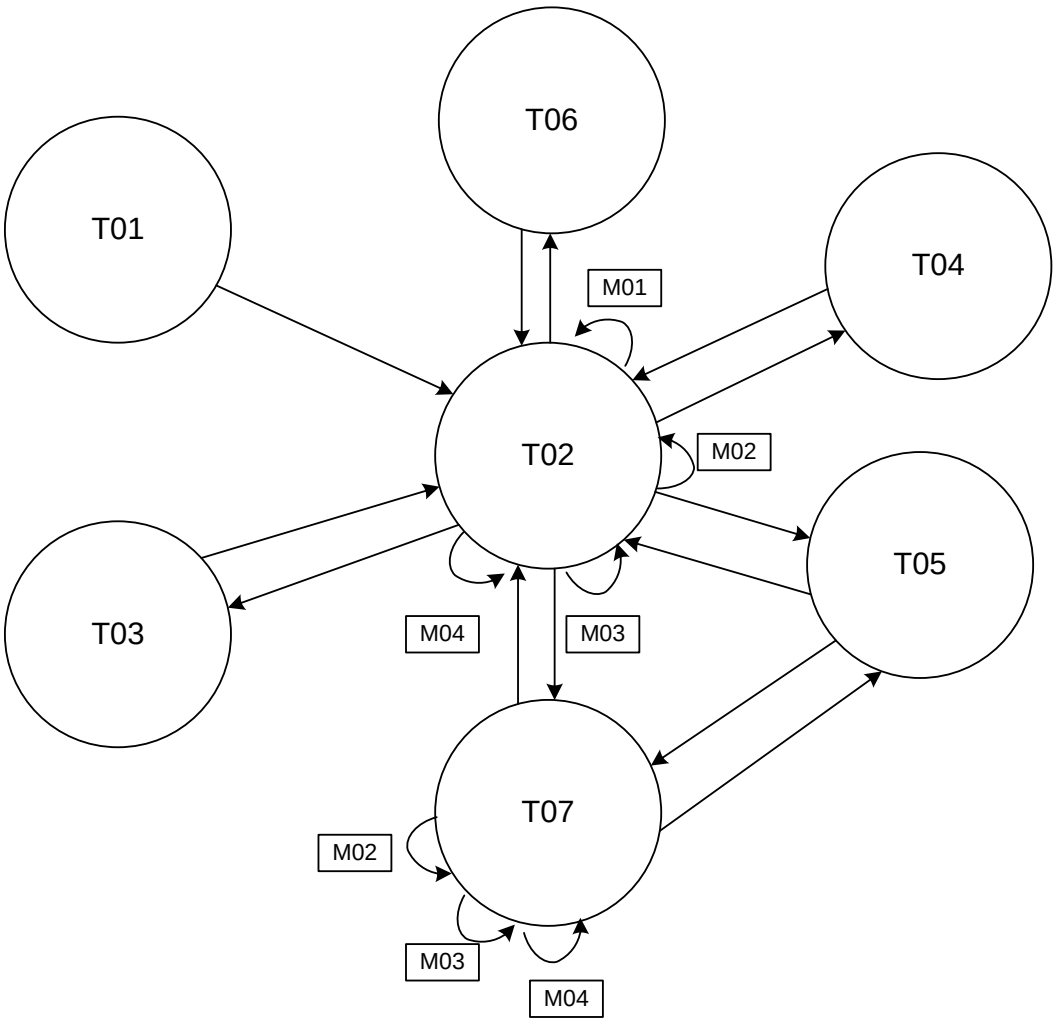
Pada gambar 3.42 adalah perancangan pesan yang muncul sebagai informasi kepada pengguna dalam mengoperasikan sistem.




**Gambar 3.42 Rancangan tampilan pesan**

### 3.1.5.4 Jaringan Semantik

Perancangan jaringan semantik menggambarkan alur informasi dari setiap antarmuka aplikasi yang dijelaskan pada gambar 3.43.




**Gambar 3.43 Jaringan semantik**