ANALISIS PENERAPAN ALGORITMA GREEDY PADA PERMAINAN CAPSA

Vandy Putrandika (13505001)

Jurusan Teknik Informatika, Institut Teknologi Bandung
Jalan Ganesha 10, Bandung
E-mail: if15001@students.if.itb.ac.id

ABSTRAK

Dalam kehidupan sehari-hari, banyak persoalan yang menuntut pencarian solusi optimum, Persoalan optimasi adalah persoalan yang tidak hanya mencari solusi, tetapi mencari solusi terbaik. Solusi terbaik adalah solusi yang bernilai maksimum atau minimum dari sekumpulan solusi yang mungkin. Algoritma Greedy adalah metode yang paling populer dalam menyelesaikan persoalan optimasi. Algoritma ini sederhana dan *straightforward*. Seperti namanya, Greedy, algoritma ini biasanya akan mengambil solusi seoptimal mungkin yang tersedia saat itu tanpa memikirkan konsekuensi ke depan. Prinsip ini sering sekali kita gunakan, misalnya saat bermain kartu, seperti yang akan dibahas dalam makalah ini.

Kata kunci: Algoritma *Greedy*, persoalan optimasi, solusi terbaik, permainan kartu.

1. PENDAHULUAN

Salah satu penerapan algoritma *greedy* pada kehidupan sehari-hari adalah pada saat bermain kartu. Setiap pemain tentu ingin menang dalam permainan tersebut dan untuk itu mereka harus mengambil langkah-langkah yang akan membawa mereka ke solusi optimal, yaitu kemenangan.

Permainan kartu yang saya pilih untuk makalah ini adalah permainan capsa. Permainan ini menggunakan kartu remi sebagai medianya. Kartu remi adalah sekumpulan kartu yang berisi simbol wajik, keriting, hati, dan sekop. Lalu masing-masing simbol memiliki nilai dari 2 sampai 10, lalu dilanjutkan J, Q, K, dan A. Sehingga masing-masing simbol memiliki 13 nilai dan satu *deck* kartu remi berisi 52 kartu.

Peraturan dari capsa cukup mudah yaitu masing-masing pemain harus menghabiskan semua kartu di tangannya secepat mungkin. Siapa yang lebih dulu memainkan kartu terakhir di tangannya dialah yang memenangkan pertandingan.

Tetapi pemain tidak bisa begitu saja mengeluarkan semua kartu di tangannya. Ada sebuah *constaint* yaitu pemain tidak bisa memainkan kartu yang nilainya lebih rendah dari kartu yang dimainkan oleh pemain sebelumnya.

Dalam capsa ada urutan nilai yaitu dimulai dari 3, 4, 5, 6, 7, 8, 9, 10, J, Q, K, A, dan yang paling tinggi adalah 2. Selain itu pada nilai yang sama juga terdapat perbadaan nilai dilihat dari simbol. Urutan nilai simbol adalah wajik < keriting < hati < sekop. Contohnya adalah kartu 5 hati memiliki nilai lebih besar dari 3 sekop, lalu kartu 7 hati nilainya lebih besar dari 7 keriting.

2. DESKRIPSI MASALAH

Dalam permainan capsa yang sebenarnya ada beberapa cara memainkan kartu. Pertama cara *single*, atau lebih dikenal di Bandung sebagai cara ketengan. Pada cara ini kita memainkan kartu satu-satu secara bergantian. Tetapi nilai kartu yang kita mainkan harus lebih tinggi dari kartu sebelumnya. Contoh, bila pemain sebelumnya memainkan kartu 4 sekop maka kita setidaknya harus memainkan kartu 5 wajik. Tidak boleh kartu 3 keriting, misalnya. Bila semua pemain sudah tidak bisa atau tidak mau memainkan kartu yang melebihi nilai kartu sebelumnya maka tumpukan akan dibersihkan dan pemain yang terakhir memainkan kartu memulai sebuah tumpukan baru dengan nilai awal yang terserah dia.

Cara kedua adalah cara *pair*, atau bisa juga disebut pasangan. Pada cara ini pemain juga mmemainkan kartu secara bergantian dan juga tidak boleh memainkan kartu yang nilainya lebih rendah dari sebelumnya. Bedanya, pemain harus memainkan kartunya secara berpasangan dua-dua dengan nilai yang sama. Contoh, 7 wajik dengan 7 hati dapat dilawan dengan 10 hati dan 10 sekop.

Cara ketiga adalah cara paket. Pada cara ini pemain memainkan langsung lima kartu yang memiliki pola tertentu secara sekaligus. Ada beberapa pola dalam paket, yaitu *straight* atau seri, lalu *flush*, kemudian *full house*, selanjutnya adalah *straight flush*, dan terakhir adalah bom.

Straight adalah pola dimana nilai dari kelima kartu yang kita mainkan terurut menaik. Contoh, kita memainkan kartu 4 hati, 5 sekop, 6 wajik, 7 hati, dan 8 sekop secara sekaligus. Kedua adalah *flush*, yaitu sebuah pola dimana kelima kartu yang kita mainkan memiliki simbol yang sama. Contoh, 4 hati, 7 hati, 8 hati, 10 hati, A hati. Cara

selanjutnya adalah *full house* yaitu pola dimana kita memainkan tiga kartu yang memiliki nilai sama ditambah dua kartu dengan nilai lain yang sama. Contoh, 4 hati, 4 keriting, 4 sekop, ditambah 7 wajik dan 7 sekop.

Setelah itu ada pola *Straight flush*, seperti namanya pola ini adalah gabungan dari pola straight dan pola flush. Yaitu kita harus memainkan kartu yang nilainya terurut dan memiliki simbol yang sama. Contoh, 6 hati, 7 hati, 8 hati, 9 hati, 10 hati. Terakhir adalah bom, pola ini mengharuskan kita memiliki empat kartu dengan nilai yang sama ditambah satu kartu apa pun. Contoh, 4 wajik, 4 hati, 4 keriting, 4 sekop, ditambah 7 hati.

Ada beberapa pola lain seperti *threes* dan *double pair*, namun pola-pola ini hanya dimainkan di kalangan tertentu.

Cara-cara ini hanya dapat dipilih saat tumpukan kartu kosong. Tumpukan kartu dikosongkan saat awal permainan atau semua pemain memilih untuk melewatkan gilirannya. Bila salah satu pemain telah memilih salah satu cara maka pemain lain harus mengikuti cara tersebut sampai tumpukan dikosongkan kembali.

Dalam makalah ini yang akan dibahas hanyalah permainan capsa dengan cara *single* dan *pair* saja. Saya akan membahas tentang algoritma *Greedy* yang dapat digunakan untuk menyelesaikan permainan kartu.

3. ALGORITMA GREEDY

Algoritma *greedy* membentuk solusi langkah per langkah. Terdapat banyak pilihan yang perlu dieksporasi pada setiap langkah solusi. Oleh karena itu, pada setiap langkah harus dibuat keputusan yang terbaik dalam menentukan pilihan. Keputusan yang telah diambil pada setiap langkah tidak dapat diubah lagi pada langkah selanjutnya. Pendekatan yang digunakan pada algoritma *greedy* adalah membuat pilihan yang tampaknya memberikan perolehan terbaik, yaitu dengan membuat pilihan optimum lokal pada setiap langkah dengan harapan bahwa sisanya akan mengarah ke solusi optimum global.

Skema umum dari algoritma *greedy* disusun oleh elemen-elemen sebagai berikut :

1. Himpunan kandidat
   Himpunan ini berisi elemen-elemen pembentuk solusi.
2. Himpunan solusi
   Himpunan ini berisi bagian dari himpunan kandidat yang terpilih sebagai solusi persoalan
3. Fungsi seleksi
   Fungsi ini adalah fungsi yang pada setiap langkah memilih solusi yang paling memungkinkan mencapai solusi optimal.
4. Fungsi kelayakan
   Fungsi ini memasukkan kandidat-kandidat yang layak dari himpunan kandidat ke himpunan solusi.
5. Fungsi obyektif
   Yaitu fungsi yang memaksimumkan atau meminimumkan nilai solusi

4. STRATEGI PENYELESAIAN MASALAH

Pada makalah ini akan dibahas tentang penggunaan algoritma *greedy* untuk menyelesaikan dua buah cara dalam permainan capsa. Yang pertama adalah cara *single* saja. Yang kedua adalah cara kombinasi antara *single* dan *pair*.

4.1 Strategi Penyelesaian Cara *Single*

Pertama-tama kita akan meninjau elemen-elemen dari algoritma greedy untuk menyelesaikan masalah ini.

1. Himpunan Kandidat : kartu-kartu yang telah dibagikan dan ada di tangan pemain.
2. Himpunan Solusi : kartu-kartu yang telah dibagikan dan memiliki nilai yang lebih besar dari kartu yang dimainkan sebelumnya.
3. Fungsi Seleksi : kartu yang memiliki nilai lebih besar daripada kartu sebelumnya namun memiliki nilai yang paling kecil diantara himpunan solusi.
4. Fungsi Kelayakan : kartu yang memiliki nilai yang lebih besar dari kartu sebelumnya
5. Fungsi Obyektif : menghabiskan semua kartu di tangan secepat mungkin

```
procedure GreedySingle (I : Kartu[] Hand, I/O :
  Kartu Last)
{ mencari solusi optimum untuk menghabiskan
  semua kartu di tangan secepat mungkin
}
Deklarasi
  Hand : array of Kartu //terurut menaik
  Last : kartu terakhir yang dimainkan

Algoritma:
  if (!isEmpty(Hand)) {
    While (Hand[i] < Last && i < Hand.size) {
      i++;
    } // solusi karena terurut menaik
    If (i < Hand.size) {
      Print(Hand[i]);
      Last = Hand[i];
    } else {
      Pass();
    }
    //lalu pemain lain mendapat gilirannya
  } else {
    Print (”you win”);
  }
```

Pada algoritma *GreedySingle* ini kita akan memilih kartu dengan nilai yang lebih besar dari kartu terakhir

yang dimainkan untuk dimasukkan ke himpunan solusi. Lalu dari himpunan solusi tersebut akan diambil kartu dengan nilai terkecil agar kartu-kartu dengan nilai lebih besar dapat digunakan pada giliran kita berikutnya, sehingga kita akan selalu memainkan kartu dan membuat semua kartu di tangan menjadi cepat habis karena tidak melewatkan giliran.

4.2 Strategi Penyelesaian Cara Kombinasi *Single* dan *Pair*

Pertama-tama kita akan meninjau elemen-elemen dari algoritma greedy untuk menyelesaikan masalah ini.

1. Himpunan Kandidat : kartu-kartu yang telah dibagikan dan ada di tangan pemain.
2. Himpunan Solusi : kartu-kartu yang telah dibagikan dan memiliki nilai yang lebih besar dari kartu yang dimainkan sebelumnya. Dan bila kartu sebelumnya *pair* maka himpunan solusi ini juga berisi kartu-kartu *pair*.
3. Fungsi Seleksi : kartu yang memiliki nilai lebih besar daripada kartu sebelumnya namun memiliki nilai yang paling kecil diantara himpunan solusi dan memiliki cara yang sama, entah *single* atau *pair*.
4. Fungsi Kelayakan : kartu yang memiliki nilai yang lebih besar dari kartu sebelumnya
5. Fungsi Obyektif : menghabiskan semua kartu di tangan secepat mungkin

```
procedure GreedyCombination (I : Kartu[] Hand,
  I/O : Kartu Last)
{ mencari solusi optimum untuk menghabiskan
  semua kartu di tangan secepat mungkin
}
Deklarasi
  Hand : array of Kartu //terurut menaik
  Last : kartu terakhir yang dimainkan

Algoritma:
  if (!isEmpty(Hand)) {
    cariPair(Hand);
    if ((isPair(Last) || (MulaiTumpukanBaru()
  && adaPairYangMemenuhiSolusi)) {

      if (adaPairYangMemenuhiSolusi) {
        Print(PairSolusiTerkecil());
        Last = PairSolusiTerkecil();
      } else {
        Pass();
      }
    } else {
      GreedySingle(Hand, Last);
    }
    //lalu pemain lain mendapat gilirannya
  } else {
    Print (”you win”);
  }
```

Pada algoritma *GreedyCombination* ini pertama-tama pemain akan mencoba memilih mengeluarkan pair bila pemain sebelumnya mengeluarkan pair atau tumpukan kosong, sehingga pemain bebas memilih *single* atau *pair*. Bila saat tumpukan kosong namun tidak memiliki pair maka pemain akan memilih untuk bermain dengan cara *single*. Namun bila tumpukan kosong dan pemain memiliki *pair*, maka pemain akan bermain dengan cara *pair* karena dalam permainan *pair* kartu yang dikeluarkan akan lebih banyak dan membuat kartu di tangan cepat habis. Seperti pada *GreedySingle*, pada cara *pair* di *GreedyCombination*, kita akan memilih *pair* dengan nilai terkecil yang memenuhi fungsi kelayakan.

5. HASIL STATISTIK PERCOBAAN MELAWAN KOMPUTER DENGAN ALGORITMA GREEDY

Saya telah membuat sebuah program Java yang dapat dimainkan untuk melawan 3 pemain yang digerakkan oleh algoritma Greedy. Bila ingin mencoba, program dapat diambil di students.if.itb.ac.id/~if15001/STIMIK, namun masih banyak *bug* karena tidak adanya validasi. Di bawah ini adalah hasil dari percobaan saya bermain dengan komputer.

5.1 Hasil Percobaan Modus *Single*

Tabel 1 Hasil 15 Percobaan Pertama Modus *Single*

No	Pemain	P2	P3	P4
1	8	4	0	4
2	0	2	2	3
3	4	2	0	2
4	5	0	4	4
5	2	2	1	0
6	5	0	4	4
7	0	2	3	2
8	0	4	5	2
9	2	3	3	0
10	1	4	0	3
11	0	3	2	1
12	1	3	0	1
13	5	3	1	0
14	2	0	3	8
15	0	2	4	5

Ket : angka di dalam tabel adalah sisa kartu saat permainan berakhir. Pemain dengan sisa kartu 0 adalah pemenangnya

Rekap hasil pertandingan *single*

1. Pemain
   Menang = 5
   Rata-rata sisa kartu = 2.333
2. P2
   Menang = 3
   Rata-rata sisa kartu = 2.267
3. P3
   Menang = 4
   Rata-rata sisa kartu = 2.133
4. P4
   Menang = 3
   Rata-rata sisa kartu = 2.600

5.2 Hasil Percobaan Modus Kombinasi

Tabel 2 Hasil 15 Percobaan Pertama Modus Kombinasi

No	Pemain	P2	P3	P4
1	0	7	9	6
2	0	2	2	2
3	2	0	2	3
4	0	2	1	4
5	2	7	0	7
6	0	5	1	4
7	1	0	4	2
8	1	1	2	0
9	0	4	1	4
10	0	4	3	2
11	4	7	4	0
12	0	5	6	5
13	0	3	5	4
14	0	7	7	5
15	3	2	0	2

Ket : angka di dalam tabel adalah sisa kartu saat permainan berakhir. Pemain dengan sisa kartu 0 adalah pemenangnya.

Rekap hasil pertandingan

5. Pemain
   Menang = 9
   Rata-rata sisa kartu = 0.867
6. P2
   Menang = 2
   Rata-rata sisa kartu = 3.733
7. P3
   Menang = 2
   Rata-rata sisa kartu = 3.133
8. P4
   Menang = 2
   Rata-rata sisa kartu = 3.333

6. KESIMPULAN

Algoritma *Greedy* ternyata memberikan hasil yang cukup baik di Permainan Capsa dengan modus *single*, namun kurang begitu bagus saat diubah ke modus kombinasi. Hal ini disebabkan oleh semakin banyaknya hal yang harus dipertimbangkan oleh pemain pada modus kombinasi. Sehingga dalam hal ini fungsi seleksi dari Algoritma *Greedy* harus diperbaiki agar lebih banyak elemen yang diperhatikan, seperti menyimpan kartu tertentu untuk dikeluarkan belakangan agar algoritma *greedy* dapat bekerja lebih optimal, karena semakin banyak elemen yang dipertimbangkan dalam algoritma *greedy* maka semakin optimal lah algoritma *greedy* tersebut.

REFERENSI

[1] Munir, Rinaldi, “Strategi Algoritmik”, ITB, 2007.