BAB 2
LANDASAN TEORI

2.1 Permainan (*Game)*

Game berasal dari kata bahasa inggris yang berarti dasar permainan. Permainan dalam hal ini merujuk pada pengertian kelincahan intelektual *(Intellectual Playability Game)* yang juga bisa diartikan sebagai arena keputusan dan aksi pemainnya. Dalam *game*, ada target-target yang ingin dicapai pemainnya.

Permainan adalah kegiatan yang kompleks yang didalamnya terdapat peraturan, *play* dan budaya. Sebuah permainan adalah sebuah sistem dimana pemain terlibat dalam konflik buatan, disini pemain berinteraksi dengan sistem dan konflik dalam permainan merupakan rekayasa atau buatan, dalam permainan terdapat peraturan yang bertujuan untuk membatasi perilaku pemain dan menentukan permainan [4].

Game bertujuan untuk menghibur, *game* banyak disukai oleh anak – anak hingga orang dewasa. *Game* sebenarnya penting untuk perkembangan otak, untuk meningkatkan konsentrasi dan melatih untuk memecahkan masalah dengan tepat dan cepat karena dalam *game* terdapat berbagai konflik atau masalah yang menuntut kita untuk menyelesaikannya dengan cepat dan tepat. G*ame* juga bisa merugikan karena apabila kita sudah kecanduan *game* kita akan lupa waktu dan akan mengganggu kegiatan atau aktivitas yang sedang kita lakukan.

2.1.1 Kriteria *Game*

Pada sebuah *game* ada penggolongan jenis *game*, baik *game* tradisional maupun *game* modern. Pada *game* modern penggolongan jenis *game* dibagi menjadi beberapa jenis yaitu [4]:

1. Petualangan (*Adventure)*

   Game jenis petualangan (*adventure)* merupakan satu *game* yang mencapai tujuan tertentu dengan berbagai tantangan, *game* ini mengutamakan masalah

eksplorasi dan pemecahan teka-teki. Terkadang meliputi masalah konseptual, dan tantangan fisik namun sangat jarang. *Game adventure* telah berkembang jauh hingga menjadi *genre* campuran (*Action bebat-em).* Dalam *game* lebih menekankan pada jalan cerita dan kemampuan berpikir pemain untuk menganalisa tempat secara visual, memeccahkan teka-teki maupun menyimpulkan rangkaian peristiwa dan percakapan. Salah satu *game adventure* adalah *game* Super Mario.

2. RPG (*Role Playing Game)*

   Pada tipe RPG pemain memainkan tokoh khayalan dan berkolaborasi untuk merajut sebuah cerita bersama. Para pemain memilih tokoh – tokoh mereka berdasarkan karakteristik tokoh tersebut, dan keberhasilan aksi mereka tergantung dari sistem peraturan yang telah ditentukan. Cerita dalam *game* ini kebanyakan berceritakan tentang sejarah, kebudayaan sesuatu tempat. Karena sebagian besar *game* tipe ini berawal dari Asia timur, khususnya Jepang, maka secara pengembangan *game* ini bisanya membuat cerita dengan arah linear. Contoh dari *game* ini adalah final fantasy, megaman X: command Mission, dan Ninja saga.

3. Strategy

   Game strategi adalah jenis permaian seperti simulasi, dengan tujuan jelas, sehingga pemain harus pintar mengolah strategi dalam memecahkan tipe *game* ini. *Game* strategi merupakan kebalikan dari *game action* yang berjalan cepat dan perlu refleks secepat kilat, akan tetapi *game* ini pemain dituntun untuk berhati hati karena bisa jadi salah menerapkan strategi bermain permainan menuju *game over*. *Game* strategi biasanya memberikan pemain atas kendali tidak hanya satu orang tapi minimal dua tang atau kelompok. *Game* strategi dibagi menjadi dua yaitu [4]:

   a. Real Time strategy

      Real Time strategy merupakan *game* berjalan dalam waktu sebenarnya dan serentak antara semua pihak dan pemain harus memutuskan setiap langkah yang diambil saat itu juga berbarengan mungkin saat itu pihak lawan juga sedang mengeksekusi strateteginya.

Contoh *game* ini adalah *Company of heroes (2006), Cage Empire II : The Cage of Kings (1999), Rome : total bar (2004)* dan sebagainya.

b. Turn based strategy

Game ini dilakukan secara bergiliran, saat kita mengambil keputusan dan menyegerakan pasukan, saat itu isak lawan menunggu, begitu pula sebaliknya. Contoh : Catur, Chekers, Gunrox, Silent Strom, dan advance War.

4. Puzzle

Game ini sesuai bercerita mengenai pemecahan teka-teki, baik itu menyusun blok, menyamakan warna bola, memecahkan perhitungan matematika, melewati labirin, sampai mendorong-dorong kotak sampai masuk ke tempatnya. Contoh dari *game* ini adalah Mahjong-Naga, Square Freenzy, dan Typer Shark Deluxe.

5. Arcade

Game arcade adalah jenis permainan *game* dengan objek yang bergerak cepat, "membahayakan" atau "menyerang" pemaian. *Game* disajikan dalam dua dimensi, sangat ringan untuk dimainkan sehingga sering disebut mini *game*. *Game arcade* digolongkan berdasarkan beberapa aspek didalamnya yaitu : kecepatan reaksi (*Speed*), kepintaran (*cleverness*), pengalaman (*experience*), pemahaman (*insight*), surprise, dan emosi.

2.2 Permainan Tradisional

Permainan tradisional adalah suatu jenis permainan yang ada pada satu daerah tertentu yang berdasarkan kepada kultur atau budaya daerah tersebut. Permainan tradisional biasanya dimainkan oleh orang-orang pada daerahnya tertentu dengan aturan dan konsep yang tradisional pada jaman dulu [5]. Permainan tradisional kurang begitu dikenal oleh anak-anak pada jaman sekarang kebanyakan orang tua jaman dulu yang masih sangat tahu bagaimana memainkan permainan ini khususnya mengenang masa kecil mereka bermain permainan tradisional ini.

Permainan tradisional merupakan unsur-unsur kebudayaan yang tidak dapat dianggap remeh, karena permainan tradisional memberikan pengaruh yang besar

terhadap perkembangan kejiwaan, sifat, dan kehidupan sosial anak dikemudian hari. Permainan tradisonal mendapat pengaruh yang kuat dari budaya setempat, oleh karena itu permainan tradisonal mengalami perubahan baik berupa pergantian, penambahan maupun pengurangan sesuai dengan kondisi daerah setempat [1]. Dengan demikian, permainan tradisional meskipun nama permainannya berbeda antar daerah, namun memiliki persamaan atau kemiripan dalam cara memainkannya.

Salah satu pengaruh Permainan tradisional besar untuk pemainnya seperti Mengembangkan kecerdasan logika anak, anak dituntut untuk menyelesaikan permainan dengan urutan yang sesuai dan menentukan langkah yang harus dilaluinya. Beberapa contoh permainannya anatara lain Damdaman, Congklak, Lompat tali atau Sapintrong, dan Sondah [1].

2.3 Damdaman

Damdaman merupakan salah satu permainan tradisional yang mampu mengembangkan kecerdasan logika anak. Permainan Damdaman berasal dari daerah Jawa sekalipun di daerah lain ada. Nama permainan ini beragam disetiap daerah seperti di masyarakat yogya mengenal dengan macanan, kemudian di masyarakat Betawi permainan ini dikenal dengan nama damdas, dan di jawabarat permaian ini dikenal dengan Damdaman. Dengan berbedanya sebutan disetiap daerah, maka dalam aturan yang digunakan sedikit berbeda, karena tidak adanya aturan yang jelas tertulis [5].

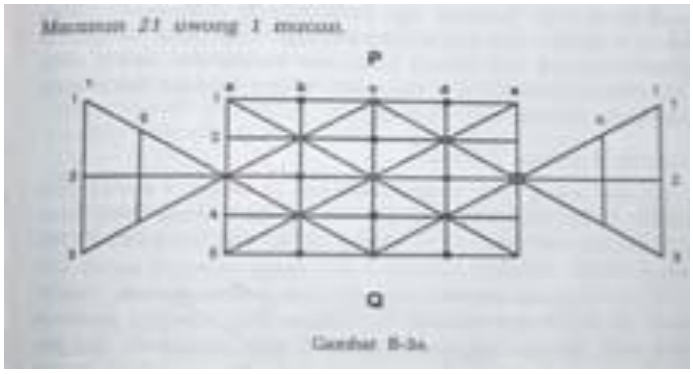
Mayoritas anak masyarakat Jawa, setidaknya yang berada di daerah Yogyakarta, yang hidup di era tahun 1970-an mengenal sebuah permainan tradisional yang bernama Macanan. permainan ini banyak digemari oleh anak laki-laki karena bermasuk permainan kompetitif, mengasah otak, ada yang kalah dan menang. Walaupun tidak ada larangan bagi anak wanita untuk bermain, namun kebanyakan dimainkan oleh anak laki-laki. Permainan Damdaman sudah dikenal sejak sebelum Perang Dunia (PD) II. Hanya saja untuk saat sekarang sudah sangat jarang dimainkan karena tergeser dolanan modern lain, seperti halma dan lainnya [1].

Permainan Damdaman dimainkan oleh 2 orang, setiap pemain memiliki 16 bidak yang disusun di arena main, adapun arena main yang digunakan bisa diatas papan atau pun diatas kertas karton yang telah diberi arena main. Pemain harus berpikir bagaimana menyusun strategi untuk menyusun bidaknya agar bisa bergerak menyerang ke daerah lawan dengan tujuan menghabiskan bidak lawan dengan aturan yang disepakati sebagai akhir permainannya. Permainan ini bertujuan untuk melatih daya ingat dan kepekaan indera serta melatih ketelitian dan kecermatan dalam menentukan keputusan.

2.3.1 Pelaksanaan Permainan Damdaman

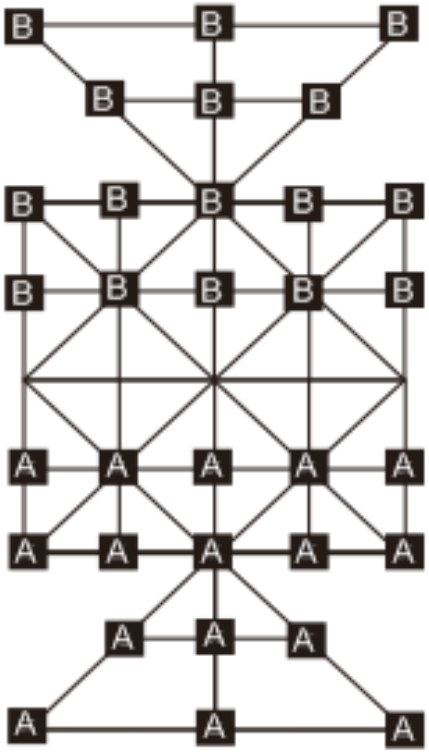
Pada permainan tradisional selain pemain dituntut untuk bisa memainkan permainan, pemain pun dituntut untuk kreatif menentukan tempat permainan. Berikut akan dijelaskan pelaksanaan permainan Damdaman [1].

a. Apabila ada dua anak, misalkan pemain A dan B, atau lebih (dalam kelompok) yang ingin bermain Damdaman, maka mereka segera bersepakat untuk menentukan tempat bermain. Tidak lupa masing-masing pemain juga sudah harus mempersiapkan orang-orangan berupa kerikil dan sejenisnya. Setiap orang dalam memilih orang-orangan bebas, asalkan satu pemain dengan pemain lain tidak boleh sama. Artinya jika pemain A telah memilih orang-orangan berupa kecik tanjung, maka pemain B harus menggunakan kerikil. Pembedaan orang-orangan ini hanya untuk memudahkan permainan agar tidak keliru. Setiap pemain harus mencari orang-orangan sebanyak 16 buah.
b. Setelah itu mereka membuat arena bermain yang berbentuk seperti pada gambar 2.1.



Gambar 2. 1 Papan Permainan Damdaman [1]

c. Langkah pertama, kedua pemain menyusun 16 bidak pada arena main, mulai dari daerah segitiga hingga daerah kotak. Sehingga bidak tersusun seperti pada gambar 2.2.

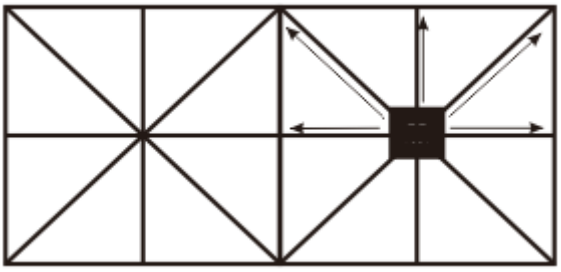


Gambar 2. 2 Susunan Bidak Pada Permainan Damdaman

d. Setelah kedua pemain menyusun bidak, maka pemain membuat kesepakatan untuk menentukan yang terlebih dalu main.
e. Permainan berakhir jika salah satu bidak dari pemain dan menghasilkan satu pemain.

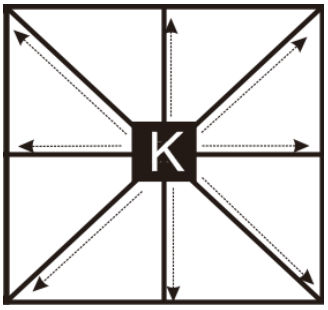
2.3.2 Langkah Bidak Pada Permainan Damdaman

Langkah bidak pada permainan Damdaman terdiri dari dua langkah yang berbeda, yaitu langkah bidak biasa dan langkah bidak raja. pada gambar 2.3 akan dijelaskan langkah bidak biasa.

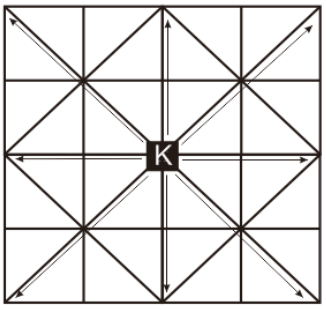


Gambar 2. 3 Langkah Bidak Biasa

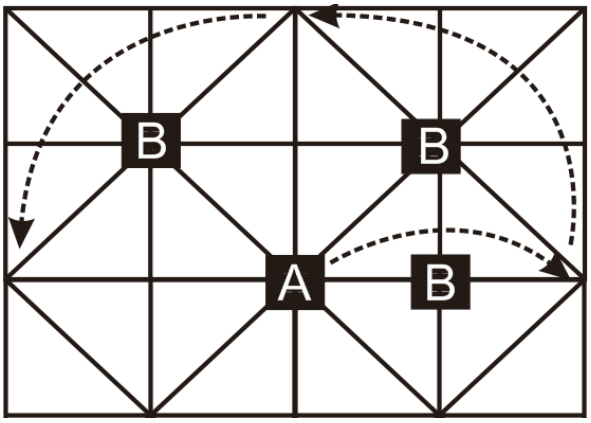
Bidak biasa bergerak maju, kekiri, kanan, miring satu langkah ke daerah lawan. Sedangkan untuk bidak raja langkah bisa lebih dari satu langkah, seperti gambar 2.4 dan gambar 2.5.



Gambar 2. 4 Langkah Bidak Raja Satu Langkah



Gambar 2. 5 Langkah Bidak Raja Dua Langkah



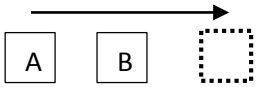
Gambar 2. 6 langkah Bidak A memakan Bidak B

Kesempatan bidak untuk memakan bidak lawan sesuai dengan kondisi papan, seperti diilustrasikan pada gambar 2.6. bidak A akan memakan bidak B maka bidak A akan melewati bidak B, jika pada jalur yang dilangkahi ada bidak lawan yang dekat satu langkah dengan jalur yang dilangkahi maka bidak A bisa mekan bidak B di jalur tersebut, dan begitu seterusnya.

2.3.3 Aturan Permainan Damdaman

Aturan permainan yang digunakan disetiap daerah sangat beragam, berikut akan dijelas aturan permainan dari daerah jawabarat [1].

a. Bidak dibagi menjadi 2 yaitu bidak biasa dan bidak raja.
b. Pada awal permainan, status bidak adalah bidak biasa.
c. Bidak biasa bisa menjadi bidak raja, jika bidak biasa telah berhasil masuk darah segitiga lawan.
d. Langkah bidak biasa melangkah satu langkah maju, ke kiri, kanan, dan miring sesuai dengan kondisi papan.
e. Langkah bidak raja melangkah lebih dari satu langkah ke semua arah, selama pada jalur tersebut tidak ada bidak kawan.



Gambar 2. 7 Langkah Makan Bidak A

f. Cara memakan bidak lawan adalah melangkahkan bidak melewati bidak lawan. Dan bidak bisa memakan bidak lawan jika bidak lawan ada satu langkah bidaknya.
g. Permainan berakhir jika salah satu bidak pemain habis.

2.4 AI (artificial Intellegence)

Para ahli mendefinisikan AI secara bebeda-beda tergantung pada sudut pandang mereka asing-masing. Ada yang fokus pada logika berpikir manusia saja. Tetapi ada juga yang mendefinisikan AI secara lebih luas pada tingkah laku manusia. Pengelompokan definisi AI ada empat kategori [14], yaitu :

1. *Thinking Humanly : The Cognitive Modeling Approach*

   Pendekatan ini dilakukan dengan dua cara yaitu :

   a. Melalui introjeksi : mencoba menangkap pemikiran – pemikiran kita sendiri saat berpikir.
   b. Melalui eksperimen-eksperimen psikologi.

2. *Actual Humanly : The Turing Test Approach*

   Kecerdasan buatan dalam kategori ini dirancang oleh Alan Turing, pada tahun 1950 beliau merancang sebuah komputer berintelejensia dengan tujuan apakah komputer tersebut mampu mengelabui seorang manusia yang mengintrograsinya melalui *teletype (*komunikasi berbasis pesan jarak jauh*).* Jika integrator tidak dapat membedakan yang diintrograsi adalah manusia atau komputer maka komputer tersebut lulus dari *turing Test.* Untuk menerapkan kelompok AI ini komputer harus memiliki kemampuan mengolah bahasa secara natural, perangkat belajar berdasarkan kondisi apapun, representasi pengetahuan, pengambilan keputusan, *Computer Vision, robotics.* turing Test sengaja menghindari interaksi fisik antara integrator dan komputer karena simulasi fisik manusia tidak memerlukan intelejensia.

3. *Thinking Rationally : The Laws Of Trought Approach*

   Masalah pada pendekatan ini diantaranya :

   a. Tidak mudah untuk membuat pengetahuan informasi dan menyatakan pengetahuan tersebut ke dalam kondisi formal yang diperlukan oleh notasi logika, khususnya ketika pengetahuan tersebut memiliki kepastian kurang dari 100%.
   b. Terdapat perbedaan besar antara dapat memecahkan masalah “dalam prinsip” dan memecahkannya “dalam dunia maya”.

4. *Acting Rationally : The Rational Agen Approach*

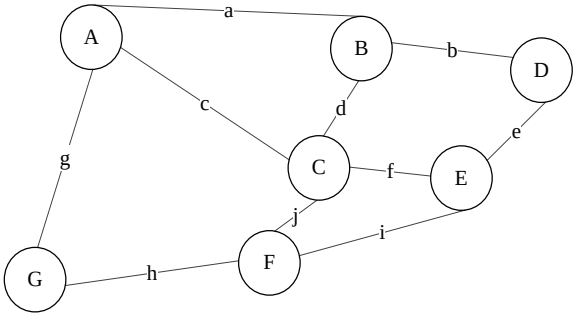
Membuat tindakan berdasarkan kesimpulan yang diambilsecara logis merupakan bagian dari satu *rational agent*. Aksi rasional harus dilakukan dengan cara menalar secara logis, dengan menalar secara logis maka bisa didapatkan kesimpulan aksi apa yang harus dilakukan untuk mencapai tujuan, sehingga *agent* memiliki aksi yang tepat.

Thinking Humanly dan *acting humanly* adalah dua definisi dam arti yang sangat luas. Sampai saat ini, pemikiran manusia diluar rasio, yakni refleks dan intuitif (berhubungan dengan perasaan), belum dapat ditirukan sepenuhnya oleh komputer. Dengan demikian, kedua definisi ini dirasa kurang tepat untuk saat ini. Jika kita menggunakan definisi ini makabanyak produk komputasi cerdas saat ini tidak layak disebut sebagai produk AI.

Definisi *thingking rationally* terasa lebih sempit dari pada *acting rationally* dengan pendekatan *rational agent*. Didasari dengan pemikiaran komputer yang mampu melakukan penalaran secara logis dan juga bisa melakukan aksi secara rasional dengan pengambilan keputusan dengan menyimpulkan penalaran logis yang dilakukan komputer.

2.5 Graph

Graph adalah sebuah struktur data yang terdiri dari kumpulan simpul *(node)* dan garis (*arc*)[15]. Sebuah arus harus diawali dan diakhiri dengan sebuah simpul. Ilustrasi sebuah *graph* dapat dilihat pada gambar 2.8.



Gambar 2. 8 Contoh *Graph*

Karena sebuah garis diawali dan diakhiri sebuah simpul maka sebuah garis misalnya untuk garis a dinyatakan dengan a = [A,B] dalam hal ini diartikan bahwa garis a diawali dengan simpul A dan diakhri dengan simpul B sedangkan simpul yang terhubung dengan sebuah garis disebut simpul tetangga sehingga dapat dikatakan simpul A bertetangga dengan simpul B.

Pada sebuah *graph* ada istilah yang disebut dengan jalur (*Path).* Jalur adalah kumpulan garis dan simpul yang saling berhubungan satu sama lainnya dengan simpul awal dan simpul akhir yang ditentukan, misalnya Z adalah sebuah jalur dari simpul A ke simpul D, maka dapat dinyatakan bahwa Z=(A,B,D). Sebuah jalur yang memiliki simpul akhir samadengan simpul awal disebut jalur tertutup *(Close Path)* sedangkan jalur yang terbentuk dari simpul-simpul yang berbeda disebut dengan jalursederhana (*Simple Path)* seperti halnya jalur dari simpul A ke simpul D.

2.5.1 Implementasi Graph

Sebuah *graph* dapat diimplentasikan dengan representasi statis dengan menggunkan *array.* Ada beberapa cara mengimplentasikan sebuah *graph* dengan menggunakan sebuah *array* [15]. Namun pada intinya dengan menggunakan sebuah mari. Perilaku matriks sebagai representasi *graph* akan dibahas pada subbab-subbab berikut.

2.5.1.1 Matrik Tetangga

Matrik tetangga merupakan salah satu cara untuk merepresentasikan *graph* berarah dengan meenggunkan *array* [15]. Berikut adalah representasi *graph* pada gambar 2.9 dengan menggunkan matrik tetangga berikut :

Simpul Tujuan

Simpul awal	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	0	1	1	0
C	1	0	0	0	0	0
D	1	0	1	0	0	0
E	0	0	0	1	0	1
F	0	0	0	1	0	0

Gambar 2. 9 Matrik Tetangga

Angka 1 (satu) menyatakan bahwa simpul awal memiliki jalur ke simpul tujuan, dan angka 0 (nol) menyatakan bahwa simpul awal tidak memiliki jalur ke simpul tujuan. Dari representasi pada gambar 2.9 dapat dilihat bahwa simpul-simpul yang memiliki jalur sebagai berikut.

a. simpul A memiliki jalur dan bertetangga dengan simpul B
b. simpul B memiliki jalur dan bertetangga dengan simpul D dan E
c. simpul C memiliki jalur dan bertetangga dengan simpul A
d. simpul D memiliki jalur dan bertetangga dengan simpul A dan C
e. simpul E memiliki jalur dan bertetangga dengan simpul D dan F
f. simpul F memiliki jalur dan bertetangga dengan simpul D

Implemetasi dengan bahasa algoritmik sebagi berikut :

```
Array [1...6, 1...6] of integer matGraf = {

{0, 1, 0, 0, 0, 0},

{0, 0, 0, 1, 1, 0},
```

```
{1, 0, 0, 0, 0, 0},

{1, 0, 1, 0, 0, 0},

{0, 0, 0, 1, 0, 1},

{0, 0, 0, 1, 0, 0}

}
```

Untuk mengakses simpul, dapat digunkan indeks *array*, misalnya simpul A ke simpul B maka yang diakses adalah mat$Graph_{2,1}$.

2.5.1.2 Matrik Jalur

Matriks jalur merupakan salah satu cara untuk merepresentasikan *graph* berarah menggunakan *array* [15]. Berikut adalah representasi *graph pada gambar* 2.10. dengan menggunakan matriks jalur berikut :

Simpul Tujuan

Simpul awal	A	B	C	D	E	F
A	1	1	1	1	1	1
B	1	1	1	1	1	1
C	1	1	1	1	1	1
D	1	1	1	1	1	1
E	1	1	1	1	1	1
F	1	1	1	1	1	1

Gambar 2. 10 Matrik Jalur

Angka 1 (satu) menyatakan bahwa simpul awal memiliki jalur ke simpul tujuan, dan 0 (nol) meyatakan bahwa simpul awal tidak memiliki jalur ke simpul tujuan, bedanya dengan matrik tetangga, angka 1 menyatakan jika simpul awal bertetangga secara langsung dengan simpul tujuan, sedangkan pada matriks jalur, simpul awal tidak harus bertetangga secara langsung dengan simpul tujuan, asalkan ada jalur yang di simpul awal ke simpul tujuan melewati simpul manapun maka dianggap bernilai 1. Dari representasi pada gambar 2.10 dapat dilihat bahwa semua

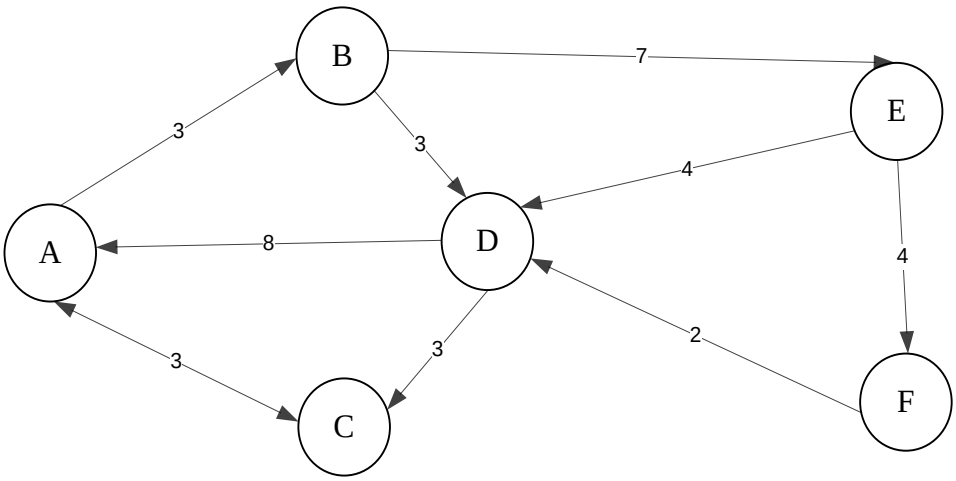
simpul dapat ditempuh dari simpul manapun. Implementasi dengan bahasa algoritma sebagai berikut :

```
Array [1...6, 1...6] of integer matGraf = {
{1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1},
{1, 1, 1, 1, 1, 1}
}
```

Untuk mengakses simpul, dapat digunkan indeks array, misalnya simpul A ke simpul B maka yang diakses adalah mat$Graph_{2,1}$.

2.5.1.3 Matrik Beban

Matrik beban merupakan salah satu cara untuk merepresentasikan *graph* berarah yang memiliki beban dengan menggunakan *array.* Matriks beban biasanya digunakan untuk merepresentasikan *graph* berarah pada jalur yang dilewatinya ada beban atau jarak tempuh dengan nilai tertentu. *Graph* berarah yang memiliki beban biasanya digunakan untuk menentukan letak kota-kota beserta jarak tempuh [15]. Sebagai contoh *graph* berarah yang memiliki beban dapat dilihat pada gambar 2.11.



Gambar 2. 11 Contoh *Graph* Berarah dengan Beban

Berikut adalah representasi *graph* pada gambar 2.11 dengan menggunkan matriks beban seperti pada gambar 2.12.

Simpul Tujuan

Simpul awal	A	B	C	D	E	F
A	0	3	0	0	0	0
B	0	0	0	3	7	0
C	3	0	0	0	0	0
D	8	0	3	0	0	0
E	0	0	0	4	0	4
F	0	0	0	2	0	0

Gambar 2. 12 Matrik Beban

Angka bukan nol menyatakan bahwa simpul awal memiliki jalur ke simpul tujuan dengan beban yang dituliskan pada matriks, dan angka 0 (nol) menyatakan bahwa simpul awal tidak memiliki jalur ke simpul tujuan. Pada dasarnya matriks beban sama dengan matrik tetangga hanya berbeda penanda adanya jalur, angka 1 (satu) pada matriks tetangga dan nilai beban pada matriks beban. Implementasi dengan bahasa algoritmik sebagai berikut :

```
Array [1...6, 1...6] of integer matGraf = {
{0, 3, 0, 0, 0, 0},
{0, 0, 0, 3, 7, 0},
{3, 0, 0, 0, 0, 0},
{8, 0, 3, 0, 0, 0},
{0, 0, 0, 4, 0, 4},
{0, 0, 0, 2, 0, 0}
}
```

Untuk mengakses simpul, dapat digunkan indeks array, misalnya simpul A ke simpul B maka yang diakses adalah mat$Graph_{2,1}$.

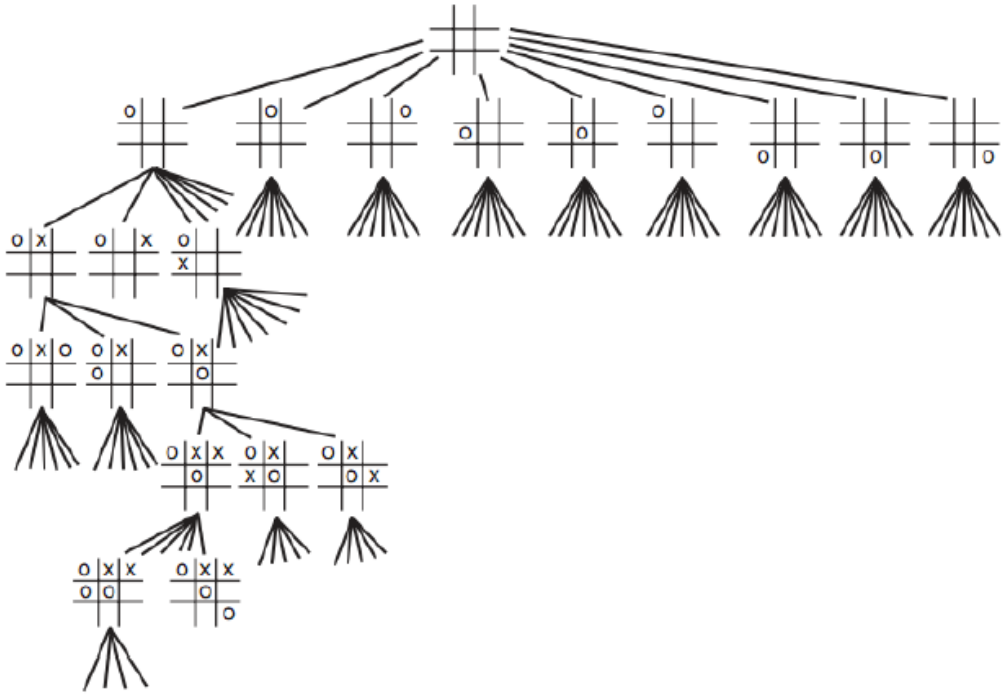
2.6 Pohon Permainan

Pohon permainan mempresentasikan kepada pemain kondisi – kondisi yang mungkin dihadapi pada permainan dimulai dari kondisi yang sedang dihadapi sekarang hingga beberapa kondisi ke depan. Sebuah pohon permainan merupakan representasi grafis dari contoh permainan. Pohon permainan menyediakan informasi pemain, hasil, strategi, dan pilihan langkah .

Pohon permainan dapat direpresentasikan dengan baik pada permainan yang dilakukan secara giliran (*turn-based-game*). Pohon permaian memiliki *root* yang merupakan representasi dari kondisi langkah yang belum diambil, *node* pada pohon mempresentasikan keadaan-keadaan yang mungkin diambil pada permainan, dan *Arcs* yang mempresentasikan langkah.

Pengguna pohon permainan pada permainan yang dimainkan oleh dua pemain direpresentasikan dengan cara bergantian. Untuk *edges* dari tingkat pertama keringat kedua mempresentasikan langkah – langkah yang dapat diambil oleh pemain pertama, sedangkan untuk *edges* dari tingkat kedua ketingkat ketiga mempresentasikan langkah-langkah yang diambil oleh pemain kedua, dan begitu seterusnya.

Leaf node pada pohon permainan mempresentasikan keadaan akhir pada permainan, Diana pemain tersebut dimenangkan, dikalahkan ataupun seri. Pada permainan sederhana, untuk mencapai *leaf node* mungkin dapat direpresentasikan, tetapi untuk permainan yang rumit seperti Damdaman, pencapaian *leaf node* sangat tidak memungkinkan karena percabangan pada pohon permainan yang sangat besar.



Gambar 2. 13 Contoh Pohon Permainan Tic-Tac-Toe [6]

Berikut penjelasan pohon permainan *tictactoe* pada gambar 2.13 :

1. Terdapat *root* yang merupakan keadaan awal Dimana pemain belum mulai bermain dan langkah belum diambil.
2. *Edges* yang menghubungkan tingkat pertama (*root)* dengan tingkat kedua merupakan langkah pemain pertama dan begitu seterusnya. Sehingga pohon permainan tersebut merepresentasikan langkah kedua pemain secara bergantian.
3. Untuk *nodes* pada pohon tersebut merepresentasikan keadaan-keadaan yang dapat diambil oleh pemain yang akan melangkah.

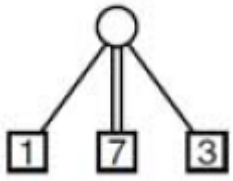
4. Percabangan pertama yang dihasilkan adalah 9, kemudian untuk percabangan berikutnya adalah 8, dan begitu seterusnya hingga mencapai keadaan akhir (*leaf nodes)*.

2.7 Algoritma Minimax

Pada permainan yang tidak terlalu kompleks, terdapat algoritma yang yangdigunakan untuk mencari langkah terbaik dari situasi sekarang. Algoritma yang paling terkenal ialah algoritma Minimax. Algoritma Minimax ialah satu metode yang sangat bermanfaat untuk tipe permainan bergiliran dengan 2 pemain. Metode ini memilih langkah terbaik yang diberikan, Dimana kedua pemain berusaha mengatur sebuah strategi untuk memenangkan permainan. Sehingga semua langkah selanjutnya akan dapat diketahui [10].

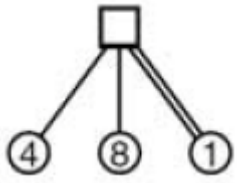
Minimax merupakan algoritma yang digunakan untuk menentukan pilihan agar kemungkinan kehilangan nilai maksimal dapat diminalisir. Algoritma ini dapat diterapkan dengan baik pada permainan yang melibatkan dua pemain yangsaling bergantian seperti atid-tac-toe, chekers, catur, dan permainan yang menggunakan strategi atau logika lainnya [11]. Permainan tersebut memiliki karakteristik yang sama yakni “permainan dengan informasi lengkap”. Setiap pemain mengetahui kemungkinan langkah yang akan dilakukan lawan. Sehingga pemain bisa tetap “memantau” kondisi permainan sewaktu permainan sedang berlangsung.

Jika pemain memilih satu langkah, maka pemain akan memilih yang terbaik. Sehingga secara nalar pemain akan mentukan posisi yang terbaik untuk melangkah. Dengan kata lain , ketika pemain mendapat giliran bermain maka pemain akan mencoba memaksimalkan nilainya, perhatikan gambar 2.14.



Gambar 2. 14 Langkah Pemain pada Algoritma Minimax [6]

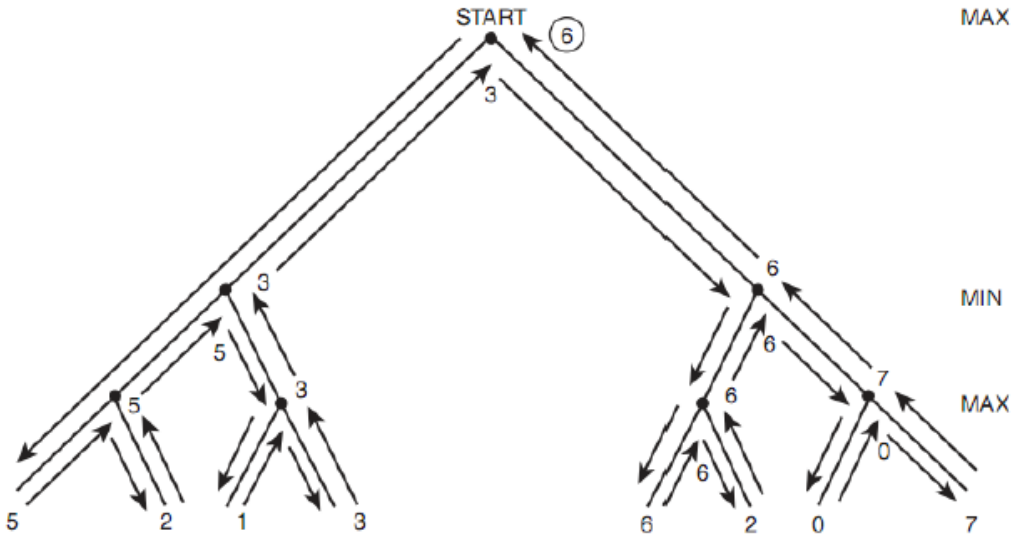
Ketika lawan mendapat giliran bermain, pemain akan mengganggap bahwa lawan akan memilih langkah yang akan memberikan pemain posisi terburuk yang tersedia. Lawan sedang berusaha meminimalisir nilai dari pemain, perhatikan gambar 2.15.



Gambar 2. 15 Langkah Lawan pada Algoritma Minimax [6]

Algoritma Minimax adalah salah satu implementasi dari pencarian DFS (*Dept-First Search)*, sehingga dalam melakukan pencarian pada pohon Minimax mirip dengan apa yang dilakukan dengan DFS. DFS akan melakukan penelusuran simpul paling dalam terlebih dahulu, setelah simpul akar dibangkitkan, algoritma ini akan membangkitkan simpul pada tingkat kedua, yang akandilanjutkan pada tingkat ketiga, dst.

Dalam representasi pohon pada algoritma Minimax, terdapat dua jenis *node*, yaitu *node min* dan *node Max.* Mak akan memilih langkah dengan nilai tertinggi dan min akan memilih langkah dengan nilai tererendah. Untuk proses dan cara kerja algoritma yang lebih jelasnya lagi, dapat dilihat pada gambar 2.16 yang merepresentasikan cara kerja algoritma Minimax.



Gambar 2. 16 Illustrasi Cara Kerja Algoritma Minimax [6]

Dari gambar 2.11, proses pencarian dimulai dari jalur paling kiri terlebih dahulu, sehingga DFS akan menelusuri simpul paling kiri bawah yaitu 5. Nilai 5 disimpan sebagai nilai maksimum sementara karena berada ditingkat oma, kemudian DFS melakukan *backtrak* dan menelusuri simpul berteangga dengan simpul 5 yaitu 2. Karena nilai 5 lebih besar dari nilai 2, maka nilai 2 tidak disimpan. Lalu DFS akan melakukan *backtrack* keringat min sehingga nilai 5 yang diperoleh akan disimpan sebagai nilai minimum sementara. Untuk simpul 1 dan 3, nilai 3 akan disimpan di *node* min karena merupakan nilai maksimum di tingkat max. Saat mencapai tingkat min, sudah ada nilai minimum sementara yaitu 5,namun karena nilai 3 lebih kecil dari nilai 5, maka nilai 5 akan digantikan dengan nilai 3. Nilai 3 akan disimpan sebagia nilai maksimum sementara di tingkat paling atas karena merupakan tingkat max. Lalu penelusuranjalur kanan akan dilakukan dengan cara yang sama dengan jalur kiri, sehingga diperoleh nilai 6. Karena nilai maksimum menara pada tingkat paling atas adalah 3, maka nilai 3 akan digantikan dengan nilai 6 karena nilai 6 lebih besar dari pada 3.dengan demikian, jalur yang akan dipilih menggunakan algoritma Minimax adalah jalur belah kanan karena untuk kondisi terburuknya, pemain akan mendpat nilai 6 sedangkan jika pemain memilih jalur kiri, pemain hanya akan mendapat nilai 3.

Pada dasarnya, algoritma Minimax sangat handal untuk menyelesaikan segala masalah dalam pencarian langkah untuk permainan komputer dengan jumlah kemungkinan penyelesaian yang kecil. Tetapi jika, algoritma Minimax digunakan untuk permainan dengan jumlah kemungkinan penyelesaian yang besar seperti permainan Damdaman, algoritma Minimax ini memelukan waktu lama untuk membangun pohon penyelesaian.

Oleh karena itu, beberapa metode lanjutan dari algoritma Minimax telah dikembangkan untuk membatasi melonjaknya jumlah simpul dalam pembangunan pohon penyelesaian. Berbagai jenis metode telah ditemukan untuk meningkatkan kinerja algoritma Minimax, salah satunya adalah Negascout. Dengan menggunakan metode ini maka diharapkan sistem dapat bekerja lebih baik. *Pseudocode* Minimax Dapat dilihat pada gambar 2.17.

```
1    function minimax(node,depth)
2    if depth <= 0 then
3    {positive values are good for the maximizing player}
4    {negative values are good for the minimizing player}
5     return objective_value(node)
6     end
7    {maximizing player is (+1)}
8    {minimizing player is (-1)}
9     local alpha = -node.player * INFINITY
10    local child = next_child(node,nil)
11    while child ~= nil do
12    local score = minimax(child,depth-1)
13    alpha = node.player==1 and math.max(alpha,score) or
     math.min(alpha,score)
14    child = next_child(node,child)
15    end
16    return alpha
17   end
```

Gambar 2. 17 *Pseudocode* Minimax

2.8 Negascout

Negascout adalah salah satu metode pencarian Minimax dengan berasumsi bahwa langkah pertama yang diambil merupakan langkah terbaik, sedangkan sisanya merupakan langkah terburuk. Namun jika ternyata ada langkah yang lebih baik dari langkah pertama, maka akan terjadi proses *Research* atau proses pencarian ulang [12].

Dalam memperkecil jendela pencarian agar mendapat jendela pencarian dengan lebar nol. Negascout akan memotong hampir semua cabang-cabang dari pohon, untuk membuat suatu pencarian yang sangat tepat. Tetapi sayangnya, ia akan memotong semua cabang-cabang berguna bersamaan dengan yang tidak berguna. Hal ini tidak berlaku jika algoritma dimulai dengan hasil yang benar. Satu jendela nol dapat dilihat sebagai uji coba. Uji coba dilakukan jika nilai sebenarnya sama dengan nilai yang ditebak [6].

Algoritma negascout cukup efisien, tetapi melakukan pemeriksaan posisi papan lebih dari yang diperlukan. Pemotongan cabang memungkinkan algoritma untuk mengabaikan bagian dari pohon yang tidak mungkin berisi langkah terbaik. Ini menghasilkan dua jenis pemotongan yaitu pemotongan alpha dan pemotongan beta. pada gambar 2.18 dapat dilihat *pseudocode* Negascout.

```
1   FUNCTION negascout (p: POSITION; alpha, beta, depth:
    INTEGER) : INTEGER;
2   VAR i,t,m,n: INTEGER;
3   BEGIN
4   IF depth = d THEN RETURN (evaluate(p))
5   ELSE
6   BEGIN m := -INFINITY;
7   n := beta;
8   FOR i := 1 TO b DO
9   BEGIN t:= -negascout (p.i, -n, -max(alpha,m), depth+1);
1   IF t > m THEN
0
1   IF (n = beta) OR (depth >= d-2)
1
1   THEN m := t
2
1   ELSE m := -negascout (p.i, -beta, -t, depth+1);
3
1   IF m >= beta THEN RETURN (m);
4
```

```
1   n := max (alpha, m) +1;
5
1   END;
6
1   RETURN (m);
7
1   END;
8
1   END;
9
```

Gambar 2. 18 *pseudocode* Negascout

Keterangan *pseudo code*:

p = kondisi papan

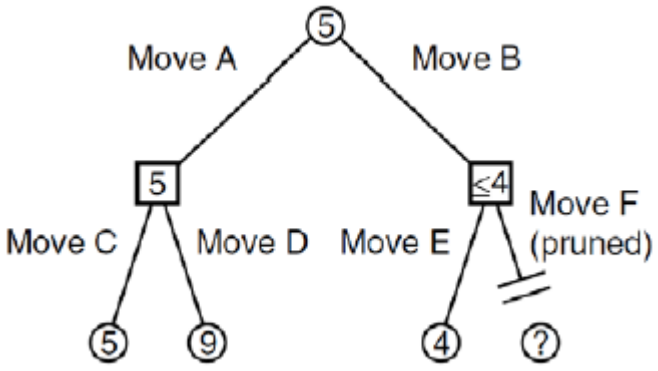
d = kedalaman maksimum dari pohon yang ditelusuri

b = lebar pohon (banyak langkah yang mungkin terjadi pada papan)

evaluate(p) = merupakan fungsi yang memberikan nilai evaluasi pada kondisi papan "p".

2.8.1 Pemotongan Alpha

Gambar 2.19 menunjukan satu pohon permainan sebelum proses pemotongan dilakukan. Untuk lebih mudah melihat bagaimana nilai nilai di proses, akan dicoba dlakukan dengan algoritma Minimax untuk ilustrasi ini.



Gambar 2. 19 Pohon Permainan dengan Proses Pemotongan Alpha [6]

Untuk proses pertama, sama halnya dengan algoritma Minimax. Jika pemain memilih langkah A, maka lawan akan membirakan respons dengan langkah C, dan memberikan pemain nilai 5. Maka nilai 5 naik pada tingkat min. Jika pemain memilih langkah B maka lawan akan melangkah ke E, sehingga pemain akan

mendapat nilai 4. Setelah melihat kondisi demikian pemain akan menyadari nilai yang didapatkan lebih rendah dari nilai sementara di tingkat max. Sehingga pemain akan menyimpulkan ini adalah langkah yang salah.

Untuk melakukan pemotongan dengan cara ini, perlu dilakukan pencarian nilai terbaik yang pasti akan tercapai. Bahkan, nilai ini membentuk batas bawah pada nilai yang akan tercapai. Sehingga urutan langkah yang lebih baik dapat ditemukan dalam pencarian, namun nilai yang paling rendah akan mustahil untuk didapatkan. Batas bawah ini dinamakan nilai alpa dan pemotongannya dinamakan dengan pemotongan alpa.

2.8.2 Pemotongan Beta

Pemotongan beta bekerja dengan cara yang sama. Nilai beta terus melacak nilai batas tertinggi dari nilai yang diharapkan. Nilai beta berubah ketika ditemukan satu langkah yang dipaksakan oleh lawan.

Pada posisi itu pemain mengetahui tidak ada nilai yang lebih besar dari beta. Jika urutan langkah yang bernilai lebih besar dari nilai beta ditemukan maka dari kondisi ini dapat diabaikan, karena kesempatan untuk mendapatkannya lebih kecil.

2.8.3 Pemotongan Alpha Beta

Bersamaan nilai – nilai dari alpa dan beta memberikan satu jendela kemungkinan langkah. Pemain etika pernah memilih untuk melakukan langkah yang dinilainya lebih kecil dari alpaha, dan lawan tidak akan pernah membiarkan ada langkah yang lebih besar dari beta. Nilai pada akhirnya harus terletak antara alpa dan beta. Ketika pencarian mulai dilakukan pada pohon, nilai - nilai dari alpha dan beta diperbaharui. Jika cabang dari pohon yang ditemukan di luar nilai – nilai ini, maka akan dilakukan pemotongan pada cabang.

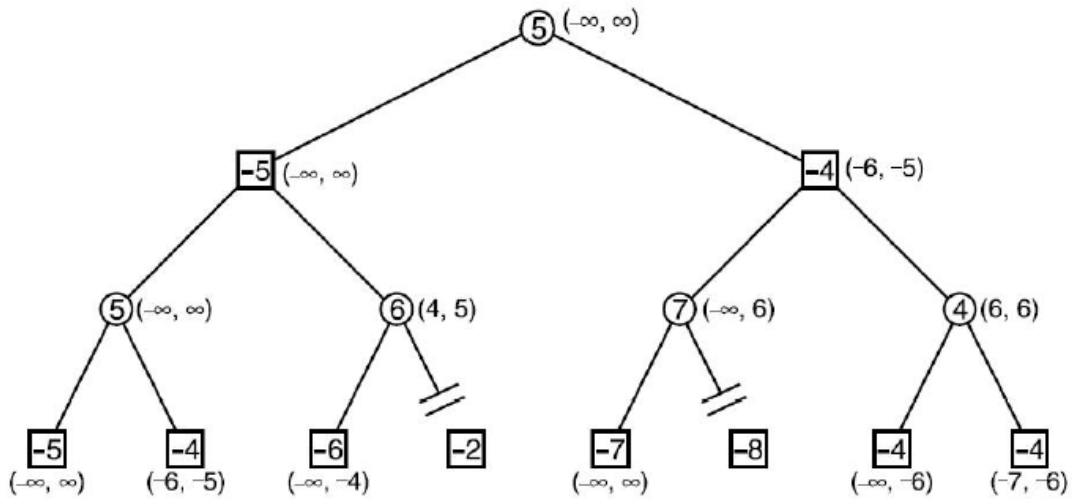
2.8.4 Zero-Width Test

Pada algoritma negascout ada yang disebut dengan kondisi *failsoft*, kondisi ini menggambarkan pemotongan alpa dan beta yang telah diketahui. Pada dasarnya pemotongan alpa beta hanya mengembalian baik nila alpa maupun nilai beta

sebagai nilai jika gagal berdasarkan gagal karena nilai yang didapatkan lebih tinggi atau lebih rendah dari nilai maksimum sementara pada tingkat pencarian tertentu. Dengan kondisi *fail soft* dapat membantu untuk menemukan satu solusi. Dengan kondisi ini memungkinkan untuk menebak nilai awal dan megulangi pencarian dengan jendela yang lebih masuk akal (*Zero width test*).

Pada dasarnya algoritma negascout menggabungkan pencarian Minimax dengan pemotongan alpa beta dan pemanggilan *zero-width Test.* Negascout tidak selebar pencarian algoritma Minimax. Negascout menggunakan pemotongan alpa dan beta untuk uji coba tersebut.

Negascout bekerja dengan melakukan pemerikasaan secara menyeluruh pada langkah pertama dalam setiap posisi papan. Hal ini dilakukan pada satu jendela pencarian luas, sehingga algoritma tidak gagal. Berturut-turut langkah diuji menggunakan satu jalur berdasarkan nilai dari langkah pertama, jika berhasil, maka akan diulangi pada jendela dengan lebar sebenarnya, perhatikan gambar 2.20.



Gambar 2. 20 Pohon Permainan dengan Algoritma Negascout [6].

2.9 Diagram Konteks

Diagram konteks merupakan alat pemodelan atau suatu diagram yang menggambarkan sistem berbasis komputer yang dirancang secara global dan merupakan suatu diagram alir data tingkat atas, dimana didalam diagram konteks

ini menggambarkan seluruh jaringan, baik masukan maupun sebuah keluaran sebuah sistem [8].

Diagram konteks terdiri dari sebuah simbol proses tunggal yang menggambarkan sebuah sistem dan menunjukan data aliran utama untuk dan dari *terminator*. Diagram ini merupakan dasar yang digunakan untuk menentukan aliran data yang mengalir menuju sistem (*input system*) dan keluar dari sistem (*output system*), yang meliputi objek berupa kesatuan luar (*eksternal entity*). Diagram konteks dapat mendefinisikan jangkauan proses penurunan sistem informasi yaitu menentukan apa yang menjadi bagian dari sistem informasi dan apa yang tidak menjadi bagian sistem informasi.

2.10 Data Flow Diagram (DFD)

Pengertian *Data Flow Diagram* (DFD) menurut Jogiyanto Hartono adalah: "Diagram yang menggunakan notasi simbol untuk menggambarkan arus data system" [8].

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem yang baru yang akan dikembangkan secara logika dan menjelaskan arus data dari mulai pemasukan sampai dengan keluaran data tingkatan diagram arus data mulai dari diagram konteks yang menjelaskan secara umum suatu *system* atau batasan *system* dari *level* 0 dikembangkan menjadi *level* 1 sampai system tergambarkan secara rinci. Gambaran ini tidak tergantung pada perangkat keras, perangkat lunak, struktur data atau organisasi file.

Menurut Jogiyanto Hartono, tahun 2005 dalam bukunya Basia Data ada beberapa simbol digunakan pada DFD untuk mewakili :

a) Kesatuan Luar (*External Entity*)

Kesatuan luar (external entity) merupakan kesatuan (*entity*) di lingkungan luar sistem yang dapat berupa orang, organisasi, atau sistem lain yang berada pada lingkungan luarnya yang memberikan *input* atau menerima *output* dari sistem.

b) Arus Data (*Data Flow*)

Arus Data (*data flow*) di DFD diberi simbol suatu panah. Arus data ini mengalir di antara proses, simpan data dan kesatuan luar. Arus data ini menunjukan arus dari data yang dapat berupa masukan untuk sistem atau hasil dari proses sistem.

c) Proses (*Process*)

Proses (*process*) menunjukan pada bagian yang mengubah input menjadi *output*, yaitu menunjukan bagaimana satu atau lebih *input* diubah menjadi beberapa *output.* Setiap proses mempunyai nama, nama dari proses ini menunjukan apa yang dikerjakan proses.

d) Simpanan Data (*Data Store*)

Data Store merupakan simpanan dari data yang dapat berupa suatu *file* atau *database* pada sistem komputer.

2.11 Pengujian

Pengujian perangkat lunak (bahasa Inggris: *software testing*) merupakan suatu investigasi yang dilakukan untuk mendapatkan informasi mengenai kualitas dari produk atau layanan yang sedang diuji (*under test*) [16]. Pengujian perangkat lunak juga memberikan pandangan mengenai perangkat lunak secara obyektif dan independen, yang bermanfaat dalam operasional bisnis untuk memahami tingkat risiko pada implementasinya. Teknik-teknik pengujian mencakup, namun tidak terbatas pada, proses mengeksekusi suatu bagian program atau keseluruhan aplikasi dengan tujuan untuk menemukan *bug* perangkat lunak (kesalahan atau cacat lainnya).

Pengujian perangkat lunak dapat dinyatakan sebagai proses *validasi* dan *verifikasi* bahwa sebuah program :

1. Memenuhi kebutuhan (*requirement*) yang mendasari perancangan dan pengembangan perangkat lunak tersebut;
2. Berjalan sesuai dengan yang diharapkan;

3. Dapat diterapkan menggunakan karakteristik yang sama;
4. Memenuhi kebutuhan semua pihak yang berkepentingan.

2.11.1 *Blackbox Testing*

Test case ini bertujuan untuk menunjukkan fungsi Perangkat Lunak tentang cara beroperasinya, apakah pemasukan data keluaran telah berjalan sebagaimana yang diharapkan dan apakah informasi yang disimpan secara eksternal selalu dijaga kemutakhirannya.

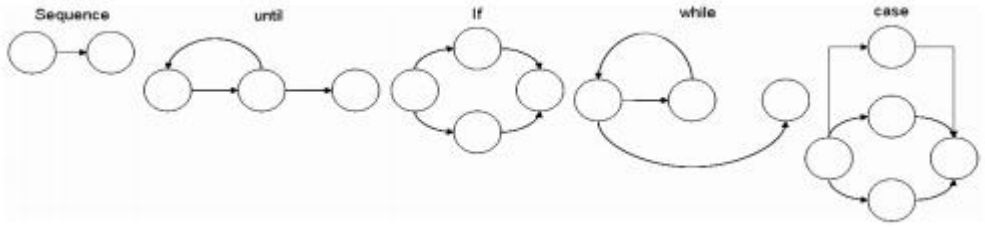
2.11.2 *Whitebox Testing*

White box Testing atau pengujian *glass box* adalah metode desain *test case* menggunakan struktur kontrol desain prosedural untuk mendapatkan *test case*. Dengan menggunakan metode *White box* analisis sistem akan memperoleh *Test case* sebagai berikut:

1. Menjamin seluruh independent *path* di dalam modul yang dikerjakan sekurang-kurangnya sekali
2. Mengerjakan seluruh keputusan logika
3. Mengerjakan seluruh loop yang sesuai dengan batasannya
4. Mengerjakan seluruh struktur data internal yang menjamin validitas.

1. Uji Coba Basis *Path*

Merupakan teknik uji coba *white box* yang diusulkan Tom McCabe. Metode ini memungkinkan perancang *test case* mendapatkan ukuran kekompleksan logical dari perancangan prosedural dan menggunakan ukuran ini sebagai petunjuk untuk mendefinisikan basis set dari jalur pengerjaan. *Test case* yang didapat digunakan untuk mengerjakan basis set yang menjamin pengerjaan setiap perintah minimal satu kali selama uji coba.



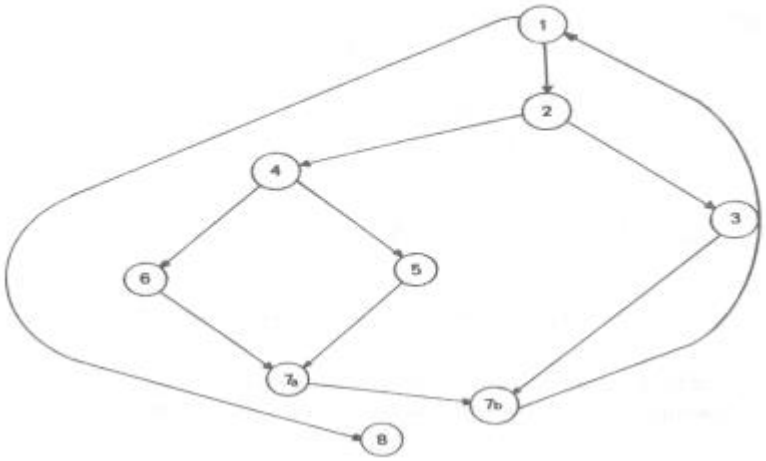
Gambar 2. 21 Notasi Diagram Alir

Lingkaran (*node*), menggambarkan satu/lebih perintah prosedural. Urutan proses dan keputusan dapat dipetakan dalam satu *node*. Tanda panah (*edge*), menggambarkan aliran kontrol. Setiap *node* harus mempunyai tujuan *node*.

Region adalah daerah yang dibatasi oleh *edge* dan *node*. Termasuk daerah diluar grafik alir. Contoh menterjemahkan *pseudocode* pada gambar 2.21 ke grafik alir akan ditunjukan pada gambar 2.22.

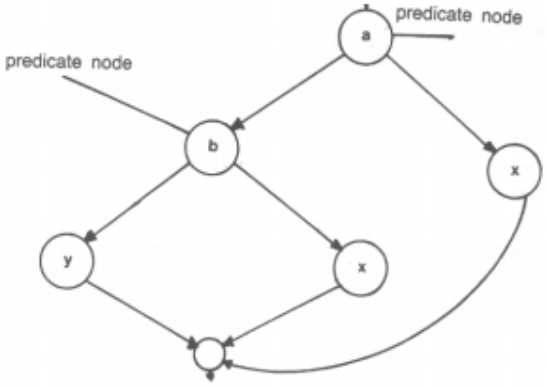
```
1   do while record masih ada baca record
2   if record ke 1 = 0
3   then proses record simpan di buffer
    naikan counter
4   else if record ke 2 = 0
5   then riser counter
6   proses record simpan pada file
7   endif
8   endif
9   end do
10  end
```

Gambar 2. 22 Contoh *pseudocode*



Gambar 2. 23 Diagram Alir

Nomor pada *pseudocode* berhubungan dengan nomor *node*. Apabila ditemukan kondisi majemuk (*compound condition*) pada *pseudocode* pembuatan grafik alir menjadi rumit. Kondisi majemuk mungkin terjadi pada operator Boolean (*AND, OR, NAND, NOR*) yang dipakai pada perintah *if*. Berikut contoh gambar akan ditunjukan pada gambar 2.24 berikut :



Gambar 2. 24 Logika gabungan

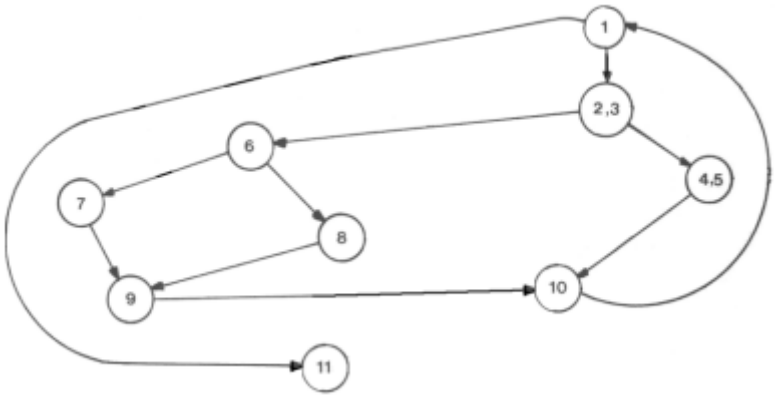
```
if A or B
then procedure x
else procedure y
endif
```

Node dibuat terpisah untuk masing-masing kondisi A dan B dari pernyataan *IF A OR B*. Masing-masing *node* berisi kondisi yang disebut *pridicate node* dan mempunyai karakteristik dua atau lebih *edge* darinya.

2. Cyclomatic complexity

Cyclomatic complexity adalah metrik software yang menyediakan ukuran kuantitatif dari kekompleksan logikal program. Apabila digunakan dalam konteks metode uji coba basis *path* , nilai yang dihitung untuk *cyclomatic complexity* menentukan jumlah jalur independen dalam basis set suatu program dan memberi batas atas untuk jumlah uji coba yang harus dikerjakan untuk menjamin bahwa seluruh perintah sekurang-kurangnya telah dikerjakan sekali.

Jalur independent adalah jalur yang melintasi atau melalui program dimana sekurang-kurangnya terdapat proses perintah yang baru atau kondisi yang baru.



Gambar 2. 25 Contoh *Flowgraph*

Path 1 = 1 - 11

Path 2 = 1 - 2 - 3 - 4 - 5 - 10 - 1 - 11

Path 3 = 1 - 2 - 3 - 6 - 8 - 9 ...: 10 - 1 - 11

Path 4 = 1 - 2 - 3 - 6 - 7 - 9 - 10 - 1 - 11

Path 1,2,3,4 yang telah didefinisikan diatas merupakan basis set untuk diagram alir.

Cyclomatic complexity digunakan untuk mencari jumlah *path* dalam satu *flowgraph*. Dapat dipergunakan rumusan sebagai berikut :

1. Jumlah region grafik alir sesuai dengan *cyclomatic complexity*.
2. *Cyclomatix complexity* V(G) untuk grafik alir dihitung dengan rumus:
   V(G) = E - N + 2
   Dimana:
   E = jumlah *edge* pada grafik alir
   N = jumlah *node* pada grafik alir

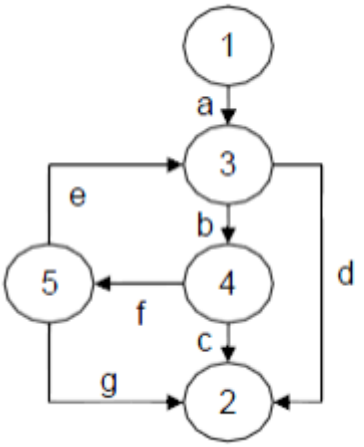
3. Cyclomatix complexity V(G) juga dapat dihitung dengan rumus:

   V(G) = P + 1
   Dimana P = jumlah predicate *node* pada grafik alir
   Pada gambar 2.5 dapat dihitung *cyclomatic complexity*:
   1. *Flowgraph* mempunyai 4 region
   2. V(G) = 11 *edge* - 9 *node* + 2 = 4
   3. V(G) = 3 predicate *node* + 1 = 4
   Jadi *cyclomatic complexity* untuk *flowgraph* adalah 4.

3. *Graph* Metrik

Graph metrik merupakan *software* yang dikembangkan untuk membantu uji coba basis *path* atau struktur data. *Graph* metrik adalah matrik empat persegi yang mempunyai ukuran yang sama dengan jumlah *node* pada *flowgraph*. Masingmasing baris dan kolom mempunyai hubungan dengan *node* yang telah ditentukan dan pemasukan data matrik berhubungan dengan hubungan (*edge*) antar

node. Berikut ini adalah contoh sederhana pemakaian *graph* metrik dapat digambarkan pada gambar 2.26.



	1	2	3	4	5
1			a		
2					
3				b	
4		c			f
5		g	e		

Gambar 2. 26 Penggunaan *Graph* Metrik

Hubungan bobot menyediakan tambahan informasi tentang aliran kontrol. Secara simpel hubungan bobot dapat diberi nilai 1 jika ada hubungan antara *node* atau nilai 0 jika tidak ada hubungan. Dapat juga hubungan bobot diberi tanda dengan:

a. Kemungkinan *link* (*edge*) dikerjakan
b. Waktu yang digunakan untuk proses selama *traversal* dari *link*
c. Memori yang diperlukan selama *traversal link*
d. Sumber daya yang diperlukan selama *traversal link*

2.11 *Tools* Yang Digunakan

Adapun perangkat lunak yang digunakan untuk Pembangunan *game* Damdaman yaitu adobe flash dan menggunakan bahasa pemograman menggunakan *ActionScript* 2.0. dan untuk versi adobe flash yang digunakan versi CS6.

2.11.1 Adobe Flash

Flash adalah salah satu perangkat lunak komputer yang digunakan untuk membuat gambar vektor maupun animasi gambar. Flash menjadi sebuah cara yang

popular untuk membuat animasi, iklan, multimedia interaktif, integrasi video pada halaman *Web*.

Sama halnya dengan aplikasi grafik lainnya Flash dapat memanipulasi *vektor* dan *bitmap,* namun Flash dapat diisi dengan pemprograman yang rumit dan dapat menghasilkan animasi yang ringan apabila dikerjakan dengan mode *vektor* yang sederhana. Keunggulan Flash lainnya adalah dapat mendukung *streaming audio* maupun *video,* hal yang tidak dapat dilakukan oleh aplikasi grafik lainnya.

Flash bekerja dengan bahasa pemprograman yang dinamakan *ActionScript*. Bahasa ini telah mengalami beberapa kali perkembangan sejak pertama kali diperkenalkan. Versi terakhir adalah *ActionScript* 3.0 atau disingkat AS3, dengan struktur bahasa yang sangat menyerupai javascript.

Dengan dukungan *ActionScript,* maka aplikasi yang dibangun dengan Flash dapat ditampilkan secara online di internet. Sebuah *game* yang ditulis dalam format Flash boleh menghasilkan file SWF berukuran beberapa *kilobyte* saja. SWF adalah kependekan dari "*ShockWave Flash*", kata lain yang biasanya dipergunakan untuk menyebutkan file swf adalah *Flash Movie*. *File Flash Movie* memiliki ekstensi .swf, yang hanya dapat ditampilkan apabila komputer yang menjalankan aplikasi tersebut telah memiliki *plug-in "Flash Player*".

2.11.2 ActionScript

ActionScript adalah bahasa pemrograman yang dibuat berdasarkan ECMAScript, yang digunakan dalam pengembangan situs *web* dan perangkat lunak menggunakan platform Adobe Flash Player. *ActionScript* juga dipakai pada beberapa aplikasi basis data, seperti *Alpha Five*. Bahasa ini awalnya dikembangkan oleh Macromedia, tapi kini sudah dimiliki dan dilanjutkan perkembangannya oleh Adobe, yang membeli Macromedia pada tahun 2005.

ActionScript terbaru saat ini adalah *ActionScript* 3.0. *ActionScript* 3.0 adalah bahasa terbaru dari edisi yang sebelumnya dikenal dengan *ActionScript* 2.0. *ActionScript* 3.0 memiliki beberapa kelebihan dibanding pendahulunya, antara lain fitur yang ditawarkan adalah *file* pada *ActionScript* 3.0 dapat dibuat terpisah saat *runtim*e.[7].