Pemanfaatan Strategi Greedy untuk Meminimalkan Jumlah Pembalikan Kartu pada Permainan Konsentrasi

Jan Meyer Saragih / 13517131
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13517131@std.stei.itb.ac.id

***Abstrak*— Algoritma *greedy* merupakan salah satu strategi algoritma yang diajarkan pada mata kuliah IF2211 – Strategi Algoritma, ITB. Pada kesempatan ini, penulis menggunakan strategi ini untuk menyelesaikan permainan konsenterasi, salah satu permainan yang menggunakan kartu remi dengan harapan meminimalkan jumlah pembalikan kartu pada permainan tersebut.**

***Kata Kunci*—strategi, algoritma, kartu, *deck*, lambang, nomor, *greedy*, kandidat, solusi, seleksi, kelayakan, obyektif.**

I. PENDAHULUAN

Kartu remi merupakan kartu yang sering digunakan sebagai permainan sampingan di tengah permainan-permainan digital di era modern ini. Kartu ini diyakini berasal dari Tiongkok, lalu menyebar ke India, Persia, dan seluruh belahan dunia. Sifat kartu ini yang sederhana, murah, mudah dibawa kemana-mana, dan dapat ditemukan di mana saja merupakan alasan kartu ini menjadi alasan kartu ini sering dijadikan permainan di waktu lkoin.

Permainan-permainan yang dapat dimainkan dari kartu remi berjumlah banyak. Mulai dari capsa, tepuk nyamuk, cangkulan yang hanya membutuhkan reflex dan keberuntungan saja, menuju ke permainan bohong, 41, fish, dan konsenterasi, yang membutuhkan insting dan ingatan, hingga ke permianan *blackjack*, *poker*, *solitaire* yang populer di tempat-tempat ternama.

Makalah ini akan membahas pemanfaatan strategi *greedy* untuk meminimalkan jumlah pembalikan kartu pada permainan konsenterasi. Permainan konsenterasi merupakan sebuah permainan ingatan yang menggunakan kartu remi di mana pemain mengacak ke-52 kartu teresbut. Setelah diacak, kartu-kartu tersebut diletakkan di atas meja secara tertutup. Dengan demikian, pemain tidak mengetahui lambang dan nomor kartu yang diletakkan di atas meja.

Permainan konsenterasi dibagi menjadi giliran-giliran. Pada setiap giliran, pemain membalik 2 kartu dan jika nomor pada kedua kartu tersebut sama, maka pemain dapat menarik kedua kartu tersebut dari daerah permainan, sehingga jumlah partu dalam permainan berkurang 2. Jika tidak sama, maka pemain mengembalikan kedua kartu tersebut ke tempatnya semula. Permainan akan terus berlanjut hingga tidak terdapat kartu pada daerah permainan. Semakin sedikit jumlah giliran yang harus dilewati untuk memenangkan pertandingan, semakin baik.

Jumlah giliran yang harus dilewati sebelum memenangkan pertandingan dapat diminimalkan dengan memanfaatkan strategi *greedy*. Strategi *greedy* merupakan salah satu strategi algoritma dari strategi-strategi algoritma yang diajarkan pada mata kuliah IF2211-Strategi Algoritma, Institut Teknologi Bandung.

Strategi *greedy* merupakan strategi yang populer untuk persoalan optimasi, yang mencakup maksimasi dan minimasi. Prinsip yang digunakan pada strategi *greedy* adalah *take what you can get now*. Strategi *greedy* mempunyai kompleksitas yang rendah sehingga diharapkan dengan menggunakan strategi ini, dapat dihasilkan solusi yang memiliki kompleksitas yang rendah namun tetap menghasilkan jawaban yang optimal. Kompleksitas *greedy* yang rendah yang memungkinkan untuk meminimalkan jumlah pembalikan kartu untuk menyelesaikan permainan dengan hasil optimum.

II. DASAR TEORI

A. Teori Strategi Greedy

Strategi *greedy* merupakan salah satu strategi penyelesaian masalah yang diajarkan pada mata kuliah IF2211 – Strategi Algoritma di Institut Teknologi Bandung.

Strategi *greedy*,populer dalam menyelesaikan masalah optimasi, seperti maksimasi dan minimasi. Dalam kasus ini, strategi ini digunakan untuk membantu dalam hal maksimasi. Seperti namanya, *greedy*, yang artinya tamak, strategi ini memiliki prinsip "*take what you can get now*".

Strategi *greedy* terbagi menjadi langkah-langkah di mana di setiap langkahnya, seperti prinsip dari strategi ini, mencoba mencari nilai optimum dari langkah tersebut, dalam kasus ini maksimum (jumlah pasangan kartu). Dengan demikian, strategi ini terus mencari nilai optimum dari setiap langkah-langkah yang dilewati, sesuai dengan prinsip "*take what you can get now*", dengan harapan nilai-nilai optimum yang diperoleh dari setiap langkah tersebut (optimum lokal) nanti akan menghasilkan nilai optimum pada akhirnya (optimum global).

Strategi *greedy* terbagi menjadi elmeen-elemen. Elemen-elemen tersebut adalah:

1. Himpunan kandidat (C).

   Merupakan himpunan yang berisi solusi-solusi yang mungkin menjadi solusi dan belum diseleksi menggunakan fungsi seleksi menjadi himpunan solusi.

2. Humpunan solusi (S).

   Hasil dari himpunan kandidat yang telah terseleksi menggunakan fungsi seleksi. Jumlah anggota dari himpunan solusi berjumlah lebih sedikit atau sama dengan jumlah anggota dari himpunan kandidat

3. Fungsi seleksi (*selection function*)

   Fungsi yang menyeleksi anggota-anggota dari himpunan kandidat. Dengan fungsi ini, akan diambil kandidat untuk diproses dan dimasukkan ke himpunan solusi.

4. Fungsi kelayakan (*feasible*)

   Fungsi yang digunakan untuk mengecek nilai dari himpunan kandidat yang tidak melebihi batasan-batasan yang ada sebelum dimasukkan ke dalam himpunan solusi.

5. Fungsi obyektif

   Fungsi yang menghasilkan solusi dari permasalahan tersebut.

Dibandingkan dengan strategi algoritma lain, strategi algoritma *greedy* merupakan salah satu strategi dengan kompleksitas yang rendah. Hal ini disebabkan strategi *greedy* tidak mengecek semua kemungkinan dari pasangan yang ada, namun lebih ke membagi penyelesaian ke langkah-langkah kecil, yang lebih sederhana dan mencari nilai optimum dari langkah-langkah tersebut.

Namun di saat yang sama, strategi *greedy* ini juga tidak menjamin bahwa hasil yang didapatkan dari optimum-optimum lokal tersebut pada akhirnya akan menghasilkan optimum global.

Sebagai contoh pada masalah penukaran koin. Persoalan penukaran koin adalah persoalan yang menentukan pecahan-pecahan koin yang dibutuhkan untuk merepresentasikan suatu jumlah koin. Hasil yang optimum adalah hasil dengan jumlah pecahan koin yang paling sedikit.

Strategi *greedy* yang digunakan untuk menyelesaikan permasalahan tersebut adalah dengan mengurutkan himpunan koin secara menurun.

Setelah terbentuk himpunan koin secara menurun (sebagai himpunan kandidat), lakukan pengulangan sebagai berikut:

1. Cek jumlah nilai himpunan kandidat dan total nilai yang terdapat dalam himpunan solusi.
2. Jika belum melebih jumlah koin, maka tambahkan himpunan kandidat tersebut ke dalam himpunan solusi.
3. Jika sudah melebihi jumlah koin, maka hapuskan dari himpunan kandidat.
4. Lakukan terus-menerus hingga himpunan kandidat habis dan solusinya terdapat dalam himpunan solusi.

```
function CoinExchange(input C : himpunan_koin, A : integer) → himpunan_koin
{ mengembalikan koin-koin yang total nilainya = A, tetapi jumlah koinnya minimum }

Deklarasi
   S : himpunan_koin
   x : koin

Algoritma
  S ← {}
  while (∑(nilai semua koin di dalam S) ≠ A) and (C ≠ {} ) do
     x ← koin yang mempunyai nilai terbesar
     C ← C - {x}
     if  (∑(nilai semua koin di dalam S) + nilai koin x ≤ A then
        S ← S ∪ {x}
     endif
  endwhile

  if (∑(nilai semua koin di dalam S) = A then
     return S
  else
     write('tidak ada solusi')
  endif
```

Gambar I. Algoritma greedy pada penukaran koin

Sumber: http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Algoritma-Greedy-(2019).pdf

Sebagai contoh *test case*, jika terdapat pecahan koin 5, 4, dan 1, maka hasil dari penukaran koin dengan jumlah 8 adalah 5 + 1 + 1 + 1 (memiliki 4 buah pecahan koin). Hal ini bukan merupakan solsui optimum karena solusi dari optimum dari pecahan koin tersebut adalah solusi 4 + 4 (memiliki 2 buah pecahan koin).

Maka dari itu, dapat dipahami bahwa kelemahan dari strategi greedy adalah solusi yang dihasilkan belum tentu solusi yang paling optimum.

Dari data-data di atas, dapat disimpulkan bahwa kelebihan dari strategi *greedy* adalah kompleksitasnya yang rendah dibanding strategi lain sementara kelemahan dari strategi *greedy* adalah strategi ini tidak selalu menghasilkan hasil yang optimum.

III. PENYELESAIAN PERMASALAHAN

A. . Pembahasan Kartu Remi

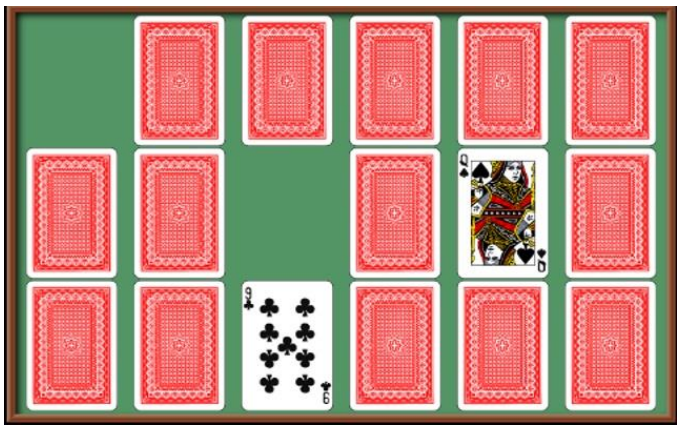
Kartu remi merupakan alat permainan yang sering digunakan untuk permainan-permainan yang ada di dunia ini. Permainan yang berasal dari Tiongkok ini awalnya masih belum berbentuk kartu. Permainan ini pertama kali mendapatkan bentuknya sebagai kartu sekitar tahun 1470 dengan nama kartu Tarot. Bentuk kartu dalam permainan ini terus berkembang hingga pada akhirnya terbentuklah kartu remi seperti yang kita lihat pada zaman ini.

Satu *deck* kartu remi teridiri dari 52 kartu. Ke-52 kartu tersebut terbagi ke dalam 2 warna kartu, yaitu hitam dan merah. Kedua warna kartu tersebut mencakup 4 lambang kartu, yaitu sekop (*spade*), hati (*heart*), wajik (*diamond*), dan keriting (*club*). Setiap lambang kartu, memiliki 13 kartu dengan nomor kartu yang berbeda. Cara penomoran kartu untuk setiap lambang kartu adalah sama. 13 nomor kartu tersebut adalah As, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack (J), Queen (Q), dan King (K). Selain ke-52 kartu tersebut, satu *deck* kartu remi juga mempunyai tambahan kartu dengan penambahan nomor kartu Joker pada beberapa jenis *deck*.

Pada makalah ini, *deck* kartu yang akan dibahas adalah *deck* kartu standar dengan 52 kartu, sehingga tidak terdapat kartu dengan nomor Joker dalam permainan.

B. Pembahasan Permainan Konsenterasi

Permainan konsenterasi, seperti yang telah disebutkan sebelumnya, merupakan permainan yang memanfaatkan kartu remi. Pada awalnya, pemain mengacak kartu-kartu pada kartu remi tersebut dan meletakkannya secara tertutup di atas daerah permainan, biasanya di atas sebuah meja. Setelah mengacak kartu dan meletakkan kartu-kartu tersebut, permainan konsenterasi dapat dimulai.



Gambar II. Contoh permainan konsenterasi dalam komputer

Sumber:https://chrome.google.com/webstore/detail/concentration/fpoeeaddlpbhdbaapjnpeipdbahiajaj

Satu permainan dibagi menjadi beberapa giliran di mana setiap giliran pemain membuka 2 kartu. Jika kedua kartu tersebut memiliki nomor yang sama, (sebagai contoh, As Hati dengan As Keriting), maka pasangan kartu tersebut dapat dipindahkan keluar dari daerah permainan. Hal ini terus berlanjut hingga tidak terdapat kartu lagi yang berada di daerah permainan tersebut. Hal ini mungkin karena jumlah kartu yang terdapat dalam daerah permainan adalah genap.

Permainan ini dapat dimainkan oleh satu atau lebih orang. Jika dimainkan oleh satu orang, maka tujuan dari permainan ini adalah untuk mendapatkan giliran sesedikit-sedikitnya dalam membalik kartu. Jika permainan ini dimainkan oleh lebih dari satu orang, maka tujuan dari permainan ini adalah mendapatkan jumlah pasangan kartu yang paling banyak.

Selain itu, terdapat banyak varian dari permainan ini. Varian-varian yang terdapat dalam permainan konsenterasi adalah sebagai berikut:

1. *Any color*

   Pasangan dengan warna kartu yang sama dapat dikeluarkan

2. *One flip*

   Pemain yang berhasil mendapatkan kartu pasangan akan mendapatkan kartu yang dikeluarkan dari daerah permainan. Namun, pemain tersebut tidak mendapatkan kesempatan untuk membalik kartu lagi).

3. *Zebra*

   Pasangan kartu yang dimungkinkan untuk dikeluarkan dari daerah permainan merupakan pasangan kartu yang memiliki nomor yang sama, namun memiliki warna kartu yang berbeda. Sebagai contoh, 8 hati dapat dikeluarkan dengan 8 sekop dan keriting namun tidak dapat dipasangkan dengan 8 hati.

4. *Two decks*

   Hal ini merupakan kemungkinan lain jika pemain menginginkan jumlah kartu yang lebih banyak. Maka dari itu, dua *deck* kartu digabungkan dan diacak dan ditebar dalam papan permainan.

5. *Missing Cards*

   Hal ini dapat dilakukan pada awal permainan. Pemain dapat mengambil kartu terurut secara acak dan tidak melihat kartu tersebut. Kartu-kartu tersebut dianggap meningkatkan elemen *surprise* pada permainan ini karena pemain tidak mengetahui kartu mana yang tidak memiliki pasangan akibat diambil keluar dari game.

Makalah ini akan membahas permasalahan untuk meminimalkan jumlah giliran tersebut, sehingga lebih ke membahas permainan jika dimainkan oleh satu orang. Sementara itu, varian permainan yang diambil untuk dibahas dalam makalah ini merupakan varian *any color, one flip,* dan *missing cards.*

C. Pengimplementasian Strategi Greedy pada Permasalahan

Strategi *greedy* adalah strategi yang digunakan untuk menyelesaikan permasalahan ini. Seperti yang dijelaskan sebelumnya, strategi algoritma ini bersifat tamak/loba karena berusaha untuk mendapatkan hasil yang optimum pada setiap langkah yang diproses.

Seperti yang dijelaskan pada bagian sebelumnya, dalam makalah ini, permainan yang akan dibahas merupakan permainan konsentrasi yang dimainkan oleh satu orang dengan varian-varian berupa *any color, one flip,* dan *missing cards.*

Sebelum menjalankan strategi *greedy* tersebut pada permasalahan ini, diperlukan untuk menjelaskan elemen-elemen strategi *greedy* yang digunakan pada persoalan ini.

1. Himpunan kandidat (C)

   Himpunan kandidat ini melambangkan semua elemen *deck* dari permainan yang belum dibuka oleh pemain.

2. Himpunan solusi (S)

   Himpunan solusi merupakan kartu-kartu yang memiliki pasangan dan telah terdeteksi oleh sistem.

3. Fungsi seleksi

   Fungsi seleksi ini mengambil 2 elemen teratas atau paling awal dari kartu-kartu yang telah disebarkan dan ditutup.

4. Fungsi kelayakan

   Fungsi kelayakan merupakan fungsi yang mengecek apakah elemen yang mau ditambahkan berpasangan atau tidak.

5. Fungsi obyektif

   Fungsi obyektif menyatakan terdapat pasangan kartu.

Strategi *greedy*, seperti yang dijelaskan sebelumnya, terbagi menjadi langkah-langkah. Langkah-langkah yang diperlukan untuk menyelesaikan masalah ini adalah sebagai berikut:

1. Siapkan sebuah *deck* (himpunan kandidat) yang sudah teracak dan telah diambil kartunya secara acak.
2. Siapkan list reminder yang berisikan list yang tidak ditemukan pasangan dan list solution (himpunan solusi) yang berisi solusi dari permasalahan tersebut.

```
# Mendata jumlah giliran pembalikan kartu
flipCount = 0
# Membuat elemen untuk mendata
reminder = [[],[],[],[],[],[],[],[],[],[],[],[],[]]
solution = []
```

3. Selama fungsi kandidat belum berjumlah paling banyak 1, lakukan kegiatan berikut:
   a. Balikkan kedua elemen pertama dari himpunan kandidat. Hal ini merupakan mekanisme dari permainan konsenterasi yaitu hanya boleh ada 2 kartu yang dibalik. Setelah itu, tambahkan jumlah giliran.

```
# Fungsi SELEKSI yang mengambil kedua elemen dari kartu tersebut
flipCount += 1

cardA = cards[0]
cards.remove(cardA)
cardB = cards[0]
cards.remove(cardB)
```

   b. Jika kedua elemen dari himpunan kandidat tersebut memiliki nomor kartu yang sama, maka kedua elemen tersebut ditambahkan ke himpunan solusi. Hal ini melambangkan membalikkan dua kartu yang memiliki nomor kartu yang sama.

```
# Melakukan pengecekan nomro dari kedua kartu
# Jika ya, tambahkan solusi ke dalam list solusi dan tambahkan marker

if checkNumber(cardA, cardB):
  solution.append(cardA)
  solution.append(cardB)
```

   c. Jika kedua elemen dari elemen kartu tersebut tidak memiliki nilai yang sama, maka lakukan hal berikut untuk kedua elemen list tersebut.
      i. Cek apakah memiliki pasangannya yang telah diingat posisinya di dalam list reminder.
      ii. Jika iya, hilangkan kartu pada reminder tersebut dan tambahkan jumlah giliran. Hal ini melambangkan membuka 2 kartu lagi, yaitu satu kartu merupakan salah satu dari kedua elemen tersebut dan satu lagi kartu lain dengan nomor yang sama yang telah dibuka dan diingat posisi kartunya.
      iii. Jika tidak, tambahkan kartu tersebut pada list reminder. Hal ini melambangkan mengingat posisi dan nilai dari kartu tersebut.
4. Cek apakah masih terdapat sisa 1 elemen pada himpunan kandidat. Jika masih terdapat elemen yang tersisa, maka lakukan langkah-langkah yang terdapat pada bagian 3.c.

   Kode di bawah ini melambangkan pengecekan yang dilakukan oleh tahap ke-3.c dan tahap ke-4.

```
# Fungsi pengecekan nilai pada mapping
def checkReminder(reminder, card, solution):
   # Cek kartu tersebut
   # Jika terdapat kartu tersebut di dalam reminder
   if (reminder[mapping(card)]):
      solution.append(card)
      solution.append(reminder[mapping(card)][0])
      reminder[mapping(card)].pop(0)
      return True
   # Jika tidak, maka tambahkan kartu tersebut ke dalam reminder
   else:
      reminder[mapping(card)].append(card)
      return False
```

5. Jika terdapat solusi dari persoalan tersebut, kembalikan himpunan solusi dan jumlah step yang dibutuhkan untuk mendapatkan himpunan solusi tersebut.
6. Jika tidak, maka kembalikan tuple (0, “Tidak ada solusi”) kepada sistem untuk diproses sebagai tidak ada solusi yang dihasilkan.

   Kode di bawah ini melambangkan hal yang dilakukan oleh tahap ke-5 dan ke-6.

```
# Kembalikan tuple berisi jumlah flip dan solusi yang ada
if (solusi(solution)):
   return ((flipCount, solution))
else:
   return ((0, “Tidak ada solusi"))
```

Adapun dalam tahap-tahap tersebut ada beberapa hal yang perlu diperhatikan

1. cards merupakan himpunan kandidat dan solution merupakan himpunan solusi.
2. Kartu-kartu pada program dilambangkan oleh sebuah *string* yang menandakan kartu tersebut. Sebagai contoh, as hati akan dituliskan sebagai “A Hati”.
3. checkNumber merupakan fungsi yang mengembalikan true apabila nomor kartu pada cardA dan cardB adalah sama.
4. Fungsi solusi mengembalikan true apabila parameternya memiliki isi dari list parameter tidak kosong.

5. flipCount melambangkan jumlah pembalikan kartu dan semakin
6. Asumsi jumlah kartu yang dikurangkan pada saat pengurangan kartu tidak melebihi jumlah kartu pada *deck* kartu.
7. Algoritma ini pasti optimum karena jumlah semua pasangan kartu di reminder pasti tidak punya pasangan kartu lagi karena setiap kartu di dalamnya dicek setiap membuka kartu yang tidak memiliki pasangan.
8. Dengan algoritma ini, jumlah pembalikan kartu maksimum yang dilakukan oleh program adalah (2 * jumlah pasangan kartu). Angka ini didapatkan dari kemungkinan terburuk yaitu jika setiap pengecekan kartu tidak berpasangan satu dengan yang lain sehingga harus dicek dengan kartu di reminder atau harus dikurangi.
9. Dengan algoritma ini, jumlah pembalikan kartu minimum yang dilakukan oleh sistem adalah sama dengan jumlah pasangan kartu. Angka ini didapatkan dari kemungkinan terbaik di mana setiap pengecekan selalu menghasilkan pasangan kartu yang sama.

IV. CONTOH KASUS DAN HASIL UJI

A. *Contoh Kasus 1*

1. Urutan kartu setelah diacak
   Urutang kartu yang sudah diacak. Urutan dimulai dari kiri ke kanan dilanjutkan oleh kartu yang di bawahnya.

9 Wajik	4 Sekop	2 Hati	5 Sekop	5 Keriting	K Sekop
8 Hati	10 Keriting	A Keriting	10 Wajik	8 Sekop	4 Wajik
4 Hati	7 Wajik	J Sekop	K Wajik	8 Wajik	8 Keriting
10 Hati	J Wajik	A Wajik	9 Hati	J Hati	2 Sekop
6 Wajik	5 Wajik	6 Hati	Q Keriting	6 Sekop	Q Wajik
3 Sekop	7 Sekop	A Hati	A Sekop	4 Keriting	7 Hati
Q Hati	Q Sekop	3 Wajik	9 Keriting	3 Hati	2 Keriting
5 Hati	K Keriting	10 Sekop	7 Keriting	3 Keriting	9 Sekop
K Hati	J Keriting	6 Keriting	2 Wajik		

2. Kartu yang dikurangi oleh player
   Pemain memasukkan input urutan dari kartu yang ingin dihapus dan sistem menunjukkan kartu yang telah dihapus. Hanya saja sistem tidak dapat menambahkan input yang telah dihapus ini ke dalam solver sama halnya dengan pemain tidak dapat melihat kartu yang dikurangi.

```
Idx remove: 3 1 0 12 10 15
Cards removed:  K Wajik
Cards removed:  4 Hati
Cards removed:  8 Sekop
Cards removed:  5 Sekop
Cards removed:  4 Sekop
Cards removed:  9 Wajik
```

Gambar III. Contoh Kasus 1 Tahap 2

3. Solusi dari permasalahan
   Solusi dari permasalahan ini ditampilkan dengan pasangan-pasangan kartu yang ada.

```
Total pembalikan kartu =  41
Total pasangan kartu =  21
10 Wajik ; 10 Keriting
8 Wajik ; 8 Keriting
J Wajik ; J Sekop
A Wajik ; A Keriting
2 Sekop ; 2 Hati
5 Wajik ; 5 Keriting
6 Hati ; 6 Wajik
Q Wajik ; Q Keriting
7 Sekop ; 7 Wajik
A Hati ; A Sekop
4 Keriting ; 4 Wajik
Q Hati ; Q Sekop
3 Wajik ; 3 Sekop
9 Keriting ; 9 Hati
K Keriting ; K Sekop
10 Sekop ; 10 Hati
7 Keriting ; 7 Hati
3 Keriting ; 3 Hati
J Keriting ; J Hati
6 Keriting ; 6 Sekop
2 Wajik ; 2 Keriting
```

Gambar IV. Contoh Kasus 1 Tahap 3

Gambar di atas melambangkan jumlah pembandingan kartu yang ada (41) dan jumlah pasangan kartu yang masuk ke dalam solusi (21). Setelah itu, program menuliskan semua pasangan kartu yang didapatkan dari kartu-kartu yang terdapat dalam permainan tersebut.

4. Analisis
   a. Tahap pertama dari kasus ini adalah pembuatan testcase dengan mengacak list yang sudah mengandung kartu-kartu remi tersebut.
   b. Tahap kedua merupakan tahap mengurangi kartu dengan indeks yang melambangkan pemain mengeluarkan beberapa kartu secara tertutup dari tumpukan *deck*.
   c. Tahap ketiga merupakan pemanfaatan algoritma *greedy* pada penyelesaian permasalahan ini. Seperti yang dilihat dari contoh di atas, hasil dari algoritma tersebut merupakan hasil yang optimum karena dari kartu-kartu yang dikurangi pada saat tahap 2, jumlah kartu yang dikeluarkan, menyebabkan 4 nomor kartu kehilangan 1 kartu dan 1 nomor kartu kehilangan 2 kartu. Dengan kata lain, *deck* kartu akan kehilangan 5 pasangan kartu. Total pasangan kartu pada saat *deck* kartu masih penuh adalah (52 / 2) = 26 pasang kartu.
   d. Hasil juga dinilai baik karena kartu-kartu yang telah dikurangi tersebut tidak muncul lagi pada solusi.
   e. Jumlah pembalikan kartu pada solusi melambangkan jumlah pembalikan kartu (pembandingan) yang harus dilakukan program. Hal ini merupakan kasus yang ideal karena jumlah perbandingan masih berada dalam batas minimum (21) dan batas maksimum (42) dari jumlah perbandingan yang mungkin dilakukan.

B. Contoh Kasus 2

1. Urutan kartu setelah diacak
   Urutang kartu yang sudah diacak. Urutan dimulai dari kiri ke kanan dilanjutkan oleh kartu yang di bawahnya.

9 Sekop	2 Wajik	A Wajik	A Sekop	Q Wajik	10 Keriting
9 Hati	2 Sekop	7 Wajik	6 Hati	8 Sekop	8 Wajik
10 Wajik	7 Sekop	4 Hati	9 Keriting	5 Keriting	6 Sekop
3 Wajik	Q Sekop	6 Keriting	3 Keriting	9 Wajik	2 Hati
K Sekop	Q Keriting	10 Hati	Q Hati	7 Keriting	8 Keriting
4 Wajik	8 Hati	J Wajik	J Sekop	5 Sekop	5 Hati
4 Sekop	6 Wajik	J Hati	10 Sekop	3 Sekop	A Keriting
K Hati	5 Wajik	K Keriting	J Keriting	7 Hati	A Hati
3 Hati	K Wajik	2 Keriting	4 Keriting		

2. Kartu yang dikurangi oleh player
   Dalam contoh kasus ini, pemain tidak menambahkn kartu apa-apa untuk dikurangi dari tumpukan kartu. Maka dari itu, jumlah kartu yang terdapat dalam *deck* masih 52 kartu

```
Idx remove:
```

Gambar V. Contoh Kasus 2 Tahap 2

3. Solusi dari permasalahan
   Solusi dari permasalahan ini ditampilkan dengan pasangan-pasangan kartu yang ada.

```
Total pembalikan kartu =  48
Total pasangan kartu =  26
A Wajik ; A Sekop
9 Hati ; 9 Sekop
2 Sekop ; 2 Wajik
8 Sekop ; 8 Wajik
10 Wajik ; 10 Keriting
7 Sekop ; 7 Wajik
6 Sekop ; 6 Hati
Q Sekop ; Q Wajik
3 Keriting ; 3 Wajik
9 Wajik ; 9 Keriting
Q Hati ; Q Keriting
4 Wajik ; 4 Hati
8 Hati ; 8 Keriting
J Wajik ; J Sekop
5 Sekop ; 5 Hati
6 Wajik ; 6 Keriting
10 Sekop ; 10 Hati
K Hati ; K Sekop
5 Wajik ; 5 Keriting
J Keriting ; J Hati
7 Hati ; 7 Keriting
A Hati ; A Keriting
3 Hati ; 3 Sekop
K Wajik ; K Keriting
2 Keriting ; 2 Hati
4 Keriting ; 4 Sekop
```

Gambar VI. Contoh Kasus 2 Tahap 3

Gambar di atas melambangkan jumlah pembandingan kartu yang ada (48) dan jumlah pasangan kartu yang masuk ke dalam solusi (26). Setelah itu, program menuliskan semua pasangan kartu yang didapatkan dari kartu-kartu yang terdapat dalam permainan tersebut.

4. Analisis
   a. Tahap pertama dari kasus ini adalah pembuatan testcase dengan mengacak list yang sudah mengandung kartu-kartu remi tersebut.
   b. Tahap kedua memastikan bahwa memungkinkan untuk memulai permainan konsenterasi dengan *deck* kartu penuh.
   c. Tahap ketiga merupakan pemanfaatan algoritma *greedy* pada penyelesaian permasalahan ini. Seperti yang dilihat dari contoh di atas, hasil dari algoritma tersebut merupakan hasil yang optimum karena dari mengandung semua kartu yang terdapat dalam *deck* kartu remi.
   d. Jumlah pembalikan kartu terdapat dalam batas yang ditentukan untuk solusi algoritma ini, yaitu di atas batas bawah (26) dan di bawah batas atas (52).

C. Kesimpulan dari Contoh Kasus

Dari kedua contoh kasus di atas, dapat disimpulkan bahwa hasil yang ditampilkan sudah merupakan solusi optimum, yaitu solusi yang menghasilkan jumlah pasangan kartu yang paling banyak yang mungkin dihasilkan dari tumpukan kartu yang sudah dikurangi oleh pemain. Selain itu, jumlah perbandingan lebih mengarah ke batas atas dari perbandingan kartu. Hal ini disebabkan karena pengacakan kartu menyebabkan sebagian besar kartu yang memiliki angka yang sama tidak bersebelahan lagi sehingga sebagian besar kartu-kartu pada pengecekan perlu dimasukkan ke dalam list reminder dan dipasangkan setelah kartu dengan nomor yang sama muncul.

V. KESIMPULAN

Algoritma *greedy* dapat diimplementasikan dalam menyelesaikan permainan kartu konsenterasi. Dengan algoritma ini, maka dapat dihasilkan solusi yang mencakup semua pasangan kartu yang mungkin diambil dari sebuah *deck* kartu. Algoritma ini juga memberikan jumlah perbandingan yang sedikit, yaitu bernilai antara total maksimum pasangan kartu hingga dua kali total maksimum pasangan kartu, sehingga cocok untuk digunakan dalam meminimalkan jumlah pembalikan kartu pada permainan konsenterasi.

UCAPAN TERIMA KASIH

Penulis ingin berterima kasih kepada Tuhan YME yang atas berkat dan rahmat-Nya memampukan penulis untuk menyelesaikan makalah ini. Penulis juga ingin berterima kasih untuk orang tua yang tak henti-hentinya memberi semangat kepada penulis untuk menyelesaikan makalah ini. Terakhir penulis juga ingin berterima kasih kepada teman-teman yang telah membantu penulis mendapatkan ide mengenai permasalahan ini beserta menemani penulis selama penulisan makalah ini.

DAFTAR PUSTAKA

[1] http://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/Algoritma-Greedy-(2019).pdf, diakses pada 23 April 2019, pukul 21.00

[2] Daniel J. Velleman, and Gregory S. Warrington. 2013. "What to Expect in a Game of Memory". *The American Mathematcial Monthly*.

[3] Foerseter, K.-T; Wattenhofer R. 2013. "The Solitaire Memory Game (Technical report)". ETH Zurich

[4] Lo, A. 2009. "The game of leaves: An inquiry into the origin of Chinese playing cards". *Bulletin of the School of Oriental and Africa Studies*. 63(3): 389.

[5] Dawson, Tom, and Judy, 2014. "Honchman Encyclopedia of American Playing Cards". *Bulletin of the School of Oreintal and African Studies*

[6] https://www.idntimes.com/science/experiment/bayu/sejarah-kartu-remi, diakses pada 23 April 2019, pukul 21.40

[7] https://jalantikus.com/tips/permainan-kartu-paling-sering-dimainkan-orang-indonesia/, diakses pada 24 April 2019, pukul 19.30

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, 29 April 2012

Jan Meyer Saragih
13517131