Aplikasi Graf pada Permainan Minesweeper

Rania Dwi Fadhilah - 13520142
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13520142@std.itb.ac.id

***Abstract*—Minesweeper adalah salah satu permainan *single-player* yang pertama kali muncul pada sistem operasi Windows. Permainan ini berbentuk sebuah kumpulan petak dengan p baris dan q kolom yang berisi sejumlah ranjau. Petak dapat berisi ranjau, angka, ataupun kosong. Objektif dari permainan ini adalah membuka semua petak bukan ranjau tanpa membuka petak ranjau. Untuk menemukan posisi ranjau, digunakan petunjuk yang tersedia pada angka pada petak. Angka pada petak menunjukkan jumlah ranjau yang ada di sekeliling petak tersebut. Permainan Minesweeper ini masuk ke dalam golongan masalah *NP-Complete*. Pada pelajaran Matematika Diskrit IF2120, Mahasiswa diajarkan tentang materi graf. Materi graf ini dapat digunakan untuk menyelesaikan permasalahan Minesweeper dengan mencari strategi khususnya.**

***Keywords*— graf, matdis, minesweeper, np-complete.**

I. PENDAHULUAN

Sejak dahulu kala, manusia, yang muda maupun tua, kerap mencari hiburan untuk mengisi waktu luang atau ketika sedang jenuh. Bermain menjadi salah satu pilihan hiburan yang sering dilakukan oleh manusia.

Definisi dari *Game* atau permainan sendiri berdasarkan kamus Oxford adalah suatu kegiatan yang dilakukan untuk mencari kesenangan, biasanya memiliki aturan tertentu, dan dapat berupa menang-kalah.

Terdapat berbagai macam permainan yang dapat dimainkan oleh seseorang. Dari mulai bermain bola, hingga Sudoku, Minesweeper, dan permainan-permainan lainnya. Ada banyak permainan yang melibatkan pemikiran yang cenderung kompleks. Tak jarang pula ada permainan yang terlihat sederhana, dengan aturan yang simpel, namun sulit dan menantang ketika dijalankan.

Minesweeper adalah salah satu contoh permainan yang terlihat simpel namun ternyata menantang. Permainan ini dibuat untuk dimainkan sendirian. Permainan ini cukup mengasah otak dan apabila dikategorikan pada tingkat kesulitan secara matematisnya, maka Minesweeper dikenal sebagai permasalahan *NP-Complete*.

Seperti layaknya semua permainan lainnya, terdapat strategi khusus yang dapat diterapkan untuk menyelesaikan suatu permainan. Oleh karena itu, dengan mempelajari graf dan metode-metode pencariannya, akan dilakukan sebuah penelitian untuk memecahkan masalah permainan Minesweeper.

II. DASAR TEORI

A. Minesweeper

Minesweeper merupakan salah satu permainan *single player* yang cukup digemari, terutama oleh para pengguna komputer Windows. Permainan ini berbentuk *frame* yang terdiri dari p baris dan q kolom yang berisikan ranjau, kosong, maupun angka. Minesweeper adalah salah satu bentuk permainan yang memiliki aturan-aturan sederhana namun implikasi yang menantang. Oleh sebab itu, permainan ini termasuk ke dalam golongan masalah *NP-Complete*.

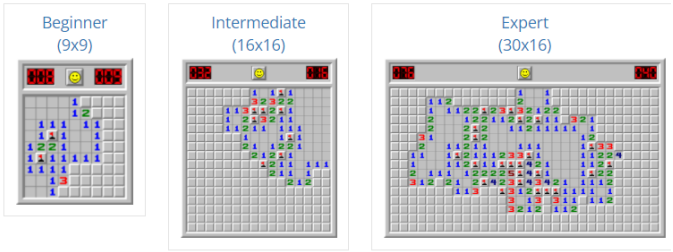
Permainan ini dibuat oleh Robert Donner dan Curt Johnson pada tahun 1991, sebagai bagian dari Windows Entertainment Pack. Namun, asal-usulnya sendiri berawal dari permainan *mainframe* yang terbentuk pada awal tahun 1960-an dan 1970-an. Ide paling awalnya sendiri adalah Jerimac Ratliff's *Cube*.

Aturan dan cara bermain Minesweeper dapat dikatakan cukup sederhana. Pada awal permainan, pemain akan disuguhkan dengan sebuah *frame* yang terdiri atas petak-petak yang belum terbuka. Klik pertama pada *frame* akan membuka beberapa petak yang berisikan angka maupun kosong. Klik pertama tidak mungkin berisi ranjau. Kemudian, pemain harus menentukan petak-petak lain yang harus dibuka. Apabila pemain memilih petak berisi ranjau, maka permainan akan secara otomatis berakhir. Petak berisi angka sendiri menunjukkan jumlah ranjau yang terdapat di sekitar petak angka tersebut, yaitu bagian atas, kanan, kiri, dan bawah dari petak angka (ukuran 3x3 dengan petak angka sebagai pusat). Oleh sebab itu, jangkauan angka yang tertera pada petak angka adalah 0-8. Objektif dari game ini adalah untuk membuka semua petak yang tidak mengandung ranjau. Untuk membantu proses permainan ini, Minesweeper memperbolehkan pemain untuk memberi *flag* pada lokasi kemungkinan ranjau. Permainan ini cukup menjebak karena ada beberapa saat dimana pemain harus menebak petak mana yang bukan ranjau tanpa penggunaan unsur lojik.

Gambar 1 Contoh petak angka 3 x 3

Gambar 2 Contoh petak angka yang menjebak

Jumlah petak dari permainan ini disesuaikan dengan tingkat kesulitannya. Umumnya, terdapat 3 tingkat kesulitan, yaitu *beginner*, *intermediate*, dan *expert*. Ukuran *frame beginner* biasanya adalah 8 x 8, 9 x 9, atau 10 x 10 dengan ranjau berjumlah 10. Sedangkan, untuk *intermediate* adalah 13 x 15 atau 16 x 16 dengan ranjau berjumlah 40. Untuk level tersulit sendiri, atau *expert*, biasanya terdiri dari 30 x 16 petak dan memiliki 99 ranjau. Semakin tinggi tingkat kesulitannya, maka semakin banyak pula penalaran deduktif yang harus diterapkan.



Gambar 3 Tingkat kesulitan permainan Minesweeper (https://minesweeper.online/)

B. NP-Complete

Waktu untuk menyelesaikan sebuah algoritma sangatlah bervariasi, mulai dari $O(1)$, $O(n)$, $O(n^2)$, dan sebagainya. Algoritma ini dikenal sebagai solusi polynomial karena kebutuhan waktunya secara asimptotik terbatas oleh suatu fungsi polinomial.

NP adalah kumpulan masalah keputusan yang dapat diselesaikan oleh *Non-deterministic Turing Machine* dengan waktu polinom. *NP-Complete* sendiri merupakan masalah paling sulit yang ada pada set NP. Tidak ada algoritma waktu polinom yang ditemukan untuk semua masalah *NP-Complete*, namun tidak ada pula bukti konkret bahwa tidak ada algoritma waktu polinom untuk semua masalah tersebut.

Sebuah masalah dapat dikategorikan sebagai *NP-Complete* apabila :

1. Solusi dari masalah dapat diverifikasi dengan cepat dan algoritma pencarian *brute-force* dapat menemukan seolusi dengan mencoba semua kemungkinan solusi.
2. Masalah dapat digunakan untuk mensimulasikan permasalahan lainnya dan dapat diverifikasi dengan cepat bahwa solusi tersebut benar.

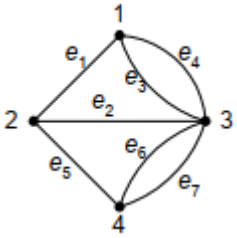
Berdasarkan buku Garey and Johnson, terdapat sekitar 3000 masalah yang terdeteksi *NP-Complete*. Contoh permainan yang bersifat *NP-Complete* adalah Battleship, Minesweeper, Numberlink, dan Nonograms.

C. Graf

a. Definisi Graf

Secara matematis, graf dapat didefinisikan sebagai pasangan himpunan *Vertices* (V) dengan *Edges* (E) yang apabila ditulis sebagai notasi akan berbentuk G = (V,E). Dalam bahasa Indonesia, *Vertices* dikenal sebagai simpul dan *Edge* dikenal sebagai sisi. Himpunan V tidak boleh kosong, sedangkan himpunan E boleh kosong.

Penamaan simpul dapat memanfaatkan angka (0,1,2,…) maupun huruf (a, b, c, …), ataupun keduanya (v0, v1, v2, …..). Sedangkan, penamaan sisi ditulis sebagai pasangan simpul (v0, v1) atau dengan lambang (e0, e1, e2, …) yang apabila e0 menghubungkan v0 dan v1, maka dapat ditulis e0 = (v0, v1).



Gambar 4 Ilustrasi graf (PPT Pak Rinaldi Munir)

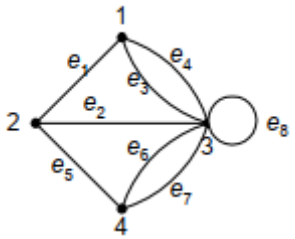
Sebagai contoh, pada gambar 4, maka dapat disimpulkan bahwa :

V = { 1, 2, 3, 4 }

E = { (1,2), (2,3), (1,3), (1,3), (2,4), (3,4), (3,4) }

= { *e*1, *e*2, *e*3, *e*4, *e*5, *e*6, *e*7 }

b. Jenis Sisi



Gambar 5 Ilustrasi graf semu

Terdapat 2 jenis sisi yang dimiliki oleh sebuah graf, yaitu :

1. Sisi ganda atau *multiple edges*

Sisi ganda adalah istilah yang digunakan untuk menggambarkan suatu kondisi dimana terdapat dua buah sisi yang menghubungkan dua buah simpul yang sama. Pada gambar 5, contoh simpul ganda adalah e_3-e_4 karena sama-sama menghubungkan simpul 1 dan 3 dan e_6-e_7 karena sama-sama menghubungkan simpul 3 dan 4.

2. Gelang atau *loop*

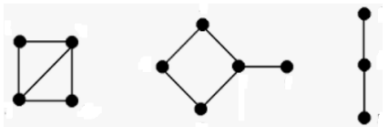
Gelang adalah istilah yang digunakan untuk kondisi dimana sisi diawali dan diakhiri oleh satu simpul yang sama. Pada gambar 5, contoh dari gelang adalah sisi e8 yang menghubungkan simpul 3 ke simpul 3.

c. Jenis Graf

Graf digolongkan ke dua jenis berdasarkan keberadaan sisi

ganda maupun gelang pada graf, yaitu :

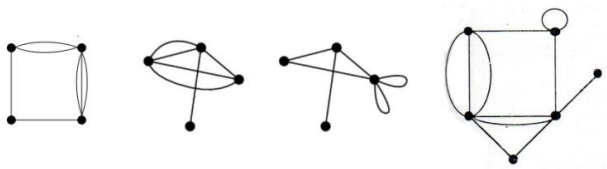
1. Graf sederhana



Gambar 6 Ilustrasi graf sederhana

Graf yang tidak memiliki gelang maupun sisi ganda dapat disebut sebagai graf sederhana.

2. Graf tak-sederhana



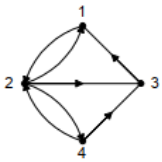
Gambar 7 Ilustrasi graf tak-sederhana

Graf yang memiliki gelang atau sisi ganda disebut sebagai graf tak-sederhana. Graf tak-sederhana dapat dibedakan lagi menjadi beberapa jenis, yaitu :

2a. Graf ganda
Graf ganda adalah graf yang memiliki sisi ganda dan tidak memiliki sisi gelang.
2b. Graf semu
Graf semu adalah graf yang memiliki sisi gelang dan dapat memiliki sisi ganda juga.

Selain berdasarkan keberadaan sisinya, graf juga dapat dikategorikan menjadi dua jenis berdasarkan orientasi arah pada sisinya., yaitu :

1. Graf berarah



Gambar 8 Ilustrasi graf berarah

Graf berarah adalah graf yang setiap sisinya memiliki orientasi arah tertentu.

2. Graf tak-berarah

Gambar 9 Ilustrasi graf tak-berarah

Graf tak-berarah adalah graf yang setiap sisinya tidak memiliki orientasi arah.

Selain itu, graf juga dapat dikategorikan berdasarkan jumlah simpulnya, yaitu :

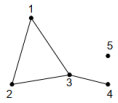
1. Graf berhingga
Graf yang jumlah simpulnya berhingga.

2. Graf tak-berhingga
Graf yang jumlah simpulnya tak berhingga.

d. Terminologi Graf
1. Ketetanggaan (*Adjacent*)
Dua buah simpul dapat disebut bertetangga apabila kedua simpulnya terhubung langsung. Sebagai contoh, pada gambar 8, simpul 1 bertetangga dengan simpul 2 dan 3 tapi tidak bertetangga dengan simpul 4.

2. Bersisian (*Incidency*)
Untuk sembarang sisi e = (v_i, v_j), dapat dikatakan bahwa e bersisian dengan simpul v_i dan simpul v_j. Sebagai contoh, pada gambar 4, maka e_1 bersisian dengan simpul 1 dan e_1 juga bersisian dengan simpul 2.

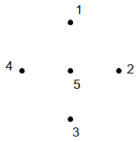
3. Simpul terpencil (*Isolated Vertex)*



Gambar 10 Ilustrasi simpul terpencil

Simpul terpencil adalah simpul yang tidak memiliki sisi yang bersisian dengannya. Pada gambar 10, simpul terpencilnya adalah simpul 5.

4. Graf kosong (*null graph)*



Gambar 11 Ilustrasi graf kosong

Graf yang himpunan sisinya berupa suatu himpunan kosong.

5. Derajat (*Degree*)
Derajat suatu simpul menyatakan jumlah sisi yang bersisian dengan simpul tersebut. Derajat dapat ditulis dengan notasi d(v). Sebagai contoh, pada gambar 4, maka :
d(1) = d(4) = d(2) = 3
d(3) = 5
Jumlah derajat semua simpul pada suatu graf pasti genap.

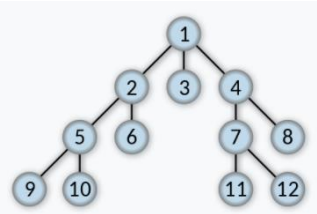
6. Lintasan
Lintasan merupakan sisi-sisi yang harus dilalui untuk menghubungkan suatu simpul dengan simpul lainnya. Panjang lintasannya sendiri dihitung berdasarkan jumlah sisi atau jumlah bobot dari tiap sisi lintasan.

7. Siklus
Siklus merupakan istilah yang digunakan untuk menggambarkan lintasan yang diawali dan diakhiri oleh simpul

yang sama.

D. Algoritma BFS dan DFS

Pada teori graf, terdapat dua algoritma pencarian yang biasa digunakan untuk menulusuri semua simpul pada graf. Algoritma tersebut biasa disebut BFS (*Breadth First Search*) dan DFS (*Depth First Search*).



Gambar 12 Contoh graf
(https://en.wikipedia.org/wiki/Breadth-first_search)

BFS adalah algoritma yang melakukan pencarian secara melebar dan memanfaatkan struktur data *Queue* untuk mencari jalur terpendeknya. Pencarian pada graf menggunakan BFS dimulai pada simpul akar, kemudian dilanjutkan dengan menelusuri seluruh simpul tetangganya. Kemudian, untuk setiap simpul terdekat, dilakukan pencarian terhadap simpul tetangga yang belum ditelusuri. Proses ini diulang hingga solusi berhasil ditemukan. Sebagai contoh, pada gambar 12, maka *queue*-nya adalah 1-2-3-4-5-6-7-8-9-10-11-12. Apabila graf berbentuk pohon berakar, maka semua simpul pada level d harus dikunjungi sebelum mengunjungi semua simpul pada level d+1.

DFS adalah algoritma yang melakukan penelusuran berdasarkan kedalamannya dan memanfaatkan struktur data *Stack*. Pencarian ini dimulai pada simpul akar, kemudian simpul anak sebelah kiri, dan dilanjutkan dengan simpul anak pertama dari simpul anak pertama pada level sebelumnya hingga mencapai level terdalamnya. Setelah mencapai level terdalamnya, maka pencarian akan kembali ke 1 level sebelumnya dan dilakukan pencarian terhadap simpul anak kedua. Langkah-langkah ini diulang hingga ditemukan solusi. Sebagai contoh, pada gambar 12, maka *stack*-nya adalah 1-2-5-9-10-6-3-4-7-11-12-8.

Kompleksitas waktu dari BFS dan DFS serupa, yaitu $O(V + E)$ apabila digunakan list ketetanggan, dan $O(V^2)$ apabila digunakan matriks ketetanggaan, dengan V sebagai simpul dan E sebagai sisi.

III. ANALISIS PERMAINAN MINESWEEPER

Pada permainan Minesweeper, terdapat berbagai pola dan petak bernomor yang muncul. Untuk menebak suatu ranjau pada petak, terdapat ketertarikan untuk menyelesaikan pola yang sudah diketahui terlebih dahulu, untuk kemudian mendapat pola-pola lainnya yang dapat mempercepat proses penyelesaian permainan.

A. Analisis Single-Square

Analisis ini digunakan apabila pemain dihadapkan dengan kasus dimana jumlah petak yang belum terbuka berjumlah sama dengan angka yang tersedia.

Gambar 13 Contoh analisis *single-square*

Sebagai contoh, pada gambar 13, petak pusatnya adalah angka 1 dan jumlah petak yang belum terbukapun hanya 1. Oleh sebab itu, petak tersebut dapat dipastikan sebagai petak ranjau dan dapat ditandai menggunakan *flag*.

Gambar 14 Contoh analisis 2 *single-square*

Pada gambar 14, petak pusat berisi angka 1. Ranjau telah ditandai dengan gambar *flag*. Oleh sebab itu, A dan B dapat dibuka dengan aman karena ranjau hanya terdapat 1 ranjau yang sudah ditandai oleh *flag* sehingga A dan B pasti bukan ranjau.

B. Analisis *Multiple-Square*

Terdapat berbagai macam pertimbangan yang harus dilakukan apabila ingin memecahkan suatu pola yang lebih kompleks.

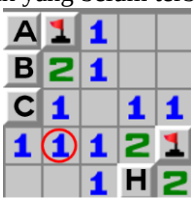
Gambar 15 Contoh analisis multiple-square

Terdapat berbagai strategi yang dapat digunakan untuk menebak letak dari ranjau, yaitu :

1. Ketika ada dua petak angka yang bertetangga, maka selisih dari angka adalah jumlah ranjau yang terdapat pada tiga petak yang bertetangga dengan masing-masing petak angka yang tidak bertetangga dengan petak angka lainnya.
2. Dengan metode yang sama, mengambil kesimpulan untuk membantu petak lain untuk menentukan jumlah ranjau yang terdapat pada sejumlah petak tanpa mengetahui mana yang aman dan berisi ranjau.

Untuk menyelesaikan permasalahan pada gambar 15, maka dengan mencoba setiap kombinasi menggunakan *brute force*, terdapat 3 konfigurasi yang valid. Apabila digambarkan menggunakan himpunan dengan 1 sebagai petak ranjau, maka {A,B,C,D,E,F,G,H} = {1,0,0,1,0,0,0,1}, {0,1,0,0,1,0,0,1}, {0,0,1,0,0,0,0,1}. Melalui analisis ini, dapat disimpulkan bahwa petak F dan G adalah petak aman yang tidak berisikan ranjau dan petak H sudah dipastikan sebagai petak ranjau.

Apabila petak F dan G dibuka, maka akan muncul petak angka atau kosong yang dapat digunakan sebagai petunjuk untuk memecahkan petak-petak yang belum terbuka lainnya.



Gambar 16 Analisis multiple square yang telah dibuka

Apabila petak F dan G dibuka, maka akan muncul petak angka atau kosong yang dapat digunakan sebagai petunjuk untuk memecahkan petak-petak yang belum terbuka lainnya. Contohnya terdapat pada gambar 16. Dengan ini, dapat disimpulkan bahwa ranjau berada di posisi C karena dapat dilakukan analisis *single-square* untuk petak berangka 1 yang dilingkari oleh warna merah.

C. Analisis Final

Analisis ini digunakan pada akhir permainan ketika seluruh petak pada papan adalah aman atau telah memiliki *flag*. Pada tahap ini, kadang terdapat petak-petak yang tidak dapat dipecahkan menggunakan lojik dan harus dilakukan penebakan. Contohnya ada pada gambar 2. Apabila pemain dihadapkan dengan masalah ini pada tengah permainan, maka masalah ini dapat diabaikan dengan cara menyelesaikan petak-petak lainnya terlebih dahulu. Namun, apabila dihadapkan dengan masalah ini pada akhir permainan, maka tidak ada strategi khusus yang dapat digunakan untuk menyelesaikan masalah selain dengan keberuntungan.

IV. PENERAPAN GRAF DALAM MENYELESAIKAN PERMAINAN MINESWEEPER

Permasalahan dari permainan Minesweeper dapat diselesaikan dengan menelusuri petak-petak yang mempunyai tetangga berupa ranjau. Dengan menganalisis angka pada petak dan jumlah-jumlah ranjau di sekitarnya, kemudian mengambil informasi tersebut untuk petak-petak tetangga, posisi ranjau dapat ditemukan.

Permainan Minesweeper dapat digambarkan sebagai suatu graf. Oleh karena itu, untuk melakukan penelusuran, dapat digunakan algoritma BFS. Dengan menggunakan algoritma ini, penelusuran akan mengunjungi setiap tetangga dan melakukan pelebaran ke samping.

Pemecahan persoalan minesweeper ini dapat dilakukan dengan algoritma berikut :

```
Program Minesweeper
{ Sebuah permainan yang terdiri atas p kolom dan q
baris dengan n buah ranjau. Permainan akan selesai
apabila semua petak bukan ranjau berhasil terbuka
dan tidak terkena ranjau }

KAMUS
    type Petak < i : integer {menunjukkan baris},
                 j : integer {menunjukkan kolom},
                 nilai : integer
                 { nilai dari petak →
                 0 = petak kosong;
                 1..8 = nilai petak angka yang
                        mungkin;
                 -1 = ditandai ?
                 9 = ditandai flag
                 999 = belum dibuka },
                 flag : integer { menunjukkan
                 jumlah flag } >
    petak : Petak
    Papan : array [1..p] of [1..q] of Petak

    function adaRanjau (P : Petak) → Boolean
    { Mengembalikan true apabila petak yang
    dikunjungi adalah petak ranjau }

    function belumCek (P : Petak) → Boolean
    { Mengembalikan true apabila ranjau belum
    dicek }

    function cekSingleSquare (P : Petak) →
    boolean
    { Mengembalikan true apabila dapat dianalisis
    menggunakan metode single square }

    function cekMultipleSquare (P : Petak) →
    boolean
    { Mengembalikan true apabila dapat dianalisis
    menggunakan metode multiple square }

    function gameSelesai () → Boolean
    { Mengembalikan true apabila semua ranjau
    telah berhasil dibuka }

    procedure telusurPetak (input P : Petak)
    { I.S. Petak terdefinisi dan petak yang belum
    terbuka bukan petak kosong }
    { F.S. Penulusuran petak pada papan berhenti }

    procedure telPetakQ ()
    { I.S. Petak terdefinisi }
    { F.S. Petak yang memiliki petak tetangga yang
    bisa dibuka dan dihapus dari Queue }

ALGORITMA PROGRAM UTAMA
    while (not adaRanjau(petak) or gameSelesai()
    do
        petak ← randomPetak()
        if (belumCek(petak)) then
            telusurPetak(petak)
            telPetakQ()

REALISASI FUNGSI DAN PROSEDUR

procedure telusurPetak (input P : Petak)
KAMUS LOKAL
    ranjauBuka : boolean
    i : integer
    temp : Petak
ALGORITMA PROSEDUR
    ranjauBuka ← false
```

```
    if ( P.flag <> 0 ) then
        ranjauBuka ← cekSingleSquare(P)
        if (not ranjauBuka) then
            insertQueue(P)
    else
        i traversal [1..8]
            temp ← petakArah(1)
            if (temp.nilai = 999 or (temp.flag ≥ 1
            and temp.flag ≤ 8)) then
                telusurPetak(temp)

procedure telPetakQ ()
KAMUS LOKAL
    ranjauBuka, satu, akhir : boolean
    i : integer
    headQ, pusat : Petak
ALGORITMA PROSEDUR
    i ← lengthQueue()
    akhir ← false
    satu ← true
    while (not akhir) do
        headQ ← headQueue()
        ranjauBuka ← cekMultiSquare(headQ)
        if (not ranjauBuka) then
            headQ ← deleteQueue()
            addQueue(headQ)
            if (satu) then
                satu ← false
                pusat ← headQ
        if (isQueueEmpty() or (headQ = pusat and
        lengthQueue() = i)) then
            akhir ← true
        else if (headQ = pusat) then
            n ← lengthQueue()
            satu ← true
```

Pemecahan masalah Minesweeper ini menggunakan beberapa struktur data, antara lain :

a. Queue : menyimpan petak bernilai 1 s.d. 8

b. Petak : menyimpan informasi letak petak dan nilai dari petak

c. Papan : terdiri atas sekumpulan petak

Program utama akan melakukan penelusuran ke petak-petak yang bisa dibuka dengan menggunakan petunjuk yang ada. Apabila penelusuran berhenti, maka petak dalam Queue akan ditelusuri untuk memberi petunjuk-petunjuk lain mengenai keberadaan ranjau ataupun petak berangka.

Terdapat berbagai macam fungsi dan prosedur yang dijalankan pada program ini. Fungsi-fungsi utamanya antara lain adalah :

1. adaRanjau → fungsi ini digunakan untuk mendeteksi keberadaan ranjau pada petak yang dipilih
2. belumCek → fungsi ini digunakan untuk mengecek apakah suatu petak sudah ditelusuri atau belum.
3. cekSingleSquare → fungsi yang memanfaatkan analisis *single square*.
4. cekMultipleSquare → fungsi yang memanfaatkan analisis *multiple square*
5. gameSelesai → fungsi yang menjadi parameter bahwa semua petak telah berhasil ditebak

Sedangkan, untuk prosedurnya sendiri, ada :

1. telusurPetak → Prosedur ini menggunakan metode rekursif BFS. Algoritma BFS ini digunakan karena sifat pencariannya lebih menyebar sehingga dapat dilakukan penelusuran secara lebih cepat dibandingkan dengan menggunakan DFS. Basisnya sendiri adalah apabila kondisi petak yang sedang dikunjungi memiliki petak tetangga yang bukan ranjau atau pasti aman. Prosedur ini akan membuka petak apabila petak tersebut aman dan akan menandai *flag* apabila isi petak dapat dipastikan sebagai ranjau.
2. telPetakQ → Prosedur ini akan menelusuri petak-petak yang terdapat dalam *queue* dimulai dari *head*. Apabila tetangga dari petak pada *head* dapat dibuka, maka tetangga tersebut akan terbuka dan *head* akan dihapus dari *queue*. Pengecekan tetangga dilakukan menggunakan analisis *single square* dan *multi square*. Apabila terdapat tetangga dari *head* yang belum terbuka, maka *head* akan diantrikan kembali pada *queue*. Apabila selama proses ini dilakukan selama 1 siklus, tidak terdapat perubahan pada *queue* maka proses akan dihentikan.

Algoritma penyelesaian Minesweeper ini belum dapat menghasilkan solusi yang akurat karena program dapat berhenti seketika ketika tidak ada petak yang dapat ditelusuri lagi, sehingga program akan mengambil petak yang berisi ranjau.

V. KESIMPULAN

Graf memiliki berbagai manfaat dalam kehidupan sehari-hari. Salah satunya adalah untuk mencoba menyelesaikan permainan Minesweeper. Dengan menggunakan *Breadth First Search* yang digunakan untuk meneliti sebuah graf, strategi penyelesaian permainan Minesweeper dapat ditemukan. Namun, secara keseluruhan, strategi penyelesaian permainan ini belum terlalu akurat dan tetap belum membuktikan adanya waktu pemecahan polinomial untuk permasalahan ini.

VI. UCAPAN TERIMA KASIH

Pertama-tama, penulis mengucapkan puji dan syukur kepana Tuhan yang Maha Esa, karena atas berkah dan rahmat yang diberikan-Nya, penulis dapat menyelesaikan masalah ini dengan tepat waktu. Penulis juga mengucapkan terima kasih yang sebesar-besarnya kepada seluruh dosen pengampu mata kuliah IF2120 Matematika Diskrit, terutama Ibu Nur Ulfa Maulidevi sebagai guru yang mengajar kelas K03 atas bimbingannya selama satu semester ini. Tak lupa, penulis juga mengucapkan terima kasih kepada keluarga dan teman-teman yang telah memberikan dukungan, baik dari segi moral maupun material sehingga dapat membantu penulis dalam menyelesaikan makalah ini. Semoga makalah ini dapat memberi manfaat kepada para pembaca.

DAFTAR REFERENSI

[1] https://www.researchgate.net/publication/220812024_Learning_Minesweeper_with_Multirelational_Learning. Diakses pada 10 Desember 2021.

[2] https://www.researchgate.net/publication/220557823_Minesweeper_on_graphs. Diakses pada 10 Desember 2021.

Waktu akses : 10 Desember 2021, pukul 16.01
[3] Kaye, Richard (2000). "Minesweeper is NP-complete". Mathematical Intelligencer. 22 (2): 9–15. Diakses pada 11 Desember 2021.
[4] Munir, Rinaldi, 2021. Graf (Bag.1). Disunting dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian1.pdf. Diakses pada 12 Desember 2021.
[5] Munir, Rinaldi, 2021. Graf (Bag.2). Disunting dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian2.pdf. Diakses pada 12 Desember 2021.
[6] Munir, Rinaldi, 2021. Graf (Bag.3). Disunting dari https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2020-2021/Graf-2020-Bagian3.pdf. Diakses pada 12 Desember 2021.
[7] https://dash.harvard.edu/bitstream/handle/1/14398552/BECERRA-SENIORTHESIS-2015.pdf. Diakses pada 13 Desember 2021.
[8] https://dspace.cvut.cz/bitstream/handle/10467/68632/F3-BP-2017-Cicvarek-Jan-Algorithms%20for%20Minesweeper%20Game%20Grid%20Generation.pdf?sequence=-1&isAllowed=y. Diakses pada 13 Desember 2021.
[9] https://www.geeksforgeeks.org/difference-between-bfs-and-dfs/. Diakses pada 13 Desember 2021.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 14 Desember 2021

Rania Dwi Fadhilah
13520142