# *Penerapan Algoritma Greedy dalam Permainan Capsa Susun*

Alvin Sullivan / 13515048
*Program Studi Teknik Informatika*
*Sekolah Teknik Elektro dan Informatika*
*Bandung, Indonesia*
*13515048@std.stei.itb.ac.id*

***Abstrak*—Makalah ini menjelaskan sedikit tentang bagaimana memanfaatkan algoritma *greedy* dalam sebuah permainan, khususnya permainan kartu remi. Algoritma *greedy* merupakan algoritma yang menyelesaikan suatu permasalahan dengan pendekatan mencari nilai maksimum sementara pada setiap langkahnya. Sedangkan kartu remi merupakan kartu seukuran tangan yang digambarkan dengan angka dan gambar. Dalam makalah ini, pembahasan akan lebih difokuskan pada bagaimana algoritma *greedy* mampu menentukan pengambilan keputusan sebuah permainan kartu remi sehingga mampu menentukan kemenangan pemain. Sampel yang digunakan yaitu sebuah permainan kartu yang biasa disebut capsa susun. Pembuatan makalah ini bertujuan untuk menganalisis bagaimana penyusunan kombinasi kartu pada permainan capsa susun sehingga mampu menghasilkan kemungkinan kemenangan setinggi mungkin.**

***Kata Kunci—Kartu Remi, Algoritma Greedy, Capsa Susun, Penyusunan Kombinasi, Kemungkinan Kemenangan***

## I. PENDAHULUAN

Kartu remi merupakan kartu yang kaya akan fungsi di berbagai bidang. Banyak yang menggunakan kartu ini untuk bermain di waktu luang, ada yang menggunakan kartu ini untuk melakukan trik sulap, hingga bahkan di negara barat kartu ini juga banyak digunakan pada perjudian. Salah satu faktor yang menyebabkan kartu ini banyak digunakan yaitu karena ada banyaknya variasi kartu namun juga mudah diingat dan digunakan menjadi ide permainan. Selain itu, kartu ini juga sudah terkenal di berbagai kalangan, baik di negara barat maupun timur, atau dari anak-anak hingga dewasa.

GAMBAR 1. Kartu Remi [6]

Kartu remi terdiri dari pasangan 13 kombinasi angka dan 4 kombinasi gambar. Angka pada kartu remi yaitu satu sampai tiga belas, di mana angka satu direpresentasikan dengan A (kartu As), angka 11 direpresentasikan dengan J (kartu Jack), angka 12 direpresentasikan dengan Q (kartu Queen), dan angka 13 direpresentasikan dengan K (kartu King). Gambar pada kartu remi yaitu wajik, keriting, hati, dan sekop. Dengan kombinasi 13 angka dan 4 gambar ini, maka akan didapatkan 13 x 4 = 52 kartu berbeda. Kartu remi juga memiliki tiga kartu Joker sebagai tiga kartu tambahan, yaitu dua Joker berwarna dan satu Joker hitam. Namun kartu Joker hanya dipakai pada permainan kartu remi tertentu saja.

Permainan remi banyak menggunakan algoritma pada pengaplikasiannya karena banyak memainkan kombinasi kartu dan peluang. Salah satu permainan kartu remi yang cukup terkenal dan menarik yang akan saya bahas yaitu permainan capsa susun. Alasan saya memilih permainan ini yaitu karena ada berbagai kombinasi tidak terduga yang sering saya dapatkan pada pengalaman saya memainkan permainan ini yang mampu memberi hasil yang tidak terduga. Untuk itu, saya tertarik untuk mencoba algoritma *greedy* untuk memeriksa apakah dengan algoritma ini, apakah mampu dihasilkan kombinasi yang menghasilkan kemungkinan kemenangan tertinggi atau tidak.

## II. DASAR TEORI

Algoritma *greedy* merupakan algoritma yang menyelesaikan suatu permasalahan dengan pendekatan mencari nilai maksimum sementara pada setiap langkahnya [2]. Kata *greedy* itu sendiri berasal dari bahasa Inggris yang artinya rakus, tamak, atau serakah. Prinsip yang digunakan pada algoritma *greedy* yaitu "*take what you can get now*" [3].

Algoritma ini merupakan algoritma yang menyelesaikan persoalan yang membutuhkan solusi dalam bentuk optimasi. Optimasi di sini berarti algoritma *greedy* menyelesaikan persoalan yang membutuhkan solusi maksium atau persoalan yang membutuhkan solusi minimum.

Untuk setiap langkah, algoritma ini mencari optimum lokal dengan berharap bahwa nilai tersebut akan mengarahkan ke solusi optimum global jika diteruskan. Optimum lokal berarti nilai maksimum atau minimum yang dapat diperoleh dari data sisa, sedangkan optimum global berarti nilai maksimum atau minimum sesungguhnya yang dapat diperoleh dari data yang ada.

Algoritma ini memiliki lima elemen yang mendukung jalannya algoritma ini, yaitu himpunan kandidat, himpunan solusi, fungsi seleksi, fungsi layak, dan fungsi obyektif [4].

1. Himpunan kandidat
   Himpunan kandidat merupakan himpunan pilihan yang akan diambil sebanyak jumlah tertentu menjadi suatu solusi.
2. Himpunan solusi
   Himpunan solusi merupakan himpunan sejumlah pilihan yang telah dipilih untuk menjadi solusi optimum dari hasil algoritma *greedy*.
3. Fungsi seleksi
   Fungsi seleksi merupakan fungsi untuk memilih pilihan dengan nilai tertinggi atau terendah dari himpunan kandidat yang tersisa.
4. Fungsi layak
   Fungsi layak merupakan fungsi untuk memeriksa apakah solusi optimum yang didapat dari hasil algoritma *greedy* merupakan solusi yang memenuhi syarat permintaan persoalan.
5. Fungsi obyektif
   Fungsi obyektif merupakan fungsi lain yang diperlukan jika mampu memaksimumkan atau meminimumkan hasil algoritma *greedy*.

Algoritma *greedy* merupakan algoritma yang tidak selalu menghasilkan solusi yang paling optimum. Solusi yang dihasilkan dari algoritma ini merupakan solusi yang cenderung mendekati optimum. Namun kerja dari algoritma ini sangat mudah dimengerti dan dapat bekerja dengan sangat cepat.

## III. Deskripsi dan Pembatasan Masalah

### A. *Deskripsi Masalah*

Capsa susun merupakan sebuah permainan kartu remi yang dapat dimainkan dengan maksimal empat orang. Setiap orang akan dibagikan tiga belas kartu dan setiap orang akan menyusun tiga belas kartu tersebut dengan membagi menjadi tiga tingkat, yaitu tingkat bawah, tingkat tengah, dan tingkat atas. Aturan penyusunan yang berlaku yaitu tingkat bawah harus memiliki poin (perhitungan poin akan dijelaskan di bagian pembatasan masalah) yang lebih besar daripada tingkat tengah. Begitu juga tingkat tengah harus memiliki poin yang lebih besar daripada tingkat atas.

GAMBAR 2. Susunan Kartu pada Capsa Susun [5]

Perhitungan poin akan dilakukan sesuai dengan kombinasi yang terbentuk dari setiap tingkat. Satu tingkat dengan tingkat lainnya bersifat independen dalam perhitungan poin, sehingga perhitungan poin tingkat atas tidak akan dipengaruhi tingkat tengah dan bawah, perhitungan poin tingkat tengah tidak akan dipengaruhi tingkat atas dan bawah, dan perhitungan poin tingkat bawah tidak akan dipengaruhi tingkat atas dan tengah.

Penentuan kemenangan akan dilakukan dengan membandingkan kombinasi antar pemain. Setiap pemain akan membandingkan kombinasi yang dibentuk dengan semua lawan yang dihadapi satu per satu. Pada setiap perbandingan antar pemain, setiap pemain akan membandingkat poin dari tingkat bawah, tengah, dan atas mereka. Pemain yang memenangkan tingkat lebih banyak akan menjadi pemenang lokal. Pemenang lokal di sini berarti pemain tersebut baru memenangkan perbandingan dengan satu lawan saja. Setelah semua pemain menghitung jumlah kemenangan yang mereka peroleh, akan dicari pemain yang memperoleh kemenangan terbanyak. Pemain yang memperoleh kemenangan terbanyak inilah yang menjadi pemenang global, yaitu pemain yang memenangkan keseluruhan permainan. Dengan perhitungan seperti ini, maka dapat ditentukan juga siapa pemenang kedua dari permainan, hingga pemenang terakhir. Ada kemungkinan juga antar pemain mendapatkan total poin yang sama sehingga mendapatkan peringkat pemenang yang sama.

Masalah timbul pada bagaimana pemain menentukan kombinasi yang akan mereka bentuk di tingkat bawah, tingkat tengah, dan tingkat atas sekaligus. Ada sangat banyak kombinasi yang dapat dibentuk hanya dari tiga belas kartu yang dibagikan. Oleh karena itu, saya akan mencoba menyelesaikan permasalahan ini dengan algoritma *greedy*.

### B. *Pembatasan Masalah*

Ada sejumlah cara perhitungan poin yang dapat diaplikasikan pada permainan capsa susun. Cara pertama yaitu dengan menjumlahkan total poin dari tingkat bawah, tingkat tengah, dan tingkat atas sehingga menjadi total poin seorang pemain. Total poin ini kemudian menjadi poin yang dibandingkan dengan pemain lain. Cara kedua yaitu dengan hanya membandingkan kombinasi mana yang lebih kuat dari tingkat bawah, tingkat tengah, dan tingkat atas. Pemain yang memenangkan tingkat terbanyak akan menjadi pemenang antar kedua pemain tersebut.

Berikut ini merupakan penjabaran kombinasi yang dikenal pada kartu remi untuk mengenal istilah-istilah yang ada terlebih dahulu.

TABEL 1. Tabel Kombinasi pada Kartu Remi

| No | Kombinasi | Definisi |
|---|---|---|
| 1 | *Single* | Tidak ada kombinasi yang terbentuk |
| 2 | *Pairs* | Dua kartu dengan angka yang sama |
| 3 | *Three of a kind* | Tiga kartu dengan angka yang sama |
| 4 | *Straight* | Lima kartu dengan angka yang berurutan |
| 5 | *Flush* | Lima kartu dengan gambar yang sama |
| 6 | *Full house* | Tiga kartu dengan angka yang sama dan dua kartu lain dengan angka yang sama |

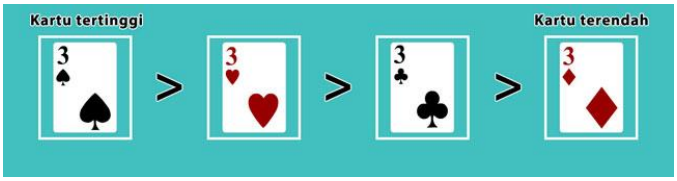
| | | |
|---|---|---|
| 7 | *Four of a kind* | Empat kartu dengan angka yang sama |
| 8 | *Straight flush* | Lima kartu dengan angka yang berurutan dan gambar yang sama |
| 9 | *Royal straight flush* | Lima kartu dengan gambar yang sama dan dengan angka terdiri dari 10, J, Q, K, dan A |
| 10 | *Dragon* | Tiga belas kartu dengan angka yang berurutan |

Berikut ini merupakan urutan kombinasi beserta poin dari cara perhitungan poin pertama [1].

TABEL 2. Tabel Poin Cara Pertama

| No | Kombinasi | Poin |
|---|---|---|
| 1 | *Three of a kind* di tingkat atas | 5 |
| 2 | *Full house* di tingkat tengah | 3 |
| 3 | *Full house* di tingkat bawah | 1 |
| 4 | *Four of a kind* di tingkat tengah | 14 |
| 5 | *Four of a kind* di tingkat bawah | 7 |
| 6 | *Straight flush* di tingkat tengah | 18 |
| 7 | *Straight flush* di tingkat bawah | 9 |
| 8 | *Royal straight flush* di tingkat tengah | 22 |
| 9 | *Royal straight flush* di tingkat bawah | 11 |
| 10 | *Three straight* | 9 |
| 11 | *Three flush* | 9 |
| 12 | *Six pairs* | 9 |
| 13 | *Five pairs* dan *one three of a kind* | 18 |
| 14 | Warna sama | 24 |
| 15 | *Dragon* | 39 |
| 16 | *Dragon* sama warna | 139 |

Cara kedua tidak memiliki aturan perhitungan poin pada penentuan kemenangan. Pemenang hanya ditentukan dengan banyaknya tingkat dengan kombinasi lebih kuat dari kombinasi lawan di tingkat yang sama. Jika kombinasi sama, maka akan dilihat kartu dengan angka dan gambar terkuat. Pada hal ini, kekuatan angka ditentukan berdasarkan besarnya angka, kecuali kartu As yang merupakan kartu terkuat pada permainan ini. Kekuatan angka juga ditentukan berdasarkan gambarnya, yaitu dengan urutan dari paling lemah ke paling kuat yaitu: wajik, keriting, hati, dan sekop. Kekuatan gambar ini dilihat jika kedua kartu terkuat yang dibandingkan memiliki angka yang sama pula. Sehingga gambar dapat menghindari adanya kondisi imbang antar kombinasi.




GAMBAR 3. Urutan Kekuatan Gambar [7]

Urutan kombinasi pada cara kedua dari paling lemah ke paling kuat dijabarkan pada tabel berikut.

TABEL 3. Tabel Urutan Kekuatan Kombinasi Cara Kedua

| No | Kombinasi |
|---|---|
| 1 | *Single* |
| 2 | *Pairs* |
| 3 | *Two pairs* |
| 3 | *Three of a kind* |
| 4 | *Straight* |
| 5 | *Flush* |
| 6 | *Full house* |
| 7 | *Four of a kind* |
| 8 | *Straight flush* |
| 9 | *Royal straight flush* |

Dengan mempertimbangkan cara penilaian yang lebih umum digunakan dan kesederhanaan cara penilaian yang ada, maka saya memutuskan untuk membatasi permasalahan dengan menggunakan cara penentuan kemenangan kedua, yaitu dengan menghitung banyak tingkat yang dimenangkan pemain berdasarkan kekuatan kombinasi yang dibentuk tiap tingkatnya.

## IV. Pengaplikasian Algoritma Greedy

### A. *Program Eksperimen*

Untuk mendemonstrasikan bagaimana algoritma *greedy* memecahkan persoalan penentuan kombinasi terbaik pada capsa susun, maka saya membuat sebuah program dalam bahasa C++ yang menerima input dan mengeluarkan output secara *text-based* pada *command prompt*.

Berikut ini merupakan sepenggal program yang saya buat yang hanya menampilkan struktur data dan prosedur inti saja.

```
typedef struct {
    int Number;
    int Image;
} Card;

typedef struct {
    Card Draw[13];
} Cards;

void GreedyResult (Cards *C) {
    cout << "Result : " << endl;
    RoyalStraightFlush(C);
    StraightFlush(C);
    FourOfAKind(C);
    FullHouse(C);
    Flush(C);
    Straight(C);
    ThreeOfAKind(C);
    TwoPairs(C);
    Pairs(C);
    Single(C);
}

int main() {
    Help();
    Cards C;
    InitializeCards(&C);
    GreedyResult(&C);
    return 0;
}
```

Program yang saya buat memanggil prosedur Help untuk menampilkan petunjuk penggunaan program terlebih dahulu. Kemudian program membentuk sebuah tipe data yang saya buat bernama Cards yang merupakan larik yang terdiri dari tipe data Card. Tipe data Card terdiri dari dua buah bilangan bulat, yaitu Number untuk merepresentasikan angka pada kartu (2..10, 11 (Jack), 12 (Queen), 13 (King), 14 (As)) dan Image untuk merepresentasikan gambar (1 (wajik), 2 (keriting), hati (3), atau sekop (4)) pada kartu. Setelah itu dipanggil prosedur InitializeCards untuk menerima input tiga belas kartu dari pengguna berupa kombinasi kode angka kartu dan kode gambar kartu. Setelah semua kartu sudah dimasukkan, maka program mulai menjalankan algoritma *greedy* untuk menghasilkan kombinasi pemecahan persoalan.

Prosedur GreedyResult memanggil semua prosedur kombinasi satu per satu dari kombinasi yang paling kuat hingga kombinasi yang paling lemah. Setiap prosedur akan memeriksa apakah kombinasi yang bersangkutan ada pada tiga belas kartu masukkan pengguna. Jika ada, maka kartu-kartu tersebut ditampilkan ke layar dan kartu-kartu tersebut dihapus dari kumpulan kartu pengguna. Selama kombinasi masih ada dan tingkatan masih memenuhi syarat, maka prosedur akan terus dijalankan. Tingkatan yang masih memenuhi syarat di sini berarti kombinasi yang membutuhkan lima kartu hanya boleh dijalankan selama program masih berada di tingkat bawah atau tingkat tengah. Jika program sudah mencapai tingkat atas, maka kombinasi lima kartu sudah tidak boleh dijalankan karena tingkat atas hanya terdiri dari tiga kartu. Setelah suatu prosedur selesai, maka program berlanjut ke prosedur selanjutnya dengan melanjutkan kondisi kartu pengguna yang sudah diproses pada prosedur-prosedur sebelumnya.

### *B. Algoritma Greedy dalam Program*

Algoritma *greedy* di sini berjalan dengan memprioritaskan pengambilan kombinasi terkuat terlebih dahulu, lalu berurut hingga prioritas terakhir yaitu kombinasi terlemah. Berikut ini elemen-elemen algoritma *greedy* yang berlaku pada program ini.

1. Himpunan kandidat
   {*Royal straight flush*, *Straight flush*, *Four of a kind*, *Full house*, *Flush*, *Straight*, *Three of a kind*, *Two pairs*, *Pairs*}
2. Himpunan solusi
   Kombinasi tiga pilihan dari himpunan kandidat.
3. Fungsi seleksi
   Ambil kombinasi yang terkuat terlebih dahulu jika ada, lalu berlanjut secara berurut ke kombinasi yang lebih lemah.
4. Fungsi layak
   Pastikan tidak ada kombinasi lima kartu pada tingkat atas.
5. Fungsi obyektif
   Analisis apakah hasil yang diperoleh dapat menjadi salah satu solusi kombinasi yang memiliki kemungkinan besar memenangkan permainan.

### *C. Hasil Eksekusi Program*

Berikut ini merupakan salah satu contoh dari eksekusi program yang saya buat tanpa tampilan prosedur Help.

```
Input cards :
13 4
13 3
13 2
12 3
12 1
10 2
9 3
8 2
7 1
6 2
5 4
4 2
3 2
Result :
Level 0 : Full House (13-4, 13-3, 13-2, 12-3, 12-
1)
Level 1 : Flush (10-2, 8-2, 6-2, 4-2, 3-2)
Single card(s) left to be inserted : (9-3) (7-1)
(5-4)
Process returned 0 (0x0)   execution time : 0.624
s
```

Program menerima input tiga belas kartu, yaitu : King sekop, King hati, King keriting, Queen hati, Queen wajik, 10 keriting, 9 hati, 8 keriting, 7 wajik, 6 keriting, 5 sekop, 4 keriting, dan 3 keriting. Pertama program memeriksa kombinasi *royal straight flush* dari input dan kombinasi tidak ditemukan. Berlanjut terus hingga pada pemeriksaan kombinasi *full house*, ditemukan kombinasi King sekop, King hati, King keriting, Queen hati, dan Queen wajik. Kelima kartu tersebut juga dihapus dari koleksi input kartu. Kemudian program berlanjut mencari kombinasi hingga pada pencarian *Flush*, ditemukan kombinasi 10 keriting, 8 keriting, 6 keriting, 4 keriting, dan 2 keriting. Kelima kartu tersebut kemudian dihapus kembali dari koleksi kartu input. Kemudian pencarian berlanjut hingga pada akhirnya tidak ada lagi kombinasi yang ditemukan. Kartu tersisa merupakan kartu yang tidak dapat membentuk kombinasi lagi sehingga dapat disisipkan di tingkat bawah, tengah, atau atas jika masih ada tempat kosong. Tempat kosong pada tingkat bawah dan tingkat tengah dapat timbul jika kombinasi yang ditemukan seperti kombinasi *Four of a kind* yang menggunakan empat kartu, *Three of a kind* yang menggunakan tiga kartu, dan sebagainya.

Jika diperiksa apakah input valid, maka kita dapat melihat input pengguna satu per satu. Sesuai aturan program, input angka dilambangkan dengan kode 2 hingga 14, di mana Jack dilambangkan dengan 11, Queen dilambangkan dengan 12, King dilambangkan dengan 13, dan As dilambangkan dengan 14. Input gambar dilambangkan dengan kode 1 hingga 4, yaitu wajik dilambangkan dengan 1, keriting dilambangkan dengan 2, hati dilambangkan dengan 3, dan sekop dilambangkan dengan 4. Melihat input pengguna di mana tidak ada input yang berada di luar jangkauan kode, maka input sudah valid.

Lalu jika diperiksa apakah hasil kombinasi valid, maka kita dapat melihat keluaran program satu per satu. Sesuai aturan permainan, tingkat bawah atau pada program dilambangkan dengan *Level 0* memiliki kombinasi lebih kuat daripada tingkat

tengah atau pada program dilambangkan dengan *Level 1*. Pada keluaran program, dihasilkan kombinasi pada tingkat bawah yaitu *Full house* dan kombinasi pada tingkat tengah yaitu *Flush*. Kombinasi *Full house* sudah lebih kuat daripada kombinasi *Flush*, maka keluaran sementara masih valid. Lalu diperiksa apakah tingkat tengah atau pada program dilambangkan dengan *Level 1* lebih kuat daripada tingkat atas atau pada program dilambangkan dengan *Level 2*. Karena keluaran program hanya menampilkan kartu tersisa, ini berarti tingkat atas akan diisi dengan kombinasi *Single* karena program sudah tidak menemukan kombinasi apapun lagi. Kombinasi *Flush* sudah lebih kuat daripada kombinasi *Single*, maka keluaran dapat disimpulkan valid.

## V. ANALISIS

Setelah dianalisis dengan mencoba mempertemukan kombinasi hasil program dengan beberapa sampel dan memutar-mutar kombinasi hasil program dengan kombinasi lain, kombinasi dari program sudah dapat memenangkan cukup banyak perbandingan. Namun ada suatu kasus yang tidak dapat ditangani algoritma ini, yaitu jika kombinasi tingkat bawah dan tengah terlalu lemah, sebenarnya dapat diterapkan trik melemahkan kombinasi tingkat-tingkat tersebut dan memperkuat tingkat atas dengan menggunakan kombinasi *Pairs* pada tingkat tersebut. Dengan trik tersebut, tingkat bawah atau tengah yang memiliki kemungkinan menang relatif kecil dapat ditangani dengan tingkat atas yang mampu menghasilkan kemungkinan menang lebih besar. Selain itu, algoritma ini juga belum menangani peletakkan *Single* sebagai sisa kartu yang disisipkan di tempat kosong sisa. Trik ini dapat diperagakan pada ilustrasi berikut.

Diberikan input tiga belas kartu yang terdiri dari As sekop, King sekop, Queen hati, Jack wajik, Jack keriting, 10 sekop, 8 hati, 8 wajik, 7 sekop, 7 keriting, 6 wajik, 5 sekop, dan 3 wajik. Dengan input ini, program akan menghasilkan kombinasi :

1. *Flush* di tingkat bawah
   (As sekop, King sekop, 10 sekop, 7 sekop, 5 sekop)
2. *Two pairs* di tingkat tengah
   (Jack keriting, Jack wajik, 8 hati, 8 wajik)
3. Kartu sisa :
   Queen hati, 7 keriting, 6 wajik, 3 wajik

Jika disusun, maka dihasilkan :

1. *Flush* di tingkat bawah
   (As sekop, King sekop, 10 sekop, 7 sekop, 5 sekop)
2. *Two pairs* di tingkat tengah
   (Jack keriting, Jack wajik, 8 hati, 8 wajik, 3 wajik)
3. *Single* di tingkat atas
   (Queen hati, 7 keriting, 6 wajik)

Kemudian kombinasi yang dihasilkan program ini akan dihadapi dengan sebuah kombinasi dari musuh dengan input tiga belas kartu yang terdiri dari . Dengan input ini, misalkan musuh membuat kombinasi :

1. *Straight* di tingkat bawah
   (7 wajik, 6 keriting, 5 hati, 4 sekop, 3 keriting)
2. *Three of a kind* di tingkat tengah
   (9 sekop, 9 hati, 9 keriting, 4 hati, 3 sekop)
3. *Single* di tingkat atas
   (King hati, Queen wajik, Jack sekop)

Pertama dipastikan terlebih dahulu bahwa input program dan musuh valid. Input program sudah terdiri dari tiga belas kartu, input musuh sudah terdiri dari tiga belas kartu, dan dua puluh enam kartu ini sudah berbeda satu dengan yang lainnya.

Kita juga perlu memeriksa apakah kombinasi sudah valid. Pada program, kombinasi tingkat bawah yaitu *Flush* sudah lebih kuat daripada tingkat tengah yaitu *Two pairs*. Selain itu, kombinasi tingkat tengah yaitu *Two pairs* juga sudah lebih kuat daripada tingkat atas yaitu *Single*. Pada musuh, kombinasi tingkat bawah yaitu *Straight* sudah lebih kuat daripada kombinasi tingkat tengah yaitu *Three of a kind*. Selain itu, kombinasi tingkat tengah yaitu *Three of a kind* juga sudah lebih kuat daripada tingkat atas yaitu *Single*.

Setelah saya mencoba jalankan program dengan input seperti di atas, hasil kombinasi yang dikeluarkan seperti yang saya ketikkan di atas. Jika dibandingkan, maka :

1. Tingkat bawah dimenangkan program karena *Flush* lebih kuat daripada *Straight*.
2. Tingkat tengah dimenangkan musuh karena *Three of a kind* lebih kuat daripada *Two pairs*.
3. Tingkat atas dimenangkan musuh karena walaupun program dan musuh sama-sama menggunakan kombinasi *Single*, namun kartu terkuat program di tingkat ini lebih lemah daripada kartu terkuat musuh di tingkat ini, yaitu Queen hati lebih lemah daripada King hati, sehingga musuh menang.

Karena program hanya memenangkan tingkat bawah, sedangkan musuh memenangkan tingkat tengah dan atas, maka permainan dimenangkan oleh musuh.

Namun dengan menerapkan trik menguatkan tingkat atas dengan melemahkan tingkat bawah atau tingkat tengah, trik ini mampu merubah hasil permainan. Penerapan trik ini diilustrasikan dengan mengubah kombinasi dari program menjadi sebagai berikut.

1. *Flush* di tingkat bawah
   (As sekop, King sekop, 10 sekop, 7 sekop, 5 sekop)
2. *Pairs* di tingkat tengah
   (Jack keriting, Jack wajik, 7 keriting, 6 wajik, 3 wajik)
3. *Pairs* di tingkat atas
   (8 hati, 8 wajik, Queen hati)

Jika diperiksa, kombinasi ini sudah valid. Hal ini dapat dilihat pada tingkat bawah dengan kombinasi *Flush* sudah lebih kuat daripada tingkat tengah dengan kombinasi *Pairs*. Tingkat tengah dan tingkat atas memiliki kombinasi yang sama yaitu *Pairs*, namun tingkat tengah merupakan *Pairs* dengan kartu terkuat Jack keriting, sedangkan tingkat atas merupakan *Pairs* dengan kartu terkuat 8 hati. Maka dapat disimpulkan tingkat tengah juga sudah lebih kuat daripada tingkat atas.

Sekarang mari kita bandingkan ulang kombinasi-kombinasi program yang sudah dimodifikasi dengan musuh, yaitu :

1. Tingkat bawah dimenangkan program karena *Flush* lebih kuat daripada *Straight*.

2. Tingkat tengah dimenangkan musuh karena *Three of a kind* lebih kuat daripada *Pairs*.
3. Tingkat atas dimenangkan program karena *Pairs* lebih kuat daripada *Single*.

Karena program yang sudah dimodifikasi memenangkan tingkat bawah dan tingkat atas, sementara musuh hanya memenangkan tingkat tengah, maka permainan dimenangkan oleh program hasil modifikasi ini.

Dari ilustrasi di atas, dapat dilihat bahwa kombinasi hasil algoritma *greedy* tidak selalu menghasilkan kombinasi terbaik, namun dapat dikatakan solusi yang dihasilkan merupakan salah satu solusi optimum karena masih dapat dipandang merupakan salah satu kombinasi terbaik. Modifikasi yang dilakukan sangat bergantung pada keberuntungan dan kombinasi musuh untuk meningkatkan kemungkinan kemenangan. Jika tidak beruntung atau kombinasi musuh tidak cocok, maka mungkin modifikasi ini justru dapat membuat kombinasi yang terbentuk kalah dari musuh dari yang tadinya menang.

## VI. Kesimpulan

Algoritma *greedy* sudah mampu menyelesaikan persoalan penentuan kombinasi terbaik pada permainan kartu remi bernama capsa susun dengan baik. Kombinasi yang dihasilkan sudah mampu memberikan salah satu solusi terbaik dari sejumlah kemungkinan solusi yang dapat dibentuk.

Kombinasi-kombinasi lain yang mampu dibentuk mungkin masih memiliki kemungkinan kemenangan lebih tinggi daripada kombinasi yang dibentuk dari algoritma *greedy*, namun hal tersebut tetap bergantung pada bagaimana kombinasi yang dibentuk lawan. Ada suatu waktu di mana kombinasi dari program mampu menang melawan kombinasi musuh tertentu, namun ada juga saat di mana kombinasi dari program perlu dipecah kembali dengan memperkuat tingkat atas sehingga mampu menang melawan kombinasi musuh.

Oleh karena itu, sesungguhnya tidak cukup menentukan kombinasi terbaik dari kartu-kartu yang didapat untuk memenangkan permainan ini. Ada beberapa faktor lain yang dapat menentukan kemenangan, seperti bagaimana musuh menyusun kombinasi mereka. Perbandingan kombinasi antar pemain sangat dipengaruhi bagaimana mereka memperkuat tingkat bawah, tingkat tengah, atau tingkat atas. Keberuntungan dalam menarik kartu juga sangat berpengaruh dalam permainan ini. Jika kombinasi yang didapat sudah merupakan kombinasi terkuat namun kartu yang didapat dari awal sudah buruk, maka kombinasi akhir yang mampu diperoleh pun tetap buruk.


## Ucapan Terima Kasih

Saya mengucapkan terima kasih kepada Tuhan Yang Maha Esa, karena hanya atas karunia-Nyalah maka saya dapat menyelesaikan makalah ini dengan baik dan tepat waktu. Saya juga mengucapkan terima kasih kepada Ibu Masayu Leylia Khodra selaku dosen IF2211 Strategi Algoritma K1, Ibu Nur Ulfa Maulidevi selaku dosen IF2211 Strategi Algoritma K2, dan Bapak Rinaldi Munir selaku dosen IF2211 Strategi Algoritma K3. Terakhir saya juga mengucapkan terima kasih kepada orang tua serta teman-teman saya yang telah mendukung saya dalam pembuatan makalah ini baik dalam wujud perbuatan nyata maupun dalam doa.


## Referensi


[1] https://www.gameloe.com/support/39-game-capsa-susun.html. Diakses pada 9 Mei 2017 pukul 21.40 WIB.

[2] https://bertzzie.com/knowledge/analisis-algoritma/Greedy.html. Diakses pada 9 Mei 2017 pukul 21.50 WIB.

[3] https://www.it-jurnal.com/pengertian-algoritma-greedy/. Diakses pada 9 Mei 2017 pukul 22.10 WIB.

[4] Munir, Rinaldi. 2009. *Diktat Kuliah IF2211 Strategi Algoritma*. Bandung: Institut Teknologi Bandung.

[5] http://i65.tinypic.com/2djuqad.jpg. Diakses pada 11 Mei 2017 pukul 20.30 WIB.

[6] http://4.bp.blogspot.com/-qtJITDdjhJA/UGclPhK0JmI/AAAAAAAAIPw/OEnuy190fmg/s1600/Bolehkah-Main-Catur-Domino-atau-Kartu-Remi-460x250.jpg. Diakses pada 11 Mei 2017 pukul 20.40 WIB.

[7] https://cdn.gameloe.com/files/capsasusun/3-capsa-susun.jpg. Diakses pada 11 Mei 2017 pukul 20.50 WIB.


## 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, 11 Mei 2017

Alvin Sullivan / 13515048