Strategi Penyelesaian Permainan Kartu Tri-Peaks dengan Pendekatan *Branch and Bound*

Daniel Pintara - 13515071
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
nieltansah@students.itb.ac.id

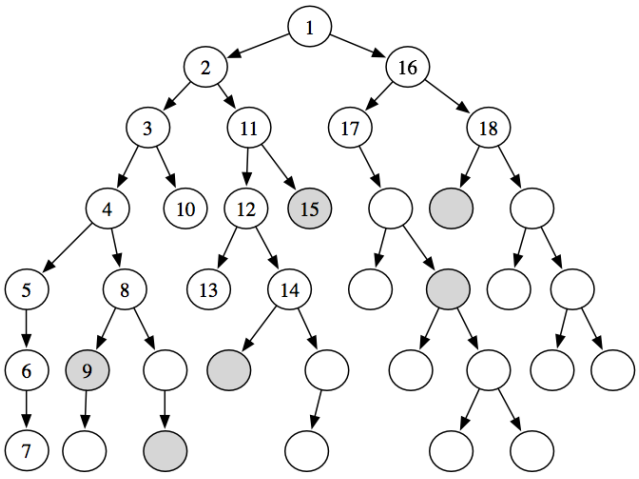
***Abstrak*—Pencarian solusi pada suatu ruang pencarian merupakan suatu kemampuan dasar dan krusial. Salah satu cara untuk melakukan pencarian solusi adalah dengan algoritma *Branch and Bound*. Algoritma ini dapat mencari solusi pada ruang pencarian yang besar dengan cara membatasi ruang pencarian untuk menyelesaikan permainan Tri-Peaks.**

Kata kunci—branch; bound; solusi; permainan Tri-Peaks

I. PENDAHULUAN

A. *Algoritma Branch and Bound*

Kemampuan untuk mencari solusi pada ruang pencarian merupakan salah satu kemampuan dasar yang dimiliki oleh sistem komputasi yang ada sekarang ini. Metode pencarian yang paling sederhana adalah *brute force*, yaitu metode untuk mencari solusi dengan cara mencoba semua kemungkinan yang ada. Namun, sistem komputasi yang ada pada masa ini tidaklah tak terbatas. Komputer memiliki batasan-batasan tersendiri, beberapa diantaranya adalah memori (RAM) yang tersedia sebagai tempat penyimpanan sementara dan kecepatan komputasi dari suatu prosesor. Hal ini menyebabkan proses pencarian solusi menggunakan teknik *brute force* tidak selalu dapat menyelesaikan persoalan dengan waktu dan memori yang wajar. Oleh karena itu, teknik-teknik lain mulai dikembangkan untuk meminimalisir ruang pencarian, sehingga pencarian dilakukan dengan lebih efisien.



Gambar 1 Ilustrasi algoritma *branch and bound*[3]

Untuk menyelesaikan suatu persoalan, maka persoalan itu harus dapat dimodelkan. Salah satu model persoalan yang umum adalah model persoalan graf, yaitu model persoalan yang melibatkan adanya simpul dan sisi, yang terhubung simpul-simpul lainnya. Model persoalan graf merupakan salah satu model yang fleksibel. Banyak persoalan yang dapat dimodelkan dengan menggunakan model graf dengan cara mengasosiasikan simpul dan sisi pada model graf dengan persoalan yang ingin diselesaikan. Salah satu contoh persoalan yang natural jika dimodelkan dengan menggunakan graf adalah persoalan pencarian jarak terdekat diantara 2 buah kota. Untuk persoalan ini, kota dapat dimodelkan sebagai simpul dan jarak antar kota dapat dimodelkan sebagai sisi dari simpul-simpul tersebut, atau dapat disimpulkan bahwa sisi menghubungkan 2 buah kota dengan jarak tertentu. Selain itu, model persoalan graf dapat digunakan untuk pencarian keputusan. Cara yang digunakan adalah mengasosiasikan keadaan (*state*) dengan simpul, transisi dengan sisi. Sebagai contoh, dalam permainan tic-tac-toe sederhana, papan permainan beserta informasi giliran dapat dimodelkan sebagai simpul, sedangkan aksi pada permainan dapat dimodelkan sebagai sisi, sehingga dapat dikatakan bahwa ada suatu keadaan X, jika dikenakan aksi A, maka keadaan berubah menjadi keadaan Y, jika dikenakan aksi B, maka keadaan berubah menjadi keadaan Z, dan sebagainya.

Proses pemodelan dengan menggunakan graf dapat membentuk ruang pencarian yang sederhana, sampai ruang pencarian yang sangat besar. Sebagai contoh, untuk memodelkan persoalan permainan tic-tac-toe, maka diperlukan sisi sebanyak kemungkinan aksi pada suatu keadaan. Papan permainan tic-tac-toe terdiri dari 9 kotak. Jika semua kotak dalam keadaan kosong, maka ada 9 kemungkinan untuk meletakkan suatu tanda pada salah satu dari kotak-kotak tersebut. Jumlah aksi yang dapat terjadi pada suatu keadaan akan menurun, seiring dengan terisinya kotak-kotak itu. Namun, untuk memodelkan persoalan permainan kartu, seperti permainan FreeCell, Klondike, ada banyak sekali kemungkinan yang dapat terjadi pada suatu keadaan. Oleh karena itu, maka algoritma *branch and bound* digunakan untuk menyelesaikan persoalan-persoalan itu.

Algoritma *branch and bound* merupakan salah satu cara yang digunakan untuk meminimalisir ruang pencarian. Algoritma ini bekerja pada model persoalan graf yang dinamis, dengan cara mengasosiasikan atribut baru dalam persoalan, yaitu biaya (*cost*) untuk setiap simpul. Biaya menunjukkan

sedekat apa simpul yang sedang diproses dengan simpul tujuan. Pada dasarnya, algoritma ini mirip dengan algoritma *Breadth-First-Search*. Hal yang membedakan algoritma dengan algoritma *Breadth-First-Search* adalah algoritma ini tidak sebarang mengunjungi simpul, namun simpul dengan biaya tertinggi atau terendah yang akan dikunjungi terlebih dahulu, sehingga proses pencarian menjadi tertuju pada simpul tujuan. Selain itu, komputer tidak perlu untuk menghasilkan anak simpul dari suatu simpul yang belum dikunjungi. Oleh karena itu, algoritma *branch and bound* merupakan salah satu cara yang efisien untuk memfokuskan proses pencarian kepada simpul tujuan.

B. Permainan Kartu Tri-Peaks

Permainan kartu remi, yaitu kartu yang terdiri dari 13 angka: A, 2, …, J, Q, K, serta 4 jenis: hati, sekop, keriting, dan wajik, merupakan salah satu dari beberapa persoalan yang menghasilkan ruang pencarian yang besar, karena jumlah kartu yang digunakan biasanya tidak sedikit. Oleh karena itu, algoritma *branch and bound* merupakan salah satu algoritma yang cocok untuk menyelesaikan persoalan ini. Salah satu permainan kartu remi yang menarik menurut penulis adalah permainan kartu Tri-Peaks.

Gambar 2 Permainan kartu Tri-Peaks

Permainan kartu Tri-Peaks merupakan permainan kartu yang cukup sederhana. Permainan ini tidak peduli terhadap jenis kartu. Pemain hanya boleh mengambil kartu dibawah atau diatas kartu sekarang (J sekop pada ilustrasi gambar), yaitu 10 atau Q. Saat pemain menekan kartu Q keriting pada ilustrasi gambar, maka kartu itu akan menggantikan kartu sekarang. Sehingga untuk aksi berikutnya, pemain hanya dapat mengambil kartu J atau K. Pemain diperbolehkan untuk mengambil kartu yang sama dengan sebelumnya, sebagai contoh, pengambilan kartu J → Q → J merupakan suatu urutan yang diperbolehkan. Jika 2 kartu yang menutupi kartu di atas sudah diambil, maka kartu yang tertutupi itu dapat diambil. Sebagai contoh, jika kartu Q keriting dan 8 sekop diambil, maka kartu diatasnya akan terbuka dan bisa diambil. Jika tidak ada aksi yang memungkinkan pada papan, maka pemain dapat menekan kartu di sebelah kartu sekarang pada ilustrasi gambar. Kartu itu merupakan dek kartu yang akan menggantikan kartu sekarang, sehingga permainan dapat dimainkan kembali. Namun, dek kartu tersebut tidak tak terbatas, sehingga jika dek tidak ada kartu yang dapat diambil dan dek kartu habis, maka permainan tidak dapat diselesaikan. Permainan selesai jika kartu-kartu pada papan terambil semuanya.

Untuk menyelesaikan permainan kartu Tri-Peaks dengan menggunakan komputer, maka permainan ini perlu dimodelkan terlebih dahulu. Model yang dipakai adalah model persoalan graf, dimana simpul merupakan keadaan utuh dari papan permainan Tri-Peaks, dan sisi merupakan aksi yang diambil. Cara ini memungkinkan untuk memodelkan seluruh jalan dari permainan ini. Hal yang menurut penulis menarik adalah menentukan biaya dari setiap simpul. Penentuan biaya merupakan salah satu hal yang penting untuk mengarahkan pencarian ke arah solusi, sehingga pencarian menjadi optimal. Biaya dari setiap simpul penulis tentukan dengan cara memainkan permainan ini berkali-kali sambil mengamati, kira-kira hal apa saja yang membawa permainan kepada solusi, sehingga permainan ini dapat diselesaikan oleh komputer.

II. DASAR TEORI

A. Graf

Graf merupakan diagram yang terdiri dari titik-titik, yang disebut dengan simpul, tersambung dengan garis, yang disebut dengan sisi.[2] Secara matematis, graf merupakan *tuple*:

V(G) = himpunan simpul-simpul (tidak kosong)
V(G) = { v_1, v_2, …, v_n }

E(G) = himpunan sisi, menghubungkan sepasang simpul
E(G) = { e_1, e_2, …, e_n }

Graf dapat dituliskan dalam notasi:

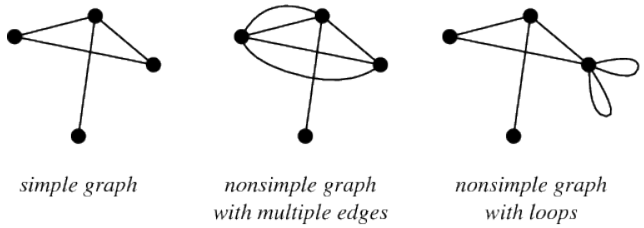
G = (V, E)

Menurut definisi sebelumnya, pada suatu graf G, himpunan V(G) tidak boleh kosong, namun himpunan E(G) boleh kosong. Oleh karena itu, sebuah graf dimungkinkan untuk tidak mempunyai sisi sama sekali, namun simpul harus ada, minimal satu buah. [1]

Graf itu sendiri dapat digolongkan jadi beberapa jenis, bergantung pada sudut pandang cara mengelompokkannya.

1) Pengelompokan Berdasarkan Sisi

Berikut ini merupakan pengelompokan graf berdasarkan jenis sisi dari suatu graf.



Gambar 3 Graf berdasarkan jenis sisi[6]

a) Graf sederhana

Graf sederhana merupakan graf yang tidak mengandung sisi gelang, yaitu sisi yang menunjuk pada simpul sendiri

maupun sisi ganda, yaitu sisi yang menunjuk pada simpul tetangga yang sama, dengan jumlah lebih dari satu.

b) Graf tak-sederhana

Graf tak-sederhana merupakan kebalikan dari graf sederhana, yaitu graf yang memiliki sisi gelang dan disebut sebagai graf semu atau *pseudograph*, ataupun sisi ganda yang disebut sebagai *multigraph*.

2) Pengelompokan Berdasarkan Jumlah Simpul

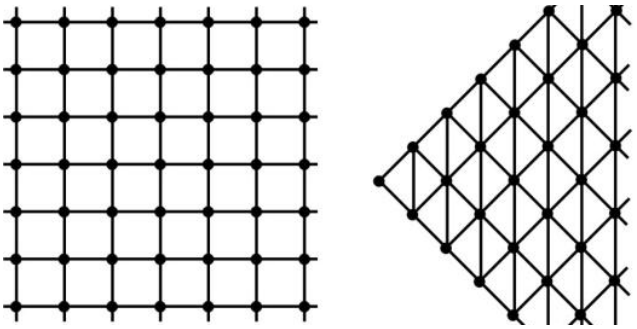
Berikut ini merupakan pengelompokan graf berdasarkan jumlah simpul yang ada pada suatu graf.

a) Graf berhingga

Graf berhingga merupakan graf yang jumlah simpulnya berhingga, yaitu memiliki batas yang jelas.

b) Graf tak-berhingga

Graf tak-berhingga merupakan graf yang memiliki jumlah simpul tak berhingga. Graf ini biasanya memiliki pola tertentu yang berulang-ulang sampai tak berhingga, sehingga jumlah simpul pun menjadi tak berhingga.



Gambar 4 Graf tak berhingga[7]

3) Pengelompokan Berdasarkan Orientasi Sisi

Berikut ini adalah pengelompokkan graf berdasarkan orientasi dari sisi-sisi pada suatu graf.

a) Graf tak-berarah

Graf tak-berarah (*undirected graph*) merupakan graf yang sisinya tidak mempunyai arah, atau tidak mempunyai orientasi arah.

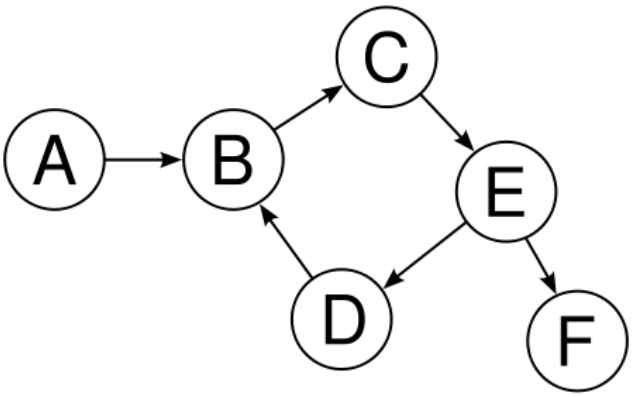
Graf tak-berarah (*directed graph*) dapat disimpan dalam bentuk graf berarah dengan cara menggunakan dua busur untuk setiap sisinya, yaitu busur yang mengarah dari simpul A ke B, dan sebaliknya.

b) Graf berarah

Graf berarah merupakan graf yang memiliki arah pada setiap sisinya, yang biasa disebutkan dengan busur. Pada graf berarah, tidak seperti graf tak-berarah yang ditemukan sebelumnya, sebuah busur dinyatakan sebagai:

E = (v_j, v_k)

Simpul v_j disebutkan sebagai simpul asal (*intial vertex*), sedangkan simpul v_k disebutkan sebagai simpul terminal (*terminal vertex*). Pada gambar, arah ditunjukkan sebagai tanda panah pada busur graf. Sisi ganda tidak diperbolehkan untuk ada pada graf berarah, namun sisi gelang diperbolehkan.



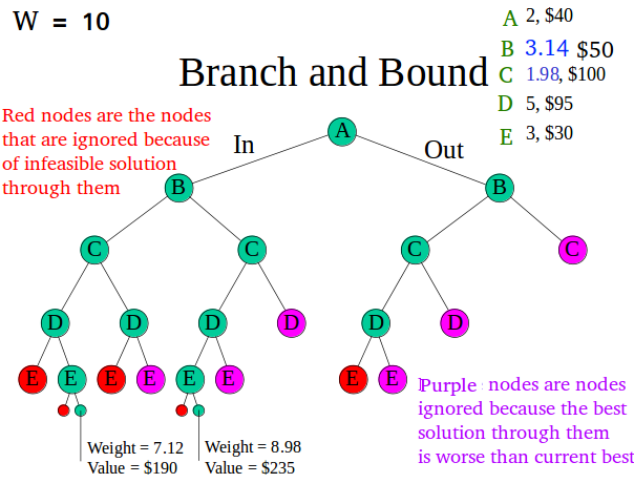
Gambar 5 Graf berarah[8]

Model penyelesaian yang digunakan untuk permainan Tri-Peaks pada makalah ini adalah model persoalan graf sederhana, berhingga, dan berarah.

B. Algoritma Branch and Bound

Algoritma *Branch and Bound* merupakan salah satu cara yang dikembangkan untuk menyelesaikan masalah optimisasi dan kombinatorial diskrit, yaitu masalah untuk mencari kombinasi terbaik dari semua kombinasi yang ada.

Menyelesaikan persoalan optimisasi diskrit *NP-hard* adalah tugas yang tidak mudah, dan memerlukan algoritma yang sangat efisien. Enumerasi semua kemungkinan dalam ruang pencarian tidak dimungkinkan, karena jumlah simpul yang terus bertumbuh secara eksponensial.[4] Oleh karena itu, diperlukan suatu fungsi pembatas untuk membatasi ruang pencarian, sehingga pencarian menjadi lebih optimal.



Gambar 6 Algoritma Branch and Bound[5]

Pada algoritma *Branch and Bound*, terdapat beberapa unsur implementasi, yaitu:

a. *Bounding function*

Fungsi yang digunakan untuk memberikan batas bawah pada ruang solusi, untuk pencarian yang dilakukan.

b. *Expanding strategy*

Strategi untuk memilih simpul untuk diekspansi pada setiap iterasinya.

c. *Branching rule*

Suatu aturan untuk diterapkan jika investigasi tidak dapat diselesaikan pada satu waktu, yaitu dengan cara membagi persoalan menjadi persoalan yang lebih kecil untuk diinvestigasi pada iterasi-iterasi berikutnya.

Berikut ini adalah gambaran program atau *pseudocode* dari algoritma *Branch and Bound*:

```
function BranchBound (initial : Node) → Node
KAMUS
  tasks           : priority queue of Node
  candidate, task : Node
ALGORITMA
  Candidate = NULL
  tasks.push (initial)

  while not tasks.empty ()
    task = tasks.pop ()

    if (not candidate)
        or (task.cost () > candidate.cost ())
      if task.goal ()
        candidate = task

      foreach child in task.childs ()
        tasks.push (child)
    end

    → candidate
  end
```

Dari *pseudocode* di atas, dapat ditarik kesimpulan bahwa setiap simpul akan diekspansi, lalu anak-anak dari simpul tersebut akan dimasukkan pada daftar simpul untuk diekspansi pada iterasi berikutnya. Simpul dengan nilai tertinggi akan diekspansi terlebih dahulu, dan seterusnya.

Jika suatu simpul merupakan simpul tujuan, maka simpul tersebut disimpan sebagai simpul kandidat. Semua simpul dalam daftar simpul yang nilainya tidak lebih dari simpul kandidat akan dibuang dan tidak akan diekspansi. Itu adalah cara algoritma *Branch and Bound* untuk mendapatkan solusi optimal tanpa harus menelusuri seluruh ruang pencarian.

C. *Permainan Kartu Tri-Peaks*

Permainan kartu Tri-Peaks merupakan permainan kartu yang bertujuan untuk menyusun kartu dengan aturan tertentu agar permainan lebih menantang. Seperti yang dibahas sebelumnya, permainan kartu ini terdiri atas:

a. Papan permainan

Papan dengan kartu-kartu yang tersusun sedemikian rupa sehingga berbentuk seperti 3 puncak.

b. Dek kartu

Kartu sebagai pengganti kartu sekarang sehingga permainan dapat terus berjalan.

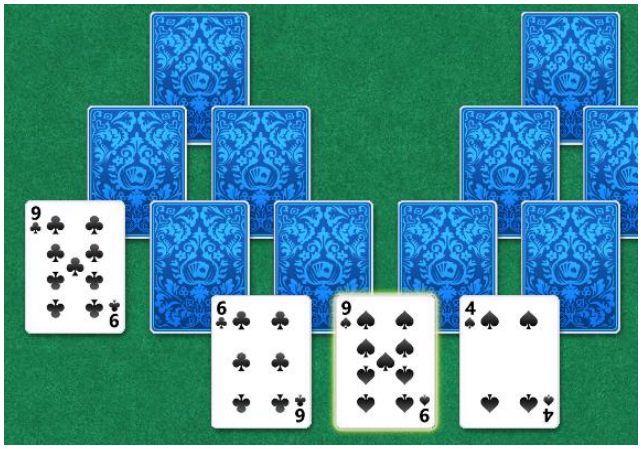
c. Kartu sekarang

Kartu hasil penyusunan.

Keadaan awal dari permainan itu adalah saat semua kartu tertutup, kecuali kartu yang berada pada baris baling bawah. Dek kartu penuh, dan kartu sekarang terdefinisi. Selain itu, ada beberapa aksi yang dapat dilakukan oleh pemain.

1) Memilih kartu pada papan permainan

Pemain dapat memilih kartu yang terbuka pada papan permainan, dengan kiteria bahwa kartu yang dipilih merupakan kartu tetangga, yaitu kartu yang satu lebih tinggi atau lebih rendah dari kartu sekarang. Sebagai contoh, kartu sekarang merupakan 2, maka kartu yang dapat dipilih adalah A atau 3.



Gambar 7 Papan permainan Tri-Peaks

Perlu diketahui bahwa pemilihan kartu tetangga itu bersifat *circular*, sebagai contoh, kartu tetangga dari A merupakan K dan 2, sedangkan kartu tetangga dari K adalah A dan Q.

Aksi ini menyebabkan kartu sekarang berubah menjadi kartu yang dipilih, lalu kartu yang dipilih menghilang. Jika kartu yang terpilih merupakan satu-satunya kartu yang menutupi karu yang ada diatas kartu itu, maka kartu itu menjadi terbuka dan dapat dipilih.

2) Memilih dek kartu

Gambar 8 Dek kartu dari permainan Tri-Peaks

Pemain dapat memilih kartu dek. Aksi ini bertujuan untuk melanjutkan permainan jika tidak ada kartu yang dapat dipilih pada papan permainan. Aksi ini hanya dapat dilakukan jika kartu dek tidak habis.

Aksi ini menyebabkan kartu dek teratas diambil, dan ditempatkan pada kartu sekarang.

Permainan Tri-Peaks berakhir ketika tidak ada gerakan yang dapat dilakukan: tidak ada kartu yang dapat dipilih dari papan permainan dan dek kartu habis, atau ketika semua kartu pada papan permainan habis (solusi).

D. Penerapan Algoritma

Pembuatan model untuk permainan Tri-Peaks tidak mudah. Hal ini dikarenakan terdapat banyak batasan yang tidak boleh dilewatkan. Selain itu, terdapat keterkaitan (*dependency*) penyusunan kartu pada permainan Tri-Peaks, yaitu kartu di belakang akan terbuka jika dan hanya jika dua kartu di depannya sudah diambil.

1) Menentukan Model Papan

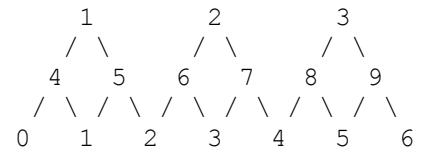
Model susunan papan permainan Tri-Peaks pada makalah ini dibuat dengan cara menghitung jumlah kartu dari jumlah baris, yaitu 4, dan jumlah puncak, yaitu 3, dengan menggunakan deret aritmatika.

Body = **Row** * (**Row** - 1) * **Peak** / 2
Leaf = **Peak** * **Row** - **Peak** + 1

Cards = **Body** + **Leaf**

Setelah jumlah kartu dikalkulasi, setiap tempat kartu pada papan dikaitkan satu dengan lainnya menggunakan iterasi, sehingga proses pengaitan memiliki kompleksitas **O(Cards)**.

Berikut ini adalah susunan Tri-Peaks dengan 3 baris:



Dari diagram di atas, dapat disimpulkan bahwa kartu no. 4 (atas), jika diambil, maka ia akan membuka kartu no. 1 (atas) jika sudah tidak ada kartu lain yang mengunci kartu tersebut, dan sebagainya.

2) Menentukan Model Keadaan

Untuk alasan performansi dan mengurangi redudansi, model keadaan atau *state* pada makalah ini dibedakan dengan model papan. Model keadaan terhubung dengan satu model papan untuk melakukan penghematan memori dan waktu *copy*.

Model keadaan itu sendiri berisi:

a. Keadaan kunci

   Larik berisi jumlah kartu yang diperlukan untuk membuka kunci dari setiap kartu. Oleh karena itu, dapat disimpulkan bahwa:

   Kartu yang masih terkunci memiliki **nilai kunci** > 0.

   Kartu yang sudah terbuka memiliki **nilai kunci** = 0.

   Larik ini juga menyimpan informasi lain mengenai apakah suatu kartu sudah diambil jika **nilai kunci** = -1.

b. Kartu sekarang

   Informasi mengenai kartu sekarang.

c. Posisi kartu dek

   Penunjuk pada daftar kartu dek yang menunjukkan posisi kartu dek mana saja yang sudah diambil dan yang masih tersedia untuk diambil.

3) Ekspansi Simpul

Proses ekspansi simpul dilakukan dengan cara enumerasi terhadap aksi-aksi yang dapat dilakukan pada suatu keadaan, seperti yang telah disebutkan sebelumnya.

a) Ekspansi Simpul untuk Aksi Pilih Kartu

Simpul diekspansi dengan cara mengenumerasi semua kartu yang tidak terkunci (**nilai kunci** = 0), lalu melakukan pemeriksaan, apakah kartu tersebut merupakan tetangga dari kartu sekarang. Jika iya, maka algoritma membuat simpul baru sesuai dengan aksi.

Jika tidak ada kartu yang memenuhi syarat, maka proses ini tidak menghasilkan simpul baru.

b) Ekspansi Simpul untuk Aksi Pilih Dek

Simpul diekspansi dengan cara mengambil kartu pada dek kartu dan meletakkannya sebagai kartu sekarang.

Jika dek kartu kosong, maka proses ini tidak menghasilkan simpul baru.

4) Penentuan Simpul Tujuan

Simpul tujuan ditentukan dengan cara mengenumerasi larik kunci, dan memeriksa apakah semua kartu sudah diambil (**nilai kunci** = -1). Jika ada salah satu kartu yang tidak memenuhi syarat, maka simpul bukan simpul tujuan.

5) Menentukan Biaya atau Cost

Biaya atau *cost* ditentukan melalui proses *research* oleh penulis. Setelah memainkan permainan Tri-Peaks berkali-kali, maka penulis menyimpulkan bahwa keadaan yang dekat dengan solusi merupakan keadaan dengan jumlah kartu yang sedikit pada papan permainan.

Penulis mencoba untuk memberikan nilai lebih pada simpul dengan kiteria tersebut dengan cara mencantumkannya di rumus biaya.

Cost = 2 * (jumlah kartu diambil + jumlah kartu terbuka)

III. Analisa Data dan Hasil Pengamatan

Penulis melakukan percobaan pada komputer penulis dengan spesifikasi Intel® Processor 5Y10 CPU dengan frekuensi dasar 0.80 GHz, dan frekuensi maksimum 2.0 GHz. Penulis melakukan percobaan pada lingkungan Bash on Ubuntu on Windows (LXSS) pada sistem operasi Microsof Windows 10 Creators Update.

A. Eksperimen

Sampel ini diambil dengan cara memainkan permainan Tri-Peaks pada Microsoft Solitaire Collection, lalu menuliskannya.

Setelah memasukkan informasi yang telah dituliskan pada *solver*, penulis mencoba menyelesaikan permainan itu dengan solusi yang sudah ditemukan.

```
         9S           7S           9S
      TS   8S       TS   3S       KS   JS
   9S   AS   QS   3S   2S   QS   4S   QS   AS
 4S   AS   4S   TS   JS   KS   8S   JS   7S   8S
```

Deck kartu:

```
7S 5S 6S 2S 4S 3S TS 2S 7S AS QS 8S
6S 5S 3S KS JS KS 9S 2S 6S 5S 5S
```

Kartu sekarang:

```
65
```

Solusi:

```
       9S           7S           9S
    TS   8S       TS   3S       KS   JS
  9S   AS   QS   3S   2S   QS   4S   QS   AS
4S   AS   4S   TS   JS   KS   8S   JS   7S   8S
Current: 6S

       9S           7S           9S
    TS   8S       TS   3S       KS   JS
  9S   AS   QS   3S   2S   QS   4S   QS   AS
4S   AS   4S   TS   JS   KS   8S   JS       8S
Current: 7S

…

                  7S



Current: 6S




Current: 7S
```

… (solusi: 52 langkah)

Waktu yang diperlukan untuk penyelesaian masalah:

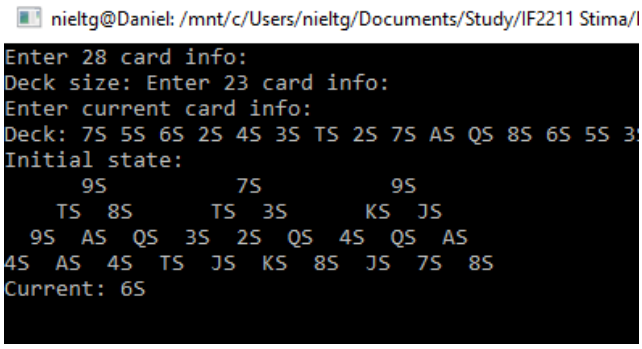
```
real    0m0.099s
user    0m0.016s
sys     0m0.078s
```

B. *Analisa Percobaan*

Penerapan algoritma *Branch and Bound* pada permainan Tri-Peaks dapat dilakukan dengan waktu yang sangat cepat. Fungsi pembatas pada algoritma *Branch and Bound* berhasil dalam meminimalisir ruang pencarian, sehingga solusi ditemukan dengan cepat.

Namun, ada beberapa kelemahan dari algoritma ini. Tidak semua papan permainan Tri-Peaks dapat diselesaikan dengan waktu yang cepat. Sebelum percobaan ini, penulis menemukan papan Tri-Peaks yang sudah penulis coba mainkan, namun tidak menghasilkan solusi. Saat dicoba pada *solver*, program berjalan dan akhirnya penulis hentikan secara paksa setelah 10 menit berjalan.

Penulis berencana untuk melampirkan kode sumber program pada akun Github penulis yang berada pada alamat: https://github.com/nieltg.



Gambar 9 *Solver* untuk Permainan Tri-Peaks

IV. Kesimpulan dan Saran

A. *Kesimpulan*

Algoritma *Branch and Bound* merupakan algoritma yang baik dan efisien untuk menyelesaikan persoalan optimasi dengan ruang solusi yang besar. Fungsi pembatas pada algoritma ini membatasi ruang solusi, sehingga pencarian fokus pada simpul-simpul yang menjanjikan saja, dan meminimalisir waktu pencarian.

Implementasi dari algoritma *Branch and Bound* berhasil menemukan solusi untuk beberapa permainan Tri-Peaks dengan fungsi pembatas yang cukup sederhana, yang bertujuan untuk mengarahkan pencarian kepada permainan dengan kartu paling sedikit, dan akhirnya sampai kepada simpul solusi, dimana tidak ada kartu pada papan permainan.

B. *Saran*

Untuk kedepannya, algoritma ini dapat dikembangkan dan dengan menganalisa permainan Tri-Peaks secara lebih lanjut, bagaimana simpul keadaannya, apakah ada fungsi biaya atau cost yang lebih baik, dan sebagainya.

Salah satu saran dari penulis adalah mencatat semua simpul keadaan yang telah dibuat dan melakukan pemeriksaan, apakah ada simpul yang sama, yang menyebabkan alur pencarian yang tidak diinginkan.

Ucapan Terima Kasih

Penulis mengucapkan terima kasih untuk Drs. Nur Ulfa Maulidevi dan Dr. Ir. Rinaldi Munir, M.T. selaku dosen pada mata kuliah IF2211 Strategi Algoritma. Penulis juga berterima kasih kepada mereka atas bimbingan yang mereka berikan, sehingga penulis dapat menyelesaikan makalah ini dengan baik.

Penulis juga berterima kasih kepada orang tua dan teman-teman penulis atas dukungan moral, serta pihak-pihak lainnya yang sudah berkontribusi dalam pembuatan makalah ini secara langsung maupun tidak langsung.

DAFTAR PUSTAKA

[1] Munir, Rinaldi. 2006. Diktat Kuliah IF2120 Matematika Diskrit, Edisi Keempat. Informatika Bandung : Bandung.

[2] http://scanftree.com/Graph-Theory
Tanggal akses: 19 Mei 2017.

[3] http://artint.info/html/ArtInt_63.html
Tanggal akses: 19 Mei 2017.

[4] http://diku.dk/OLD/undervisning/2003e/datV-optimer/JensClausenNoter.pdf
Tanggal akses: 19 Mei 2017.

[5] http://geeksforgeeks.org/branch-and-bound-set-1-introduction-with-01-knapsack
Tanggal akses: 19 Mei 2017.

[6] http://mathworld.wolfram.com/SimpleGraph.html
Tanggal akses: 19 Mei 2017.

[7] http://jokoprasetyo8.wordpress.com/2015/02/11/graph-hingga-dan-graph-tak-hingga
Tanggal akses: 19 Mei 2017.

[8] http://mrgeek.me/technology/datascience/graph-theory-101-directed-and-undirected-graphs
Tanggal akses: 19 Mei 2017.

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, 19 April 2017

Daniel Pintara
13515071