Aplikasi Algoritma Greedy pada Permainan *Pixelated*

Rachmat Arifin - 13508070
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18070@students.if.itb.ac.id

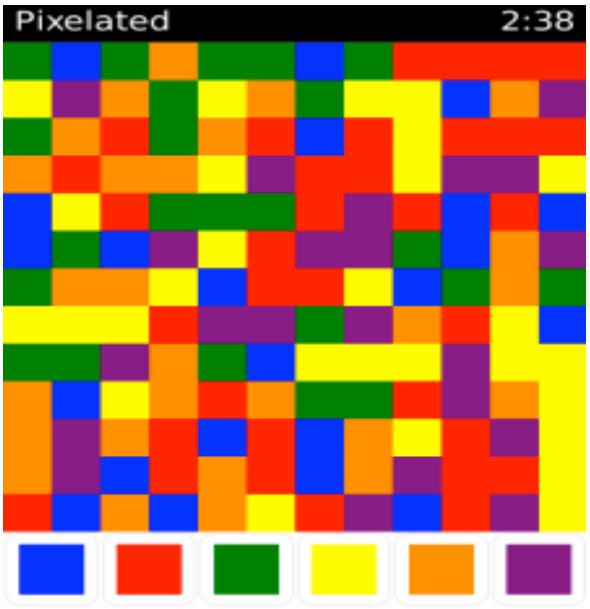
Abstrak—Pixelated merupakan salah satu permainan berbasis mobile yang berjalan pada platform BB(Black Berry). Pixelated merupakan permainan yang bertujuan membuat sejumlah pixel yang terdiri dari berbagai warna menjadi seragam. Permainan selalu dimulai dari Pixel paling kiri atas dan ketika pixel ini merubah warnanya, seluruh pixel yang bersinggungan dan memiliki warna yang sama akan menjadi bagian yang sama dengan pixel tersebut. Hal ini terus dilakukan sehingga seluruh layar menjadi satu warna atau pemain telah melakukan 22 kali gerakan. Apabila seluruh layar berhasil menjadi satu warna maka pemain dinyatakan menang, tetapi apabila sampai 22 kali gerakan seluruh layar belum memiliki warna yang seragam maka pemain dinyatakan kalah. Dengan menerapkan algoritma greedy diharapkan permainan dapat mencapai hasil optimalnya.

Kata kunci : *Pixelated*, Algoritma greedy, warna, seragam

I. PENDAHULUAN

Aplikasi berbasis *mobile* merupakan salah satu basis pemrograman yang sedang cukup diminati. BB-OS (*BlackBerry Operating System*) merupakan salah satu platform untuk aplikasi *mobile* yang dikembangkan oleh RIM (*Research In Motion*). RIM menyediakan fitur *AppWorld* pada produk – produknya yang merupakan fitur untuk mengunduh aplikasi yang berbasis BB-OS. Salah satu aplikasi permainan yang terdapat pada fitur tersebut adalah *Pixelated*. *Pixelated* merupakan aplikasi *freeware* yang dikembangkan oleh *vendor Ebscer*.

Objective atau tujuan utama dari permainan ini adalah membuat seluruh layar menjadi satu warna dengan jumlah langkah tidak lebih dari niali yang telah ditetapkan yaitu 22 langkah. Permainan dimulai dari bagian -dalam konteks ini disebut sebagai *Pixel*- paling kiri atas. Setelah itu pemain diberikan 6 pilihan warna untuk merubah *pixel* tersebut menjadi warna yang dikehendakinya. Setelah *pixel* berubah menjadi warna yang dipilih, seluruh *pixel* yang bersinggungan dengan *pixel* tersebut dan memiliki warna yang sama setelah *pixel* berubah warna akan menjadi bagian dari *pixel* tersebut.



Gambar 1 Kondisi awal permainan *Pixelated*

Tantangan dalam permainan ini adalah bagaimana melakukan langkah yang baik sehingga seluruh layar dapt menjadi satu warna dengan jumlah langkah kurang dari 21 kali.

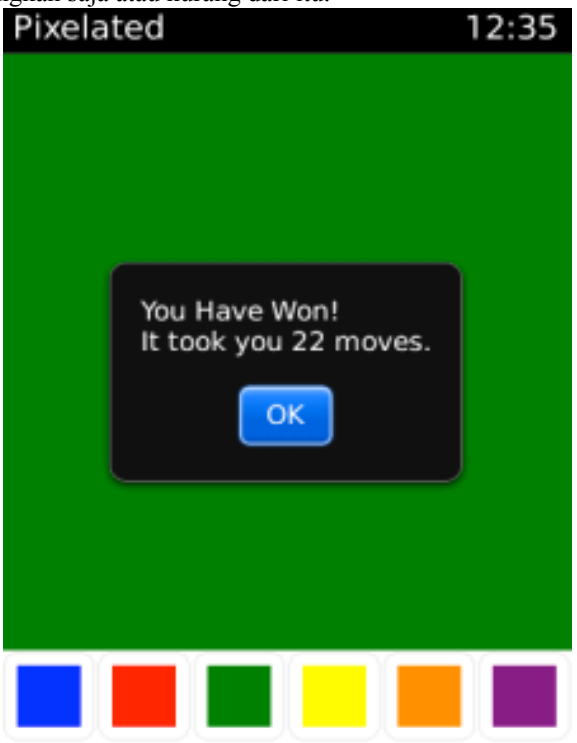
Ketika semakin banyak warna yang bisa diambil menjadi bagian dari *pixel* utama, akan semakin baik karena semakin besar area yang dimiliki. Hala ini pun akan memberikan keuntungan lebih besar dan mengoptimalkan setiap langkah yang digunakan.

Salah satu solusi yang dapat digunakan untuk memecahkan permasalahan ini adalah dengan menerapkan algoritma Greedy. Salah satu dasar dalam bermain permainan ini adalah mengambil warna yang memiliki jumlah *pixel* yang bersinggungan dengan warna yang sama paling banyak. Konsep bermain seperti memiliki prinsi yang sama dengan algoritma Greedy, yaitu mengambil nilai maksimum atau minimum dari sejumlah nilai – nilai lokal.

Gambar 2 Help screen dari *pixelated*

II. DESKRIPSI PERMASALAHAN

Permainan ini merupakan permainan yang biasa saja ketika yang dituju hanyalah membuat seluruh layar memiliki warna yang sama, tetapi akan berbeda ketika hanya 22 langkah saja yang dapat digunakan pemain untuk memenangkan permainan ini. Setiap langkah yang diambil memerlukan petimbangan yang baik agar permainan ini dapat diselesaikan hanya dengan 22 langkah saja atau kurang dari itu.



Gambar 3 Kondisi ketika pemain menang

Ketika sebuah warna dipilih seluruh warna yang sama dan bersinggungan dengan *pixel* tersebut akan menjadi bagian dari *pixel* tersebut. Apabila tidak warna yang sama dengan pilihan dari pemain maka tidak aka nada perubahan apapun pada *pixel* yang ada tetapi jumlah langkah yang digunakan tetap akan bertambah.

Permainan ini merupakan permainan yang menarik, karena apabila dikaji lebih dalam banyak cara – cara atau strategi yang bisa digunakan untuk menyelesaikan permasalahan ini. Pada awalnya mungkin orang akan mencoba mengambil jumlah pixel terbanyak yang bersinggungan dengan pixel pemain. Tetapi, apakah memilih jumlah terbnayak adalah pilihan terbaik? Apakah ketika memilih pilihan yang paling banyak akan memberikan hasil yang optimal? Atau apakah dengan mengambil memilih warna yang terbanyak pasti akan memberikan kemenangan bagi pemain? Hal tersebut masih belum dapat dibuktikan.

III. ALGORITMA GREEDY

Selalu mengambil nilai maksimum atau minimum merupakan sifat dari algoritma greedy. Dengan menggunakan algoritma greedy, atau dapat dikatakan dengan mengambil nilai paling besar atau kecil, diharapkan dapat memberikan hasil yang paling optimal, walaupun pada beberapa kasus hal tersebut tidak selalu terpenuhi.

Berikut ini adalah salah satu contoh skema algoritma Greedy

```
function greedy(input C: himpunan_kandidat)→ himpunan_kandidat
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy
  Masukan: himpunan kandidat C
  Keluaran: himpunan solusi yang bertipe himpunan_kandidat
}
Deklarasi
    x : kandidat
    S : himpunan_kandidat

Algoritma:
    S ← {}   { inisialisasi S dengan kosong }
    while (not SOLUSI(S)) and (C ≠ {} ) do
      x ← SELEKSI(C)        { pilih sebuah kandidat dari C}
      C ← C - {x}           { elemen himpunan kandidat berkurang satu }
      if LAYAK(S ∪ {x}) then
         S ← S ∪ {x}
      endif
    endwhile
    {SOLUSI(S) or C = {} }

    if SOLUSI(S) then
      return S
    else
      write('tidak ada solusi')
    endif
```

Gambar 4 Algoritma Greedy

Algoritma Greedy merupakan strategi algoritma yang menentukan dari setiap langkahnya, tidak menentukan solusi secara keseluruhan permasalahan. Hal inilah yang

membuat strategi ini tidak akan selalu memberikan hasil yang optimum ketika diimplementasikan. Akan tetapi algoritma greedy merupakan salah satu pilihan yang sering digunakan orang karena *simplicities* yang dimilikinya. Algoritma Greedy memiliki beberapa elemen yang terdapat di dalamnya, antara lain :

1. Himpunan kandidat
2. Himpunan solusi
3. Fungsi seleksi
4. Fungsi kelayakan
5. Fungsi obyektif

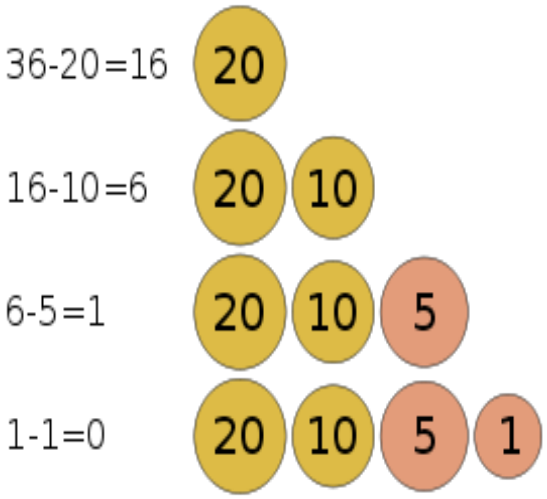
Dapat dikatakan bahwa algoritma Greedy melibatkan pencarian sejumlah himpunan bagian dari himpunan kandidat yang dalam hal ini harus memenuhi beberapa kriteria yang telah ditentukan.

Keoptimalan hasil dari algoritma greedy, seperti disebutkan sebelumnya, belum terjamin. Pada beberapa kasus algoritma greedy akan menghasilkan hasil yang tidak optimal sementara hasil pada kasus lain akan memberikan nilai yang optimum. Ketika algoritma greedy memeberikan nilai yang optimal maka keoptimalan dari nilai tersebut dapat dibuktikan secara matematis.

Terdapat 2 properti penting yang menjadi permasalahan ketika algorima greedy tidak dapat memberikan nilai optimalnya, yaitu *greedy choice property* dan *optimal substructure.*

Greedy choice property memperlihatkan bahwa algoritma greedy hanya mengambil keputusan berdasarkan fakta yang ada sekarang tanpa adanya pertimbangan lebih lanjut mengenai fakta – fakta yang mungkin muncul pada langkah – langkah berikutnya. Dapat dikatakan pula bahwa algoritma greedy tidak pernah melakukan pertimbangan ulang terhadap pilihannya.

Hal lain yang bisa dianalisis dari algoritma ini adlaah *optimal substructure.* Algoritma greedy memberikan solusi pada permasalahan lokal (*substructure*). Apabila keputusan telah diambil dan langkah – langkah selanjutnya memberikan nilai yang optimal maka hal ini disebut sebagai *optimal substructure.* Kebalikannya, ketika pengambilan nilai optimal pada suatu permasalahan membuat nilai yang akan diambil pada langkah – langkah berikutnya tidak optimal secara global, maka hal ini disebut sebagai *non-optimal substructure.*



Gambar 5 Ilustrasi permasalahan penukaran koin

III. IMPLEMENTASI

Seperti telah disebutkan sebelumnya bahwa permainan *pixelated* dapat disimulasikan dengan algoritma greedy dan mungkin akan memberikan hasil yang optimal.

Dengan menggunakan algoritma greedy, permainan *pixelated* akan selalu mencari jumlah *pixel* dengan warna yang seragan dengan jumlah paling banyak. Langkah tersebut terus berulang hingga seluruh *pixel* pada layar memiliki warna yang sama.

Ketika permainan dimulai *pixel* paling kiri atas adalah *pixel* awal dari permainan. Kemudian informasi dari seluruh pixel yang bersinggungan langsung dengan *pixel* tersebut dikumpulkan. Setelah semua informasi warna telah terkumpul, lalu diputuskan warna apa yang akan dipilih berdasarkan jumlahnya. Jumlah warna yang paling banyak lah yang akan dipilih berdasarkan algoritma greedy ini. *Pixel* dengan warna yang sama pun makin banyak. Langkah tadi pun diulangi lagi dengan melakukan pengecekan terhadap seluruh *pixel* yang bersingungan. Langkah – langkah tersebut terus diulangi hingga seluruh *pixel* pada layar menjadi satu warna. Apabila jumlah langkah yang dibutuhkan bernilai 22 langkah atau kurang dari nilai tersebut maka algoritma greedy merupakan salah satu strategi yang dapat digunakan untuk menyelesaikan permasalahan ini. Akan tetapi, hal tersebut belum dapat membuktikan bahwa algoritma greedy memberikan hasil yang paling baik jika dibandingkan dengan algoritma atau strategi yang lain.

Tetapi, permasalahan yang memang telah diperkirakan sebelumnya dapat terjadi yang akan berakibat pada hasil yang tidak akan optimal. Salah satu contoh kasu dapat diilustrasikan seperti berikut :

1. Pilihan 1
    a. Hijau : 2
    b. Biru : 3
2. Pilihan a
    a. Merah : 3
    b. Biru : 5
3. Pilihan b

a. Kuning : 1
b. Biru : 1
c. Kuning : 3

Ketika pada pilihan 1, pilihan akan jatuh pada warna biru, yang berarti jumlah *pixel* yang telah dimiliki ada 4. Saat masuk pada langkah kedua pilihan akan jatuh pada warna kuning dan jumlah *pixel* sekarang ada 7. Apabila pada pilihan 1 warna hijau yang diambil, lalu mengambil warna biru pad apilihan keduanya jumlah *pixel* yang ada mencapai 8 buah.

Hal inilah yang menjadi kekurangan dari algoritma greedy. Algoritma greedy hanya mengambil keputusan berdasarkan informasi saat itu saja, tanpa memperhatikan kondisi disaat langkah – langkah berikutnya.

IV. HASIL PERCOBAAN

Berikut ini adalah hasil percobaan dari implementasi algoritma greedy pada permainan *Pixelated.*

Percobaan No.	Jumlah Langkah	Keterangan
1.	22	Berhasil
2.	23	Gagal
3.	27	Gagal
4.	28	Gagal
5.	28	Gagal
6.	21	Berhasil
7.	24	Gagal
8.	20	Berhasil
9.	29	Gagal
10.	29	Gagal
11.	23	Gagal
12.	22	Berhasil
13.	21	Berhasil
14.	25	Gagal
15.	30	Gagal
16.	22	Berhasil
17.	21	Berhasil
18.	27	Gagal
19.	25	Gagal
20.	31	Gagal

Tabel 1 Tabel hasil percobaan

Dari 20 kali percobaan didapat 7 kali percobaan yang berhasil dan 13 kali percobaan yang gagal. Nilai dari langkah pun bervariasi dalam rentang 20 – 31 langkah.

V. ANALISIS

Dari hasil 20 kali percobaan didapat bahwa dengan mengimplementasikan algoritma greedy tidak selalu membuat pemain memenangkan permainan ini.

Dengan jumlah 7 kali berhasil dan 13 kali gagal berarti kemungkinan keberhasilan strategi algoritma greedy untuk diimplementasikan pada permainan *pixelated* adalah 35% dan kemungkinan kegagalnnya adalah 65%. Walaupun algoritma greedy dapat menyelesaikan permasalahn pada permainan ini, nilai tersebut menunjukkan bahwa algoritma greedy sebenarnya tidak efektif untuk diimplementasikan pada permainan ini.

Jika dilihat pada table, ada 7 kali kebehasilan dari hasil tersebut. Rentang nilai keberhasilan tersebut adalah 20 – 22. Saat jumlah langkah adalah 22, berarti algoritma greedy berhasil menyelesaikan permainan pada ambang batasnya. Sama halnya pada keberhasilan dengan jumlah langkah 21 dan 20. Keberhasilan dengan 22 langkah berjumlah 3 kali, 21 langkah 3 kali dan 1 kali dengan jumlah langkah 20. Hal ini menunjukkan walaupun algoritma greedy terkadang berhasil menyelesaikan memberikan nilai 22 atau kurang, algoritma greedy masih sangat riskan untuk diimplementasikan pada permainan ini karena kemungkinan keberhasilannya yang masih sangat kecil. Kemungkinan jumlah langkah 20 adalah 5%, 21 langkah 15% dan 22 langkah adalah 15%.

Dari analisis diatas dapat dilihat bahwa secara umum algoritma greedy tidak dapat menyelesaikan permasalahanpada permainan *pixelated.*

Salah satu solusi yang dapat penulis tawarkan adalah DFS dan backtracking. Algoritma greedy hanya mampu mengambil keputusan berdasarkan info dan fakta yang ada pada saat itu saja. Algoritma greedy belum mampu memperkirakan fakta yang mungkin akan nada nantinya. Hal inilah sebenernya yang menjadi permasalahan implementasi algoritma greedy. Dengan menggunakan algoritma DFS dan backtracking semua informasi mengenai kemungkinan langkah yang ada dapat dikumpulkan. Sehingga langkah – langkah dengan jumlah paling sedikit dapat ditemukan.

Dengan menggunakan strategi algoritma DFS serta backtracking, diharapkan dapat memberikan hasil yang lebih baik sehingga dapat memberikan hasil yang lebih optimal dibandingkan dengan algoritma greedy.

VI. CONCLUSION

Algoritma greedy merupakan strategi yang digunaan dengan konsep memilih sesuatu yang memiliki nilai maksismum atau minimum, bergantung pada *objective*. Algoritma greedy dapat diimplementasikan pada permainan *pixelated* yang merupakan permainan berbasis *mobile* pada *platform BlackBerry Operating System*.

Setelah dilakukan percobaan dan analisis algoritma greedy memiliki nilai probabilitas sebanyak 35% untuk keberhasilannya.

Dapat disimpulkan bahwa implementasi algoritma greedy untuk penyelesaian permainan *pixelated* tidaklah efektif karena nilai kemungkinan keberhasilanya lebih kecil disbanding nilai kemungkinan kegagalannya.

Penulis menyarankan untuk mengimplementasikan algoritma DFS serta backtracking untuk menyelesaikan perasalahan pada permainan ini.

REFERENCES

[1] Cormen, Leiserson, and Rivest,"Introduction to Algorithms", Chapter 16 pp. 329

[2] http://www.informatika.org/~rinaldi/Stmik/2010-2011/stmik10-11.htm#SlideKuliah

[3] http://www.personal.kent.edu/~rmuhamma/Algorithms/MyAlgorithms/Greedy/greedyIntro.htm

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, 7 Desember 2010

Rachmat Arifin – 13508070