BAB III

ANALISIS DAN PERANCANGAN SISTEM

III.1 Analisis Sistem

Pada analisis sistem akan dibahas mengenai analisis *game* sejenis dan analis *game* pacman Arkage ini. Pada analisis *game* sejenis dilakukan observasi terhadap beberapa *game* pacman.

III.1.1 Analisis Game Pacman

Pacman adalah sebuah permainan arkade yang cukup terkenal. Cara bermainnya mudah yaitu pemain (pacman) diharuskan memakan makanan (berbentuk titik-titik kecil) dan sebuah bulatan besar (*energizer*) sampai habis di dalam sebuah labirin yang berliku-liku. Tidak hanya menghabiskan makanan tersebut, pemain juga harus menghindari 4 ‘hantu’ yang berkeliaran secara random untuk menangkap pemain. Jika pemain bertemu dengan hantu-hantu tersebut maka pemain dinyatakan gagal dan harus mengulangi dari awal lagi. Tetapi pemain bisa mengalahkan hantu tersebut dengan memakan *energizer* yang terdapat di pojokkan labirin. Jika pemain memakan titik besar tersebut, maka para hantu akan ketakutan dan berusaha menjauh dari pemain. Dalam hal ini pemain bisa memakan hantu tersebut dan mendapatkan bonus yang besar, tetapi para hantu yang termakan tidak mati begitu saja, mereka kembali ke posisi semula dan kembali mengejar pemain. Pemain dinyatakan menang jika semua makanan habis tak tersisa dan pemain akan memasuki *level* berikutnya.

Pergerakan para hantu ini dipengaruhi oleh kecerdasan buatan atau *Artificial intelligence* (AI), dimana para hantu diberi kecerdasan untuk menentukan langkah dan mengambil keputusan akan bergerak kemana dengan menentukan rute yang paling pendek (minimum), tujuannya adalah menangkap pemain. Setiap hantu harus memiliki pemikiran berbeda dan memiliki kemampuan bekerja sama untuk mengejar pemain, sehingga permainan akan tampak lebih menarik. Persoalan mendekati karakter Pacman ini dapat diselesaikan dengan berbagai macam cara, salah satunya dengan menggunakan algoritma *greedy.*

III.1.2 Analisis Penyelesaian Masalah Dengan Algoritma *Greedy*

Persoalan optimasi pada musuh *Pacman* ini termasuk persoalan minimasi, yaitu mencari rute terpendek saat ini dari posisi karakterk musuh ke posisi karakter Pacman.

Greedy diterjemahkan ke dalam bahasa Indonesia berarti : rakus, tamak, atau loba. Sesuai dengan namanya, prinsip dari algoritma *Greedy* adalah memilih keputusan yang terbaik pada setiap langkahnya (*optimum local*). Keputusan – keputusan local yang diambil pada akhirnya menjadi solusi optimum yang mutlak (*Optimum Global*).

Elemen-elemen dari algoritma *greedy* adalah sebagai berikut :

1. Himpunan Kandidat
2. Himpunan Solusi : himpunan bagian dari himpunan kandidat yang merupakan solusi atas permasalahan.

3. Fungsi Seleksi : fungsi yang memiliki unsur *greedy* di dalamnya, yang digunakan untuk menyeleksi himpunan kandidat.
4. Fungsi Layak : untuk memeriksa apakah solusi yang dipilih merupakan solusi yang layak atau tidak.
5. Fungsi Objektif : solusi yang dihasilkan optimum.

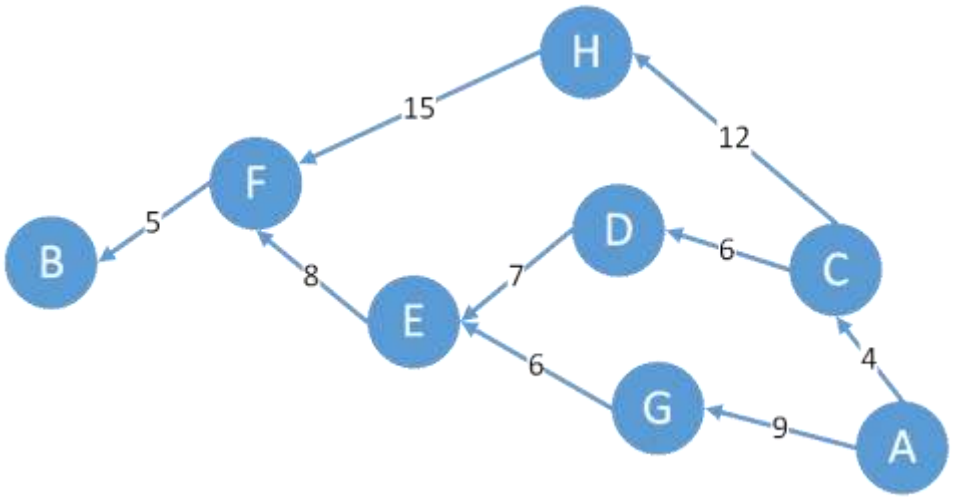
Namun demikian, kekurangan dari algoritma *greedy* adalah solusi akhir yang dibentuk oleh fungsi seleksi tidak selalu menghasilkan solusi global yang paling optimum. Hal ini dikarenakan algoritma *greedy* tidak memeriksa semua kemungkinan dan hanya mengambil yang terbaik relatif terhadap fungsi seleksi yang didefinisikan.

Meskipun demikian, algoritma *greedy* sangat cocok diterapkan untuk mendapatkan solusi yang mendekati optimum. Untuk dapat menghasilkan solusi yang semakin mendekati optimum, hal yang sangat menentukan adalah pemilihan fungsi seleksi, karena fungsi tersebut yang menentukan langkah mana yang diambil di tiap tahap.

Strategi atau pendekatan algoritma *greedy* pada suatu persoalan bukan hanya satu, namun bisa menjadi sangat banyak, dan masing-masing strategi bisa menghasilkan solusi yang berbeda-beda. Karena algoritma *greedy* tidak menjamin optimalitas solusi, maka pemilihan fungsi seleksi pada algoritma ini menjadi sangat penting.

III.2. Penerapan Metode

Pergerakan algoritma *greedy* diterapkan pada *ghost*. Pergerakan para *ghost* ini dipengaruhi oleh kecerdasan buatan atau *Artificial intelligence* (AI), dimana para hantu diberi kecerdasan untuk menentukan langkah dan mengambil keputusan akan bergerak kemana dengan menentukan rute yang paling pendek (minimum), tujuannya adalah menangkap pemain. Setiap hantu harus memiliki pemikiran berbeda dan memiliki kemampuan bekerja sama untuk mengejar pemain, sehingga permainan akan tampak lebih menarik, Berikut adalah contoh graph algoritma *greedy*:



Gambar III.1 Graph Berarah dari Titik A ke B

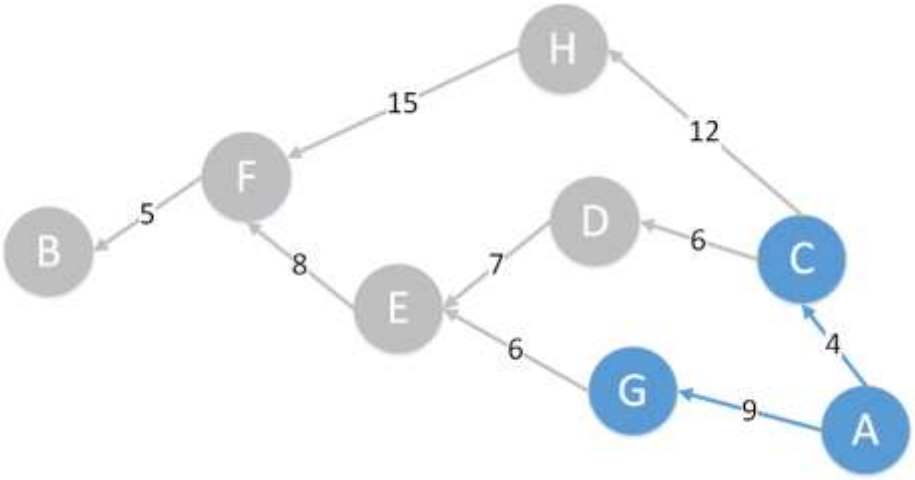
Untuk mencari jarak terpendek dari A ke B, sebuah algoritma *greedy* akan menjalankan langkah-langkah seperti berikut:

1. Kunjungi satu titik pada graph, dan ambil seluruh titik yang dapat dikunjungi dari titik sekarang.
2. Cari *local maximum* ke titik selanjutnya.

3. Tandai graph sekarang sebagai graph yang telah dikunjungi, dan pindah ke *local maximum* yang telah ditentukan.
4. Kembali ke langkah 1 sampai titik tujuan didapatkan.

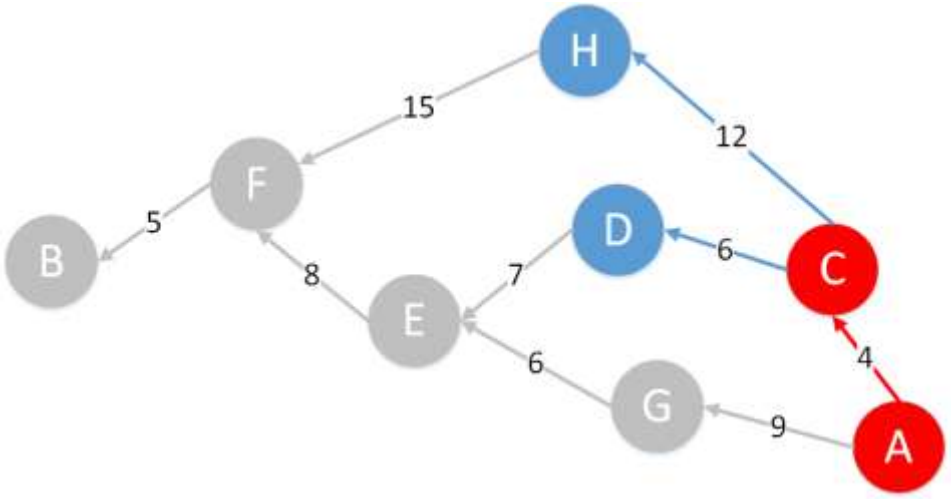
Jika mengapliikasikan langkah-langkah di atas pada graph A ke B sebelumnya maka kita akan mendapatkan pergerakan seperti berikut:

1. Mulai dari titik awal (A). Ambil seluruh titik yang dapat dikunjungi.



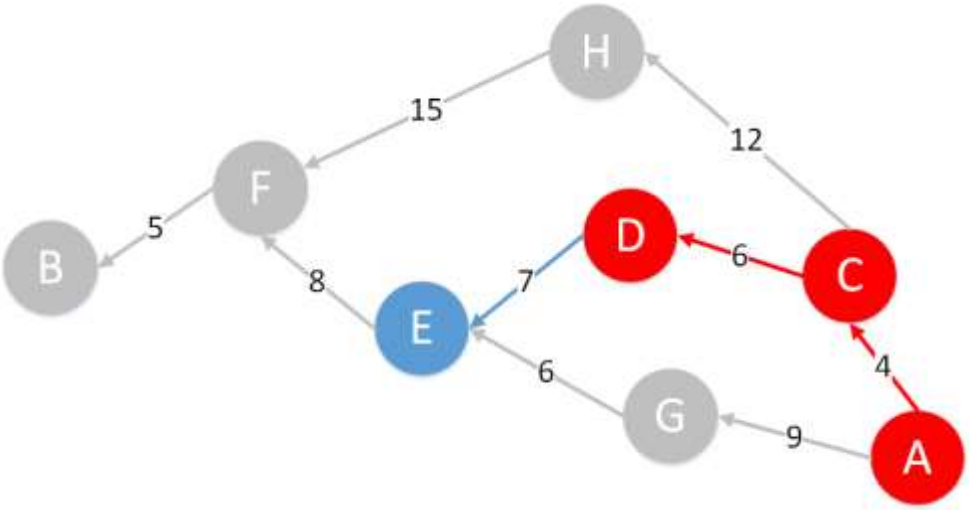
Gambar III.2 Graph Langkah Pertama *Greedy*

2. Local maximum adalah ke C, karena jarak ke C adalah yang paling dekat.
3. Tandai A sebagai titik yang telah dikunjungi, dan pindah ke C.
4. Ambil seluruh titik yang dapat dikunjungi dari C.



Gambar III.3 Graph Langkah Kedua *Greedy*

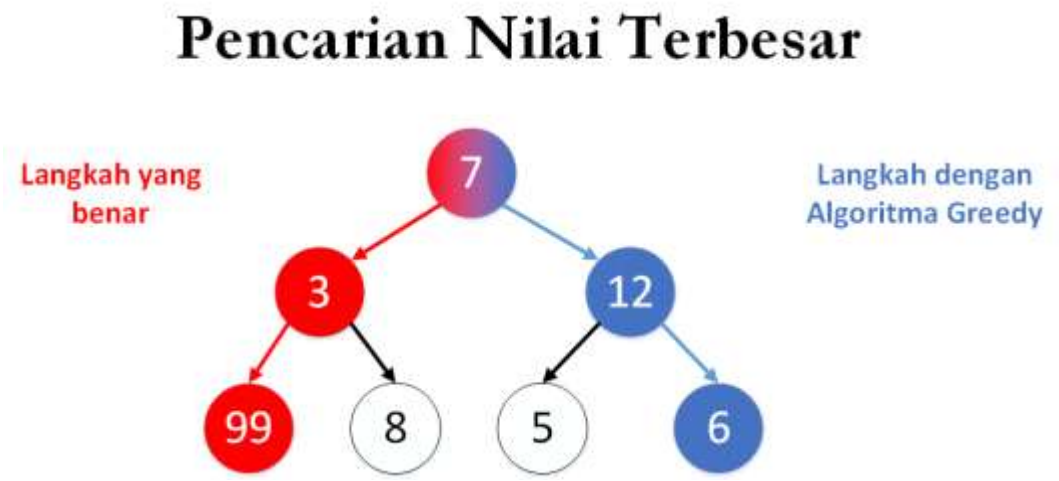
5. Local maximum adaah ke D, dengan jarak 6.
6. Tandai C sebagai titik yang telah dikunjungi, dan pindah ke D.



Gambar III.4 Graph Langkah Ketiga *Greedy*

Dengan menggunakan algoritma *greedy* pada graph di atas, hasil akhir yang akan didapatkan sebagai jarak terpendek adalah A-C-D-E-F-B. Hasi jarak terpendek yang didapatkan ini tidak tepat dengan jarak terpendek yang sebenarnya (A-G-E-F-B). Algoritma *greedy* memang tidak selamanya memberikan solusi yang optimal, dikarenakan pencarian *local maximum* pada setiap langkahnya,

tanpa memperhatikan solusi secara keseluruhan. Gambar berikut memperlihatkan bagaimana algoritma *greedy* dapat memberikan solusi yang kurang optimal:

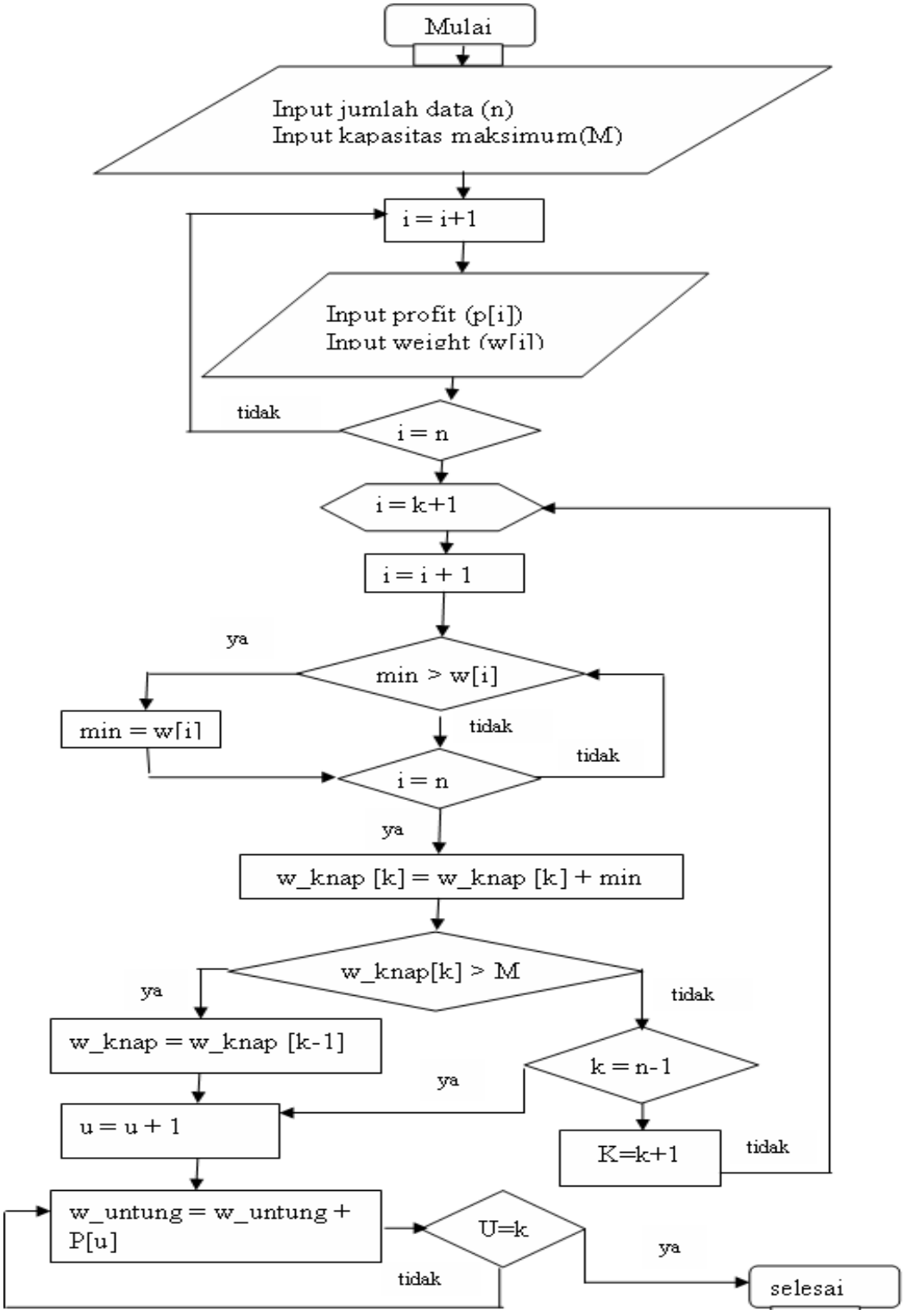


Gambar III.5 Graph Solusi Kurang Optimal dari *Greedy*

Tetapi ingat bahwa untuk kasus umum, kerap kali algoritma *greedy* memberikan hasil yang *cukup baik* dengan kompleksitas waktu yang cepat. Hal ini mengakibatkan algoritma *greedy* sering digunakan untuk menyelesaikan permasalahan kompleks yang memerlukan kecepatan jawaban, bukan solusi optimal, misalnya pada game.

III.3 *Flowchart* Algoritma *Greedy*

Flowchart adalah suatu bagan atau symbol-symbol yang menggambarkan urutan proses secara mendetail hubungan antara suatu proses (instruksi) dengan proses lainnya dalam suatu program.



Gambar III.6 *Flowchart* Algoritma *Greedy*

1. *Greedy by weight* berdasarkan *Flowchart* Gambar III.6

   Perhitungan loop mencari min > [i]

   Proses pilihan ya

   Untuk mencari min = w[i], jumlah perhitungannya = 2 kali dan

   proses looping sampai n − 1 kali, jadi jumlah perhitungannya = 2(− 1)

   Proses pilihan tidak

   Untuk *i* = 2, jumlah perhitungan = n − 1 kali

   Untuk = n, jumlah perhitungan = n − 1 kali

   Jadi jumlah perhitungan kompleksitas waktunya adalah

   T(n) = 2(− 1) + − 1 = 2 − 2 + − 1 = 3 − 3

   Perhitungan w _knap [ ] =w _ knap[ ] + min= 1 kali 48

   Perhitungan syarat w _knap [k] >M

   Proses pilihan tidak

2. Untuk mencari w _knap [k] > M dengan syarat K = − 1

   adalah (3n − 3 + 1 + 1)n − 1 = (3 − 1)n − 1

   Jadi ( ) = $3n^2$ – 4n + 1

   Proses pilihan ya

   Perhitungan w _knap = w _ knap[− 1] = 1 kali,

   Perhitungan looping w _untung =w _untung +p [u]

   Untuk *u* = 1, jumlah perhitungan =k

   Untuk = k, jumlah perhitungan = k

   Jadi ( ) = 3 − 3 + 1 + 1 + = 3 − 1 + k

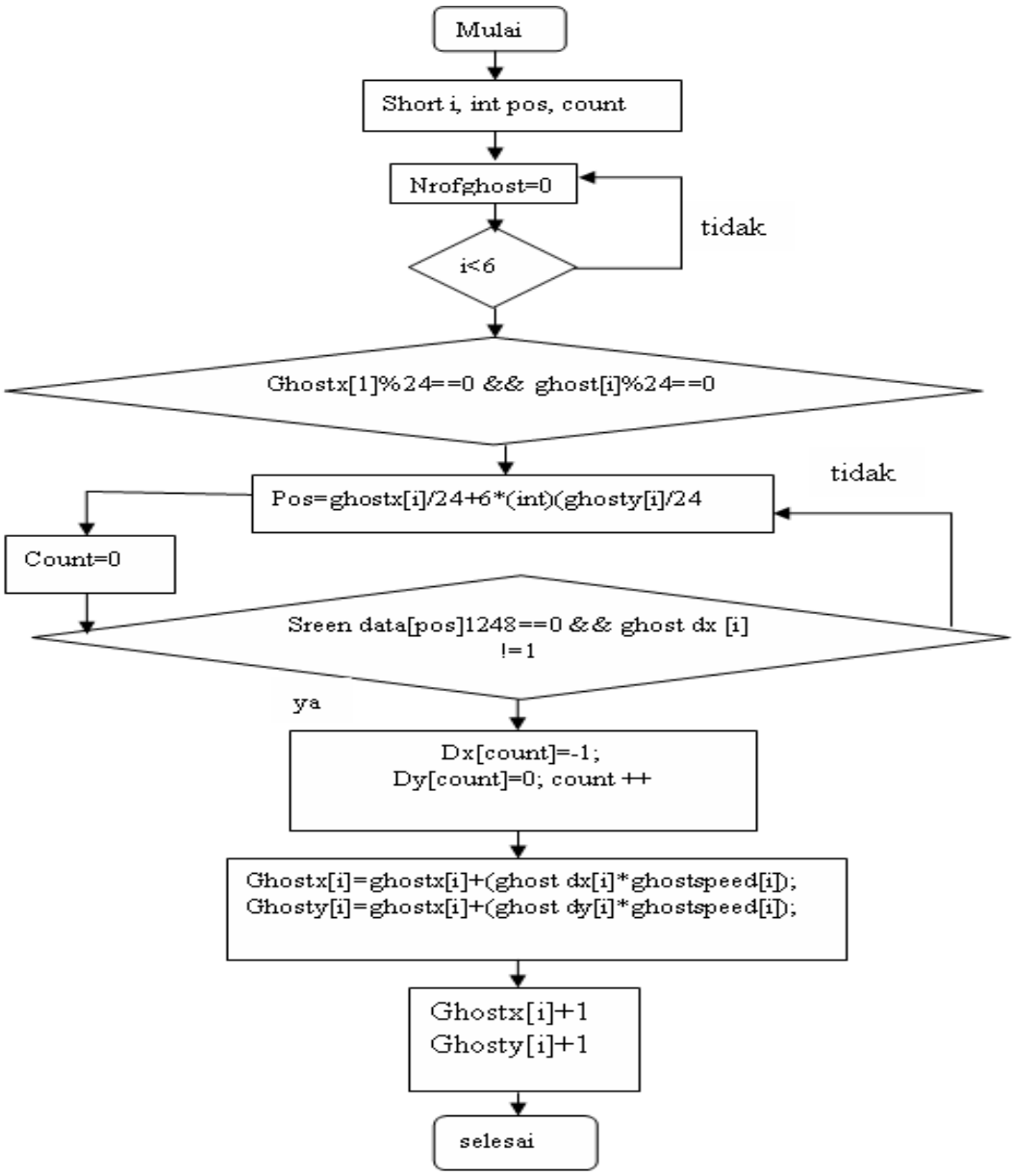
   Jadi jumlah perhitungan kompleksitas waktunya adalah

$T(n) = 3n^2 - 4 + 1 + 3 - 1 + = 3n^2 - + k$

Jadi jumlah perhitungan kompleksitas waktu *Greedy by Weight* adalah

$T( ) = 3 - 3 + 1 + 3n^2 - + = 3n^2 + 2 + - 2, (n^2)$

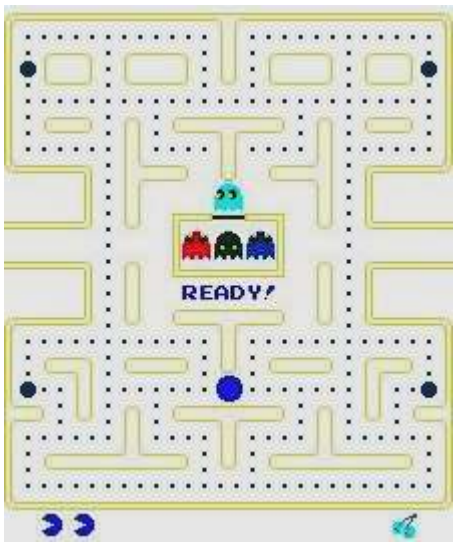
III.4 *Flowchart* Pergerakan *Ghost*



Gambar III.7 *Flowchart* Pergerakan *Ghost*

Penyelesaian Kasus

Untuk melakukan hal ini, kita harus memberikan prioritas yang berbeda-beda pada masing-masing musuh, maka dengan sendirinya dia akan bergerak ke arah yang berbeda. Kita ambil contoh dari keempat hantu itu, misalnya hantu yang berwarna merah. Hantu merah ini memulai gerakannya diluar rumah hantu. Dan digambarkan sebagai ancaman pertama yang muncul karena gerakannya hampir selalu di belakang pemain, begitu juga untuk ketiga hantu yang lainnya, mereka memiliki karakteristik masing-masing untuk mengejar target. Berikut merupakan tampilan awal dari game Pacman :



Gambar III.8. Pacman

Hantu yang berwarna merah akan terus mengikuti targetnya, jika target ke kanan maka hantu akan ke kanan, jika ke kiri maka dia juga ke kiri. Untuk hantu-hantu yang lainnya juga memiliki kemampuan dan karakteristik yang berbeda, misalkan hantu biru yang baru keluar dari rumah hantunya, dia ikut mengejar pemain, maka dengan menggunakan kemampuan yang diprogramkan, dapat dilihat apakah saat mengejar targetnya dia akan mendapat halangan (dinding

labirin) atau tidak, maka disinilah para hantu di berikan kecerdasan untuk mengambil sebuah keputusan yang baik.

III.5 Analisis Kebutuhan Non Fungsional

Analisis kebutuhan non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Analisis ini juga berisi komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun hingga sistem tersebut dapat digunakan.

Pada analisis kebutuhan non fungsional ini dijelaskan analisis kebutuhan perangkat keras, analisis kebutuhan perangkat lunak, dan analisis pengguna.

III.5.1 Analisis Perangkat Keras

Perangkat keras yang digunakan untuk membangun Penerapan Algoritma *Greedy* Untuk Pergerakan *Ghost* Pada Permainan *Pacman* adalah laptop Asus dengan spesifikasi sebagai berikut :

1. Processor Intel i3
2. Hardisk 500 GB
3. RAM 2 GB
4. VGA NVDIA GeForce GTS 250 2796MB
5. Keyboard

III.5.2 Analisis Kebutuhan Perangkat Lunak

Untuk membangun aplikasi ini, perangkat lunak yang dibutuhkan adalah:

1. Sistem Operasi Windows 8.1 64-bit
2. Netbeans
3. Java

Sedangkan untuk menjalankan aplikasi ini, perangkat lunak yang dibutuhkan hanyalah sebuah Sistem Operasi yang menjadi kebutuhan minimal pada kebutuhan non fungsional.

Dari analisis kebutuhan perangat lunak dapat disimpulkan bahwa untuk menjalankan *game* ini, pemain dapat langsung meng klik tombol permainanya tanpa harus menambah *software* apapun selain dari *software* dari komponen *windows* yang sudah ada.

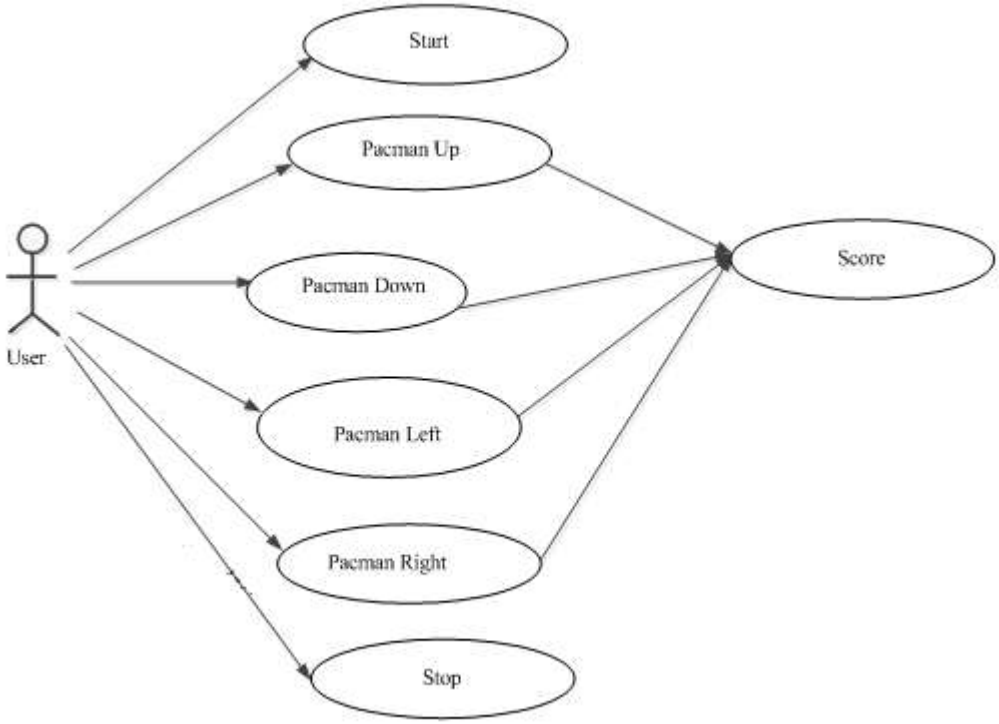
III.6 Perancangan Sistem

Perancangan adalah langkah pertama dalam fase pengembangan rekayasa Produk atau sistem. Perancangan itu adalah proses penerapan berbagai teknik dan prinsip yang bertujuan untuk mendefinisikan sebuah peralatan, satu proses atau satu system secara detail yang membolehkan dilakukan realisasi fisik. Fase ini adalah inti teknis dari proses rekayasa perangkat lunak.

Pada fase ini elemen-elemen dari model analisa dikonversikan. Dengan menggunakan satu dari sejumlah metode perancangan, fase perancangan akan menghasilkan Perancangan Antarmuka, *Use case*, *Class Diagram*, *Activity Diagram* sampai ke pembuatan program.

III.6.1. Use Case Diagram

Use case diagram adalah suatu bentuk *diagram* yang menggambarkan fungsionalitas yang diharapkan dari sebuah sistem dilihat dari perspektif pengguna diluar sistem.



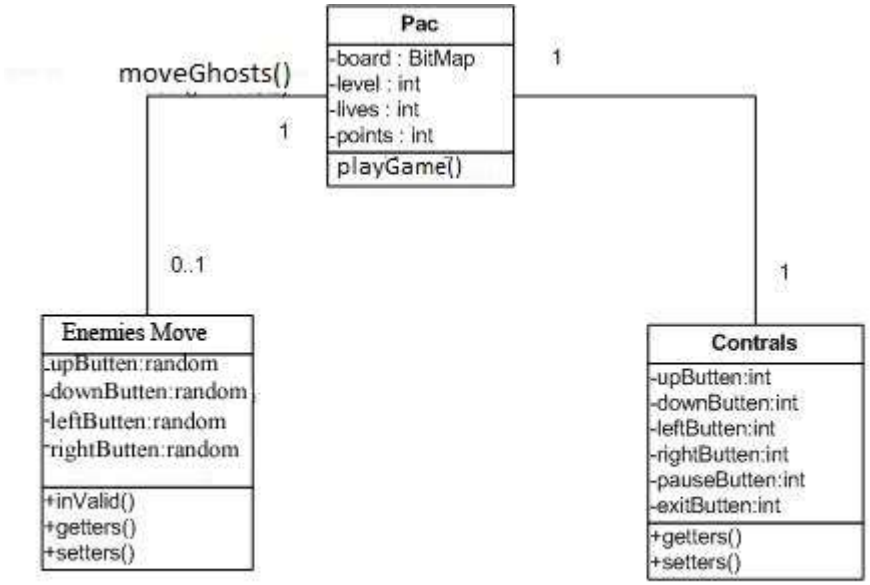
Gambar III.9 *Use Case Diagram*

Keterangan untuk *use case* di atas adalah :

1. *User* Klik *Start* Untuk Memulai permainan
2. *User* menggerakkan kursor tanda panah pada *keyboard*
    a. Pacman *Up* akan bergerak jika *user* menekan tombol atas
    b. Pacman *Down* akan bergerak jika *user* menekan tombol bawah
    c. Pacman *Left* akan bergerak jika *user* menekan tombol kanan
    d. Pacman *Right* akan bergerak jika *user* menekan tombol kiri
2. Melihat *score* dari permainan

III.6.2. Class diagram

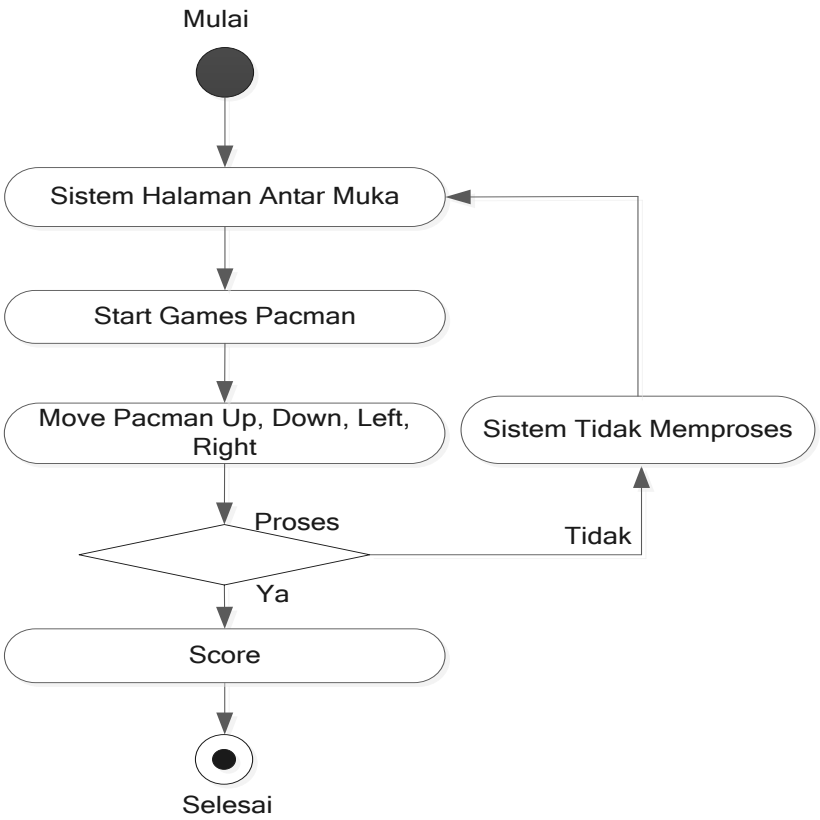
Class diagram menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang dibuat untuk membangun sistem. Perancangan struktur sistem yang terdapat pada Penerapan Algoritma *Greedy* Untuk Pergerakan *Ghost* Pada Permainan Pac-Man dapat dilihat pada Gambar III.3.



Gambar III.10 *Class Diagram*

III.6.3. *Activity Diagam*

Activity diagram merupakan suatu *diagram* yang dapat menampilkan secara detail urutan proses dari aplikasi. Perancangan aplikasi dapat digambarkan dengan menggunakan *activity diagram* sebagai berikut:



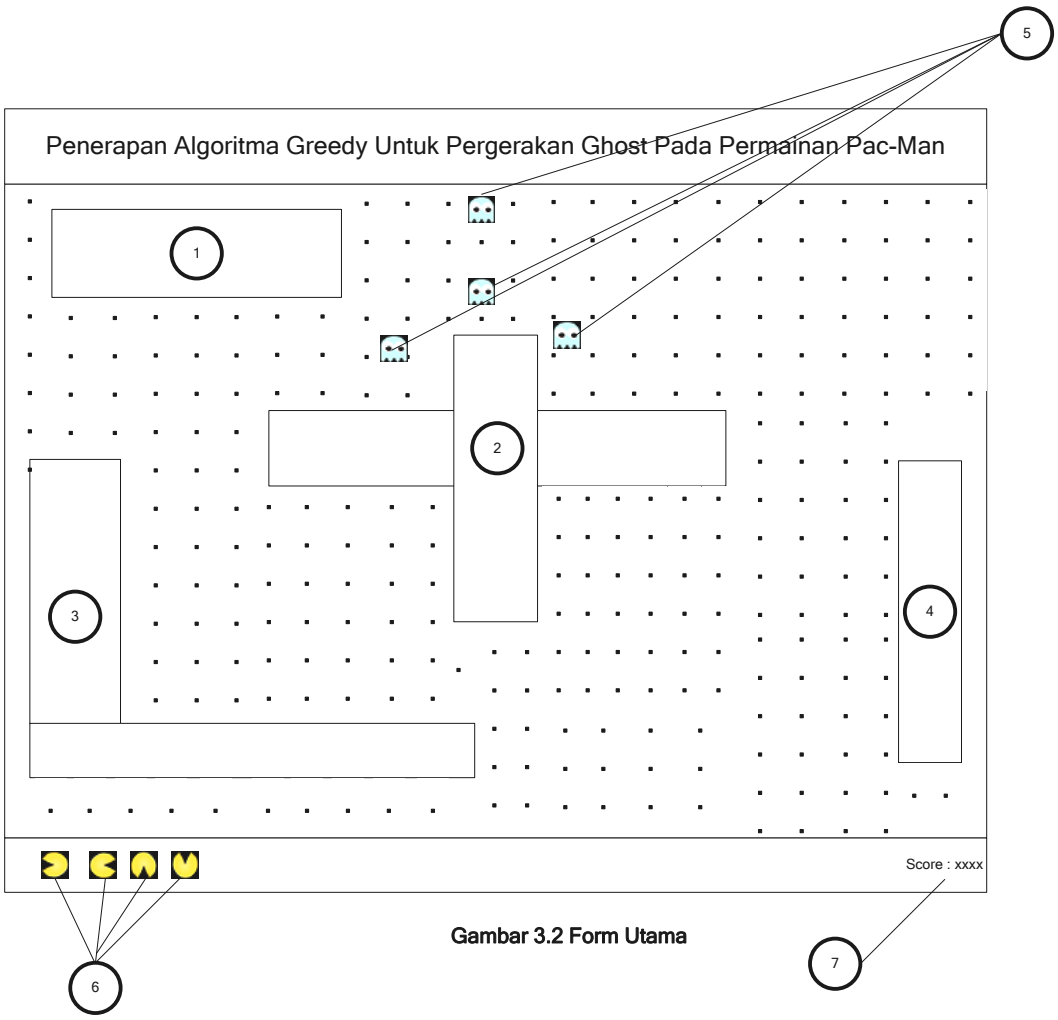
Gambar III.11 *Activity Diagram*

Keterangan :

1. *User* masuk ke halaman utama atau antar muka
2. Pemain menekan tombon panah atas, bawa, kiri dan kanan pada *keyboard*
3. Jika *user* memproses maka akan ditampilkan *score* dan jika tidak nilai variabel atau attribut tidak diproses.

1. Form Antar Muka

Rancangan antar muka ini nantinya yang menghubungan *user* dengan sistem. Pada rancangan ini terdiri dari beberapa batas-batas untuk *pacman*, *pacman* terdiri dari 4 dan *ghost* di 4 posisi



Gambar III.12 Form Menu Utama

Keterangan dari rancangan *interface* Penerapan Algoritma *Greedy* Untuk Pergerakan *Ghost* Pada Permainan *Pacman* adalah sebagai berikut :

1. Nomor 1,2,3 dan 4 adalah batas batas pacman dalam melakukan pergerakan
2. Nomor 6 Terdiri dari 4 Posisi pacman, yaitu pacman yang mengarah ke kiri, ke kanan, ke atas dan ke bawah, dan pacman akan berganti sesuai dengan tombol pada keyboard yang di tekan, jika memilih tanda panah kiri maka pacman yang kearah kiri akan digerakkan, jika memilih tanda panah kanan maka pacman yang kearah kanan akan digerakkan jika memilih tanda panah

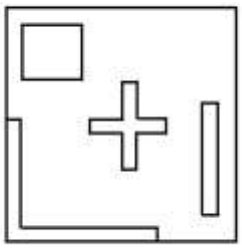
atas maka pacman yang kearah atas akan digerakkan dan jika memilih tanda panah bawah maka pacman yang ke arah bawah akan digerakkan.

3. Nomor 5 adalah *ghost* dimana *ghost icon* terdiri dari 4 yang akan bergerak secara random, jika *pacman* mengenai salah satu *ghost* maka pacman dianggap kalah.
4. Nomor 7 adalah *score* yang akan di tampilkan buat pemain.

III.4 *Storyboard*

Tabel III.1. Storyboard Penerapan Algoritma *Greedy* Untuk Pergerakan *Ghost* Pada Permainan Pacman

No	Tampilan Objek	Keterangan
1		Tampilan ini menunjukkan ckarakter pacman ke kanan jika tombol kanan di tekan maka arah pacman ke kanan
2		Tampilan ini menunjukan karakter pacman ke bawah jika tombol bawah di tekan maka arah pacman ke bawa

3		Tampilan ini menunjukkan karakter pacman ke kiri jika tombol kiri di tekan maka arah pacman ke kiri
4		Tampilan ini menunjukkan karakter pacman ke atas jika tombol atas di tekan maka arah pacman ke atas
5		Tampilan ini menunjukkan karakter *ghost* musuh pacman
6		Tampilan ini menunjukkan area permainan pacman