# Penyelesaian Permainan Minesweeper dengan Algoritma BFS dengan Optimasi Algoritma Greedy

Erdiansyah Fajar Nugraha / 13508055
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia*
*18055@students.if.itb.ac.id*

***Minesweeper*** **merupakan salah satu permainan yang sudah terinstall pada sistem operasi Windows. Permasalah pada permainan ini adalah menemukan seluruh ranjau bom pada kumpulan petak yang ada tanpa meledakannya.**

**Cara menemukan ranjau adalah menggunakan petunjuk yang ada pada petak yang terbuka (non ranjau), petak tersebut berisikan angka yang menyatakan banyaknya ranjau yang ada disekitar petak yang berisikan angka tersebut.**

**Maka untuk memecahkan masalah ini dibutuhkan cara tertentu, yaitu menggunakan Algoritma BFS (*Breadth First Search*) atau Pencarian Melebar, dan optimasi pemilihan petak dengan Algoritma Greedy.**

**Algoritma BFS memiliki karakteristik pencarian melebar, yaitu mengunjungi seluruh anak-anak dari suatu simpul atau keadaan status dahulu baru setelah itu memperluasnya kembali. Algoritma ini digunakan sebagai bagian dari pemecahan masalah, yaitu untuk memfasilitasi proses penelusuran petak-petak yang ada pada papan *minesweeper*.**

**Algoritma Greedy ini digunakan pada saat pemilihan petak jika tidak dapat dipastikan posisi ranjau relatif terhadap petak angka yang terbuka.**

**Menurut penelitian, masalah *minesweeper* tergolong ke dalam masalah *NP-complete*. Hal ini dikarenakan masalah *minesweeper* tidak dapat diselesaikan dalam orde waktu polinomial.**

**Kata Kunci :** Minesweeper, BFS, Greedy, NP-complete.

## I. PENDAHULUAN

Pemecahan permainan *minesweeper* merupakan salah satu masalah yang termasuk ke dalam golongan masalah *NP-complete*. Bahkan *Clay Mathematics Institute of Chambridge*, Massachusetts (CMI) menggolongkannya ke dalam salah satu dari tujuh masalah millenium yang dikenal dengan nama *Prize Problems*. CMI akan menganugrahkan $1 juta bagi yang dapat memecahkan salah satu masalah tersebut terkait hal masalah P = NP.

Karena *minesweeper* termasuk ke dalam *NP-complete*, tidak ada algoritma yang dapat digunakan untuk memecahkan masalah tersebut dalam waktu polinomial.

Walaupun begitu, tetap ada langkah-langkah yang dapat digunakan untuk mencoba memecahkan masalah tersebut. Makalah ini mencoba meberikan salah satu alternatif penyelesaian permainan minesweeper dengan mengunakan algoritma BFS dengan optimasi pemilihan petak menggunakan algoritma Greedy

## II. MINESWEEPER

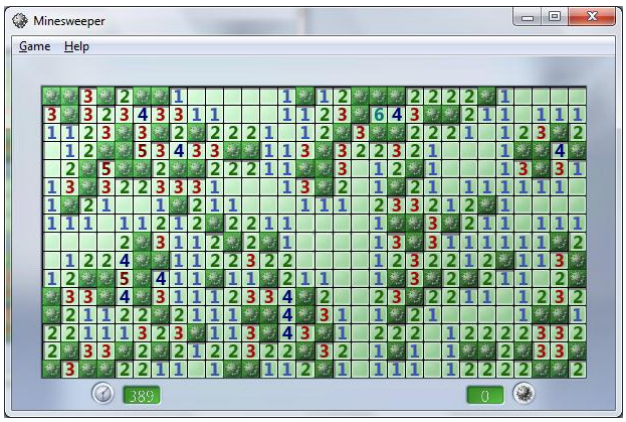
Pada awal permainan pemain dihadapkan dengan kumpulan petak atau kotak yang masih tertutup (belum bertanda apapun). Jumlah petak atau kotak menentukan skill yang dipilih pemain, tingkat skill tersebut tidak hanya menentukan banyaknya petak namun juga menentukan jumlah ranjau yang ada pada kumpulan petak tersebut.

Jika pemain mengklik salah satu petak yang bukan ranjau maka akan petak tersebut akan berisikan angka dan akan terbuka petak-petak yang berisikan angka disekitar petak yang diklik tersebut.

Angka pada petak tersebut menyatakan banyaknya ranjau disekitar petak tersebut, maksimal angka yang tertera pada petak adalah 8, karena sebuah petak hanya akan memiliki 8 petak tetangga.

Dengan menggunakan fasilitas yang terdapat pada minesweeper yaitu mengklik kedua tombol mouse pemain dapat menyimpulkan petak mana yang ranjau dan petak mana yang bukan ranjau, berdasarkan informasi pada petak yang berisi angka tersebut. Namun ada kalanya pemain harus menebak kira-kira petak mana yang bukan ranjau.

Pemain dapat menandai petak yang dipastikan ranjau dengan mengklik kanan tombol mouse, dan petak tersebut akan ditandai dengan gambar bendera, dan dapat menandai petak yang diduga ranjau dengan mengklik kanan tombol mouse 2 kali hingga muncul tanda (?). Hal ini dapat dilakukan pemain untuk memudahkan dalam bermain agar tidak melakukan kesalahan yang tidak penting yaitu mengklik petak yang jelas-jelas adalah ranjau.




**Gambar 1 Gambar Permainan *Minesweeper* tingkat *advance***

## III. DASAR TEORI

### *3.1 NP-Complete*

NP Problem adalah himpunan persoalan keputusan yang dapat diselesaikan oleh algoritma non-deterministik dalam waktu polinom.

Algoritma non-deterministik adalah algoritma yang berhadapan dengan beberapa opsi pilihan, dan algoritma memiliki kemampuan untuk menerka opsi pilihan yang tepat.

Kebanyakan persoalan keputusan adalah NP. Semua persoalan P juga adalah NP, sebab tahap menerka tidak terdapat di dalam persoalan P. Karena itu, P adalah himpunan bagian dari NP. Namun, belum ada yang bisa membuktikan apakah masalah P = NP atau P ≠ NP, sehingga ada banyak persoalan yang dapat di pecahkan secara mangkus dengan algoritma yang kebutuhan waktunya polinom.

Namun kenyataannya, banyak ahli yang telah gagal menemukan algoritma waktu-polinom untuk persoalan NP. Karena itu cukup aman jika kita menduga-duga bahwa P ≠ NP.

NP-Complete (NPC) adalah persoalan NP yang paling sulit. Sebuah persoalan X dikatakan NPC jika:

1. X termasuk ke dalam kelas NP
2. Setiap persoalan di dalam NP dapat direduksi dalam waktu polinom menjadi X.

### *3.2 Algoritma BFS*

Pada teori graf, BFS adalah algoritma pencarian pada graf yang dimulai dari simpul akar dan menelusuri seluruh simpul tetangganya. Kemudian untuk setiap simpul terdekat, algoritma ini menelusuri simpul tetangganya yang belum ditelusuri, dan seterusnya hingga menemukan solusi.

BFS adalah metode pencarian yang bertujuan untuk memperluas dan memeriksa semua simpul pada graf atau urutan kombinasi dengan pencarian secara sistematis melalui setiap solusi. Dengan kata lain, ia akan melakukan pencarian secara mendalam pada keseluruhan graf atau urutan tanpa memperhatikan tujuan hingga menemukan tujuan tersebut. Algoritma ini tidak menggunakan algoritma heuristik.\

Dari sudut pandang algoritma, semua simpul anak didapatkan dengan memperluas simpul yang ditambahkan pada *queue FIFO*.

Dengan metode BFS, simpul akar dibangkitkan pertama kali, kemudian simpul anak-anaknya, kemudia *successor* dari setiap simpul anak, dan seterusnya. Secara umum, semua simpul pada aras d dibangkit kan terlebih dahulu sebelum simpul-simpul pada aras d+1.

### *3.3 Algoritma Greedy*

Algoritma greedy membentuk solusi langkah per langkah (*step by step*). Terdapat banyak pilihan yang perlu dieksplorasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan terbaik dalam menentukan pilihan.

Pendekatan yang digunakan di dalam algoritma greedy adalah membuat pilihan yang "tampaknya" memberikan perolehan yang terbaik, yaitu dengan membuat pilihan optimum lokal pada setiap langkah dengan harapan bahwa sisanya mengarah ke solusi optimum global.

## IV. ANALISIS PERMAINAN MINESWEEPER

Terdapat banyak petak bernomor dengan pola tertentu yang mungkin ada selama permainan, pola tersebut hanyak memiliki satu kemungkinan peletakan ranjau di sekitar petak tersebut. Sehingga pasti ada petak yang benar-benar harus ditebak apakah ranjau atau bukan.

Karena dalam permainan ini waktu merupakan bagian dari permainan, selain dapat diselesaikan namun juga harus diselesaikan dengan waktu yang singkat. Ada beberapa analisis untuk memecahkan masalah tersebut.

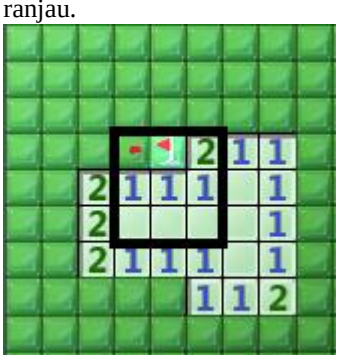
### *4.1 Analisis Single-Square*

Ada dua kasus khusus yang dapat dianalisis dengan metode ini:

- Jika jumlah petak *unlcliked* (*blank* atau memiliki tanda bendera) yang bertetangga sama dengan angka pada petak bernomor, maka semua petak *unclicked* tersebut adalah ranjau.

**Gambar 2 petak yang ditandai coretan merah adalah ranjau.**

- Untuk petak bernomor berapapun, jika jumlah ranjau yang telah ditemukan atau ditandai sama dengan angka pada petak maka, petak lain yang bertetangga dengan petak bernomor tersebut bukan ranjau.



**Gambar 3 petak dengan noda merah aman dibuka.**

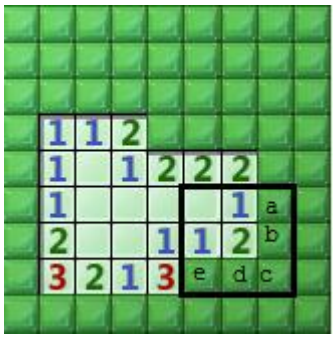
### *4.2 Analisis Multiple Square*

Untuk analisis yang lebih kompleks, pemain perlu mempertimbangkan lebih dari satu petak sekaligus. Beberapa strategi yang digunakan adalah :

- Jika terdapat dua angka bertetangga, selisih antara kedua angka tersebut sama dengan jumlah ranjau untuk tiga petak yang bertetangga dengan masing-masing petak yang tidak bertetangga dengan anka lain.
- Dengan metode yang sama, kadang-kadang tidak dapat diketahui bahwa terdapat sejumlah ranjau pada sejumlah petak dan informasi tersebut dapat digunakan untuk menemukan informasi mengenai petak lain.

Variabel yang tak dikenal adalah petak yang belum terbuka dan pembatasnya adalah petak tetangga yang telah terbuka. Algoritma pada analisa ini terdiri dari mencoba setiap kombinasi yang ranjau yang memenuhi setiap angka pada petak tetangga, dari hal tersebut dibuatlah kesimpulan.

Contoh :




**Gambar 4 Contoh analisis *multi-square***

Pada bagian yang diberi kotak hitam terdapat 3 petak bernomor yang bertetangga, maka dicoba setiap kombinasi dengan brute force dan menghasilkan 4 konfigurasi yang valid, {a,b,c,d,e} = {1,0,0,0,1}, {0,1,0,0,1}, {1,0,0,1,0} dan {0,1,0,1,0}, angka 1 menandakan ranjau.

Dari hasil tersebut dapat disimpulkan variabel c tidak pernah ranjau, sehingga petak c aman untuk dibuka.

## V. PENERAPAN ALGORITMA BFS DAN OPTIMASI PEMILIHAN PETAK DENGAN ALGORITMA GREEDY

Algoritma BFS ini digunakan untuk memecahkan masalah *minesweeper* yaitu dengan cara menelusuri seluruh petak yang memiliki tetangga ranjau yang dapat ditentukan dengan pasti. Untuk petak-petak yang tidak dapat dipastikan ranjau atau bukan digunakanlah algoritma greedy.

Algoritma BFS ini akan menelusuri petak pada papan secara rekursif sehingga petak-petak tidak dapat ditelusuri lagi. Papan akan membuka petaknya jika petak tersebut aman untuk dibuka dan menandai dengan bendera jika peta tersebut adalah ranjau.

Jika pada penelusurannya menemukan sebuah petak yang belum dapat dipastikan makan petak tersebut akan disimpan dalam sebuah queue yang nantinya akan diproses lebih lanjut.

Pseudo code Algoritma BFS untuk minesweeper

```
cekpetak(input pt : petak)
{
belumpasti ← false
If (pt <> petakkosong)then {basis}
  belumpasti ← cekSingleSquare(pt)
  if (not belumpasti)then
    insertqueue(pt)
  endif
else //cek petak tetangga
{rekurens}
```

```
  i traversal [1..8]
    pttemp ← pt(i)
    if (pttemp = belumdibuka
or
        pttemp =
petakbernomor)
        then
      cekpetak(pttemp)
    endif
endif
}
```

Fungsi `cekSingleSqure` adalah fungsi yang mengecek kepastian petak-petak tetangga dari petak yang dikunjungi. Fungsi ini menggunakan analisa *single-square*. Sedanggakn prosedur `insertQueue` adalah memasukkan petak ke dalam queue untuk dicek dengan algoritma greedy.

Untuk petak-petak yang ada dalam queue maka akan diproses dengan algoritma greedy untuk ditentukan apakah petak tersebut aman untuk dibuka atau tidak.

Algoritma greedy disini menghitung persentase kemungkinan petak itu ranjau berdasarkan analisis multi-squre yang telah dijelaskan diatas, kemudian dipilihlah petak yang memiliki persentase ranjau yang paling kecil.

Prosesnya yaitu menelusuri queue mulai dari head. Jika petak tersebut memiliki presentase yang paling kecil maka petak tersebut akan dibuka dan petak tersebut dihapus dari queue, jika petak head bukan petak yang memiliki persentase paling kecil maka petak tersebut dipindah ke tail.

Pseudo code Algoritma greedy

```
procedure TelusurPetakInQ()
{
bukanmin ← false
min ← persentasemin(Q)
n ← panjangQ()
selesai ← false
pertama ← true

while (not selesai)do
  headQ ← headQueue()
  if(min = headQ)then
    min ← persentasemin(Q)
  else
    headQ ← deleteQueue()
    addQueue(headQ)
    //penanda 1 siklus
    if(pertama)then
      patokan ← headQ
      pertama ← false
    endif
  endif
  if(isEmptyQueue() or
     (headQ = patokan and
      panjangQueue() =
n))then
    selesai ← true
  else if (headQ =
patokan)then
  {siklus baru}
    n ← panjangQueue()
    pertama ← true
  endif
endwhile
}
```

Fungsi persentasemin adalah fungsi yang mengembalikan petak dengan persentase minimun berdasarkan algoritma greedy, petak yang memiliki persentase minimum adalah petak yang kemungkinan petak tersebut ranjau adalah paling kecil berdasarkan analisis multi-square.

## VI. Analisis Solusi

Algoritma BFS pada persoalana ini digunakan sebagai metode penelusuran petak-petak. Algoritma ini dipilih karena sifat mebangkitkan seluruh kemungkinan sehingga proses penelusuran petak hingga petak yang tidak dapat dipastikan ranjau atau bukan dapat segera tercapai, dan dapat ter-*list* seluruh petak yang belum pasti ranjau.

Algoritma Greedy pada persoalan ini digunakan sebagai metode pemilihan petak mana yang aman untuk dibuka berdasarkan analisis *multi-square*.

Secara keseluruhan, solusi ini belum dapat menghasilkan hasil yang akurat. Karena ada saatnya program tidak dapat memperluas petak yang dapat ditelusuri sehingga program harus mengambil petak selanjutnya secara random. Akibatnya terdapat kemungkinan program salah mengambil petak dan petak yang diambil tersebut adalah ranjau.

Algoritma greedy disini juga belum optimal karena mungkin saja persentase yang terkecil petak bukan ranjau tersebut adalah salah.

Seperti yang sudah dijelaskan sebelumnya, permainan minesweeper ini termasuk masalah NP-complete dan pemecahan yang telah dicoba ini belum dapat memberikan solusi polinomial. Hal ini dapat dilihat dari kompleksitas algoritma BFS sendiri.

## VII. Kesimpulan

Algoritma BFS dengan optimasi Algoritma greedy dapat digunakan sebagai bagian dari pemecahan masalah minesweeper. Secara keseluruhan, solusi yang ditawarkan belum menjadi solusi yang akurat dan belum dapat memberikan waktu pemecahan polinomial.

## REFERENSI


[1] http://en.wikipedia.org/wiki/Breadth-first_search

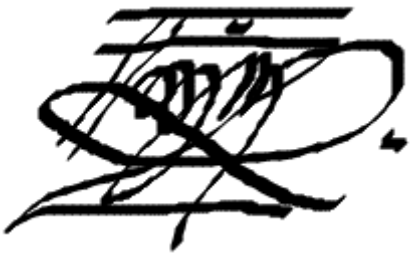
[2] http://en.wikipedia.org/wiki/Minesweeper_%28computer_game%29

[3] Munir, Rinaldi. *Diktat Kuliah IF3051 Startegi Algoritma.* Program Studi Teknik Informatika ITB 2009


## 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, 8 Desember 2010



Erdiansyah Fajar Nugraha
(13508055)