Penerapan Algoritma Greedy Pada Permainan Papan *Pandemic*

Margaretha Olivia Haryono - 13521071
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 13521071@std.stei.itb.ac.id

***Abstract*—*Pandemic* adalah permainan papan kooperatif yang menggambarkan penyebaran penyakit di seluruh dunia serta usaha pemain untuk mencegah terjadinya wabah global. Salah satu strategi yang dapat digunakan untuk menemukan solusi yang optimal yaitu dengan pendekatan algoritma *greedy*. Algoritma ini akan menentukan aksi yang akan dilakukan berdasarkan jumlah penyakit pada setiap kota. Algoritma *greedy* tersebut diterapkan pada penentuan langkah setiap gilirannya sehingga dapat memaksimalkan potensi kemenangan pada permainan.**

Keywords—greedy, papan permainan, Pandemic board game.

I. PENDAHULUAN

Permainan papan adalah jenis permainan yang dimainkan di atas papan atau permukaan datar lainnya. Permainan papan adalah bentuk permainan hiburan yang populer di kalangan masyarakat. Selain sebagai sarana rekreasi, permainan papan juga dapat digunakan sebagai alat pembelajaran dan eksperimen untuk memahami berbagai konsep dan strategi.

Dalam beberapa tahun terakhir, permainan papan yang berfokus pada tema penyakit dan epidemi telah mendapatkan perhatian yang signifikan. Salah satu permainan papan yang mengangkat tema ini yaitu *Pandemic*. *Pandemic* adalah permainan papan kooperatif dan pertama kali diterbitkan oleh *Z-Man Games* di Amerika Serikat pada tahun 2008. Dalam permainan ini, semua pemain bertindak sebagai anggota tim yang bekerja sama untuk mengendalikan penyebaran penyakit di seluruh dunia serta mencari obat untuk penyakit-penyakit tersebut. Setiap pemain memiliki peran khusus dengan kemampuan unik yang dapat digunakan untuk memerangi wabah global ini.

Pandemic dapat dimainkan oleh dua hingga lima orang yang bekerja sama sebagai satu tim untuk mengobati penyakit di seluruh dunia sambil mengumpulkan sumber daya untuk pengobatannya. Karena pemain bekerja sama sebagai tim, semua pemain akan menang atau kalah secara bersama-sama. Tim dianggap memenangkan permainan ketika dapat menghentikan penyebaran empat penyakit di seluruh peta dunia dengan menemukan obat untuk keempat penyakit tersebut. Selama pencarian obat, "*epidemic*" dan "*outbreak*" dapat terjadi yang meningkatkan tingkat infeksi kota-kota tersebut. Jika infeksi terlalu meluas, maka tim kalah.

Gambar 1.1 Permainan Papan *Pandemic*
(Sumber: https://www.newyorker.com/)

Permainan yang dibuat oleh Matt Leacock ini telah diikuti oleh sejumlah ekspansi dan *spin-off* dalam satu dekade lebih sejak diterbitkannya, terutama trilogi *Pandemic Legacy* yang cukup terkenal. Permainan papan kooperatif seperti *Pandemic* sangat diminati oleh banyak orang saat ini.

Dalam mengoptimasi pengambilan langkah untuk mencapai kemenangan pada permainan ini, dapat diterapkan suatu strategi dalam penentuan aksi yang diambil oleh setiap pemain pada setiap gilirannya. Salah satu algoritma yang sering digunakan dalam situasi optimasi adalah algoritma *greedy*. Algoritma *greedy* mencari solusi yang optimal secara lokal pada setiap langkahnya, dengan memilih tindakan terbaik berdasarkan kriteria tertentu. Dengan demikian, probabilitas kemenangannya dapat ditingkatkan.

II. LANDASAN TEORI

A. *Algoritma Greedy*

Algoritma *greedy* adalah algoritma yang memecahkan persoalan secara langkah per langkah sedemikian sehingga pada setiap langkah mengambil pilihan yang terbaik yang dapat diperoleh pada saat itu tanpa memperhatikan konsekuensi ke depan (optimum lokal) dengan harapan bahwa kumpulan solusi optimal lokal tersebut akan membawa solusi yang optimal global. Algoritma *greedy* merupakan metode yang paling populer dan sedehana untuk memecahkan persoalan optimasi. Terdapat dua macam persoalan optimasi, yaitu maksimasi (*maximization*) dan minimasi (*minimization*).

Pada sebagian persoalan, algoritma *greedy* tidak selalu memberikan solusi yang optimal, namun sub-optimal. Hal ini karena algoritma *greedy* tidak melakukan pencarian penuh

seperti algoritma *brute* force. Oleh karena itu, algoritma *greedy* dapat digunakan untuk menghasilkan solusi hampiran (*approximation*) karena kompleksitas waktunya cukup rendah dibandingkan algoritma *brute force*.

Terdapat beberapa elemen pada algoritma *greedy*, yaitu sebagai berikut.

1. Himpunan kandidat, yaitu himpunan yang berisi kandidat yang akan dipilih pada setiap langkah. Himpunan kandidat diberi simbol C.
2. Himpunan solusi, yaitu himpunan yang berisi kandidat yang sudah dipilih. Himpunan solusi diberi simbol S.
3. Fungsi solusi, yaitu fungsi yang digunakan untuk menentukan apakah himpunan kandidat yang dipilih sudah memberikan solusi.
4. Fungsi seleksi (*selection function*), yaitu fungsi yang digunakan untuk memilih kandidat berdasarkan strategi *greedy* tertentu. Strategi *greedy* ini bersifat heuristik.
5. Fungsi kelayakan (*feasible*), yaitu fungsi yang digunakan untuk memeriksa apakah kandidat yang dipilih dapat dimasukkan ke dalam himpunan solusi (layak atau tidak).
6. Fungsi obyektif, yaitu fungsi yang digunakan untuk memaksimumkan atau meminimumkan solusi yang didapat dari persoalan yang diselesaikan sesuai dengan tujuannya.

Dengan menggunakan elemen-elemen di atas, dapat disimpulkan bahwa algoritma *greedy* melibatkan pencarian sebuah himpunan bagian *S* dari himpunan kandidat *C*, yang dalam hal ini, *S* harus memenuhi beberapa kriteria yang ditentukan, yaitu *S* menyatakan suatu solusi dan *S* dioptimisasi oleh fungsi obyektif.

Berikut merupakan skema umum dari algoritma *greedy* yang ditulis dalam *pseudocode*.

```
function greedy(C : himpunan_kandidat) → himpunan_solusi
{ Mengembalikan solusi dari persoalan optimasi dengan algoritma greedy }

Deklarasi
  x : kandidat
  S : himpunan_solusi

Algoritma
  { inisialisasi S dengan kosong }
  S ← {}

  while (not SOLUSI(S)) and (C ≠ {} ) do
    { pilih sebuah kandidat dari C }
    x ← SELEKSI(C)
    { buang x dari C karena sudah dipilih }
    C ← C – {x}
    { x memenuhi kelayakan untuk dimasukkan ke dalam himpunan solusi }
    if LAYAK(S ∪ {x}) then
      { masukkan x ke dalam himpunan solusi }
      S ← S ∪ {x}
    endif
  endwhile

  { SOLUSI(S) or C = {} }

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

Terdapat beberapa persoalan yang dapat diselesaikan dengan algoritma *greedy*. Beberapa diantaranya yaitu persoalan penukaran uang, pemilihan aktivitas, minimisasi waktu di dalam sistem, *knapsack problem*, penjadwalan *job* dengan tenggat waktu, pohon merentang minimum, lintasan terpendek, kode Huffman, dan pecahan Mesir.

B. Permainan Papan Pandemic

Satu set permainan papan *Pandemic* terdiri dari beberapa bagian yang dibutuhkan untuk memainkan permainan, yaitu:

- 1 papan yang menggambarkan peta dunia dengan koneksi antar kota
- 5 pion pemain
- 6 pion stasiun penelitian (*research station*)
- 6 penanda (4 *cure*, 1 *outbreak*, 1*infection rate*)
- 96 *disease cubes* (terdapat 4 warna dengan jumlah 24 untuk setiap warna)
- 7 kartu peran atau *role*
- 59 kartu pemain (48 kartu *city*, 6 kartu *epidemic*, dan 5 kartu *event*)
- 4 kartu *reference*
- 48 kartu *infection*

Sebelum memulai permainan, letakkan satu *research station* di Atlanta serta penanda *outbreak* pada angka 0 di *outbreak track*. Letakkan juga *infection rate* pada angka 2. Kemudian, lakukan *infect cities* dengan cara membalik tiga kartu teratas dari tumpukan kartu *infection* dan menempatkan tiga *disease cubes* dari setiap warna kartu di kota yang ditunjukkan. Lakukan kembali langkah ini dengan dua *disease cubes*, dan kemudian satu *disease cubes*, sehingga terdapat total 18 *disease cubes* yang berada di papan. Setiap kartu yang telah dipakai harus menghadap ke atas di tumpukan kartu yang sudah dibuang.

Selanjutnya, setiap pemain akan mendapatkan satu kartu *role* secara acak. Setiap kartu *role* memiliki kekuatan yang berbeda dan dapat membantu keberjalanan permainan. Berikut adalah daftar *role* yang tersedia.

- *Dispatcher*, *role* ini dapat memindahkan pion pemain lain seolah-olah pion miliknya sebagai suatu aksi. Mereka juga dapat memindahkan bidak apa pun ke bidak lain tanpa memainkan kartu (juga sebagai aksi).
- *Operations expert*, *role* ini dapat membangun stasiun penelitian tanpa memainkan kartu (dianggap sebagai aksi).
- *Scientist*, *role* ini dapat menyembuhkan penyakit dengan empat kartu dengan warna yang sama. Hal ini tentu saja membutuhkan aksi dan *research station*.
- *Medic*, *role* ini dapat menghilangkan semua kubus dengan satu warna saat mengobati penyakit. Hal ini tidak dihitung sebagai suatu aksi.
- *Researcher*, *role* ini dapat memberikan kartu apa saja kepada pemain lain di kota yang sama. *Researcher* juga dapat melakukan hal ini pada giliran pemain lain.

Setiap giliran pemain akan melakukan ketiga hal berikut.

1. Melakukan 4 aksi
2. Mengambil 2 kartu *player*
3. Melakukan *infect cities*

Berikut adalah daftar aksi yang dapat dilakukan pemain.

- *Drive / Ferry*, yaitu berpindah ke kota yang terkoneksi dengan kota dimana pemain berada saat ini.
- *Direct Flight*, yaitu membuang kartu *city* untuk pindah ke kota tersebut secara langsung.
- *Charter Flight*, yaitu membuang kartu *city* yang sesuai dengan kota dimana pemain berada saat ini untuk pindah ke kota manapun.
- *Shuttle Flight*, yaitu berpindah dari kota yang memiliki *research station* ke kota lain yang juga memiliki *research station*.

Terdapat juga aksi-aksi spesial yang dapat dilakukan sebagai berikut.

- *Build a Research Station*, yaitu membuang kartu *city* yang sama dengan kota tempat pemain berada untuk menempatkan *research station* di tempat tersebut.
- *Treat Disease*, yaitu menyembuhkan 1 *disease cubes* di kota tempat pemain berada. Jika obat untuk penyakit warna tersebut telah ditemukan, pemain dapat menghapus semua kubus dengan warna tersebut dari kota sebagai satu aksi.
- *Share Knowledge*, aksi ini dapat dilakukan dengan dua cara, yaitu memberikan atau mengambil kartu *city* dari pemain lain. Untuk melakukannya, kedua pemain harus berada di kota yang sama dengan kartu yang ingin diberikan atau diambil.
- *Discover a Cure*, yaitu pada kota yang memiliki *research station*, pemain dapat membuang 5 kartu *city* dengan warna yang sama dari tangan pemain untuk menyembuhkan penyakit dengan warna tersebut.

Permainan berakhir setelah pemain menemukan obat untuk keempat penyakit yang ditandai dengan warna yang berbeda-beda. Ketika kondisi ini tercapai, maka seluruh pemain telah memenangkan permainan. Meskipun hanya ada satu cara untuk memenangkan permainan, terdapat tiga cara untuk kalah, yaitu:

- Ketika penanda *outbreak* mencapai ruang terakhir di *outbreak track*.
- Ketika pemain perlu meletakkan *disease cubes* tetapi *disease cubes* dengan warna tersebut telah habis.
- Ketika tumpukan kartu pemain telah habis.

Jika salah satu dari ketika kondisi tersebut tercapai, maka seluruh pemain kalah dari permainan *Pandemic* ini.

III. PEMBAHASAN

A. Asumsi dan Batasan

Permainan papan *Pandemic* yang sebenarnya sangatlah kompleks. Untuk penyederhanaan dalam algoritma *greedy* yang akan diterapkan untuk pengambilan keputusan aksi yang dilakukan, penulis membuat asumsi dan batasan pada beberapa faktor sebagai berikut.

1. Representasi *Game State*

Permainan *Pandemic* yang sebenarnya memiliki representasi *game state* yang jauh lebih kompleks. Hal ini mencakup berbagai elemen seperti posisi *disease cubes* pada papan, status setiap kota (misalnya terinfeksi atau *outbreak*), peran/*role* pemain, kartu pada setiap pemain, tingkat infeksi, stasiun penelitian, penyakit yang disembuhkan, dan lain sebagainya. Elemen-elemen ini berinteraksi satu sama lain dan berkontribusi pada keseluruhan kompleksitas dan pengambilan keputusan pada permainan. Algoritma *greedy* yang akan dianalisis serta implementasikan hanya berfokus pada evaluasi kartu individu tanpa mempertimbangkan elemen yang lebih luas.

2. Aksi yang Tersedia

Dalam permainan *Pandemic* yang sebenarnya, pemain memiliki berbagai aksi yang dapat mereka pilih saat giliran mereka. Aksi ini termasuk menyembuhkan penyakit untuk menghilangkan penyakit dari kota, berpindah antar kota, berbagi pengetahuan dengan pemain lain, membangun stasiun penelitian, menemukan obat, menggunakan kemampuan khusus berdasarkan peran/*role* pemain, dan sebagainya. Setiap aksi memiliki kekuatan dan pertimbangannya sendiri. Namun, algoritma *greedy* yang akan diimplementasikan hanya mengevaluasi aksi memainkan kartu yang ada di tangan pemain serta tidak mempertimbangkan berbagai aksi lain yang tersedia dalam permainan.

3. Kriteria Evaluasi Aksi

Algoritma *greedy* yang akan diimplementasikan menggunakan kumpulan kriteria yang disederhanakan untuk mengevaluasi aksi yang dipilih. Algoritma ini terutama mempertimbangkan jumlah *disease cubes* pada

kota tujuan serta warna kartu yang dimainkan. Meskipun faktor-faktor ini penting dalam proses pengambilan langkah, terdapat berbagai hal lain yang berperan dalam keberjalanan permainan. Dalam permainan sebenarnya, pemain perlu mempertimbangkan juga urgensi terjadinya *outbreak* penyakit yang berbeda, konsentrasi *disease cubes* pada berbagai wilayah, ketersediaan sumber daya, jarak posisi pemain dengan kota yang terdampak, potensi terjadinya *outbreak* di masa depan, dan lain sebagainya. Kriteria evaluasi yang diterapkan tidak memperhitungkan pertimbangan-pertimbangan ini.

4. Interaksi Antar Pemain

Permainan *Pandemi*c yang sebenarnya menekankan kerjasama dan koordinasi antar pemain. Pemain perlu menyusun strategi dan mengoordinasikan aksi mereka untuk mengoptimalkan upaya mereka sebagai satu tim. Namun, algoritma *greedy* yang diimplementasikan tidak mempertimbangkan interaksi antar pemain. Faktor kooperatif ini menambah tingkat kompleksitas pada proses pengambilan keputusan yang tidak tercakup pada algoritma.

B. Implementasi Algoritma Greedy Pada Permainan

Untuk menerapkan algoritma *greedy* pada permainan papan *Pandemic*, perlu ditentukan terlebih dahulu tujuan dari permainan ini, yang adalah menemukan obat untuk keempat penyakit yang ada sebelum penyakit tersebut menginfeksi lebih banyak kota. Oleh karena itu, kita akan memprioritaskan aksi-aksi yang memiliki dampak yang paling besar dalam mencegah penyebaran penyakit. Berikut adalah langkah-langkah dalam penerapan algoritma *greedy* pada permainan ini.

1. Mengidentifikasi kota dengan jumlah infeksi terbesar

Langkah pertama yaitu mengidentifikasi kota-kota yang paling banyak terinfeksi. Kondisi ini dapat dilihat dari jumlah *disease cubes* pada setiap kota. Kota-kota dengan infeksi terbesar memiliki risiko terjadinya *outbreak* tertinggi, sehingga harus diprioritaskan dalam hal menerima sumber daya. Dengan berfokus pada kota yang paling banyak terinfeksi, pemain dapat mencegah penyebaran penyakit lebih lanjut.

2. Menentukan aksi yang paling berharga

Setelah kota yang paling terinfeksi telah diidentifikasi, pemain harus menentukan aksi paling berharga yang dapat diambil untuk mencegah penyebaran penyakit. Misalnya, pemain mungkin perlu pindah ke kota lain untuk mengobati penyakit, mencegah terjadinya *outbreak*, atau membangun *research station* untuk membantu menemukan obat. Dengan mengidentifikasi tindakan yang paling berharga, pemain dapat memfokuskan upaya mereka pada tindakan yang akan memberikan dampak terbesar.

3. Memprioritaskan aksi yang berharga tersebut pada kota dengan jumlah infeksi terbesar

Setelah mengidentifikasi tindakan atau aksi yang paling berharga, pemain harus memprioritaskan aksi tersebut di kota yang paling terinfeksi. Misalnya, jika sebuah kota berisiko terkena *outbreak*, pemain harus memprioritaskan mencegah terjadinya *outbreak*. Dengan berfokus pada tindakan paling berharga di kota yang paling banyak terinfeksi, pemain dapat mencegah penyebaran penyakit dengan lebih efektif.

4. Mengulangi proses

Setelah melakukan aksi paling berharga pada kota-kota yang paling terinfeksi, proses tersebut harus diulangi untuk mengidentifikasi kota-kota baru yang paling terinfeksi dan aksi paling berharga yang dapat diambil pada kota-kota tersebut. Proses ini memastikan bahwa pemain selalu berfokus pada area paling kritis di papan dan mengambil aksi yang paling berdampak.

Dengan menggunakan langkah-langkah yang telah dijelaskan di atas, maka dapat ditentukan elemen-elemen algoritma *greedy* pada permainan papan *Pandemic* yaitu sebagai berikut.

- Himpunan kandidat, yaitu himpunan seluruh kemungkinan aksi yang dapat dilakukan pada keadaan permainan saat itu.
- Himpunan solusi, yaitu himpunan aksi yang dipilih untuk solusi yang final.
- Fungsi solusi, yaitu memilih aksi terbaik dari himpunan kandidat berdasarkan nilai prioritas penggunaan aksi tersebut.
- Fungsi seleksi, yaitu memilih aksi terbaik berdasarkan nilai tiap aksi. Nilai ini dihitung berdasarkan keadaan permainan saat itu, yaitu jumlah *disease cubes* pada kota destinasi serta pada warna yang sama dengan kartu yang sedang dimainkan.
- Fungsi kelayakan, yaitu menentukan kelayakan aksi pilihan fungsi seleksi sehingga tidak melanggar aturan permainan.
- Fungsi obyektif, yaitu menentukan aksi yang dipilih dengan tujuan memaksimalkan nilai aksi serta meminimalkan banyaknya *disease cubes*.

Secara umum, strategi *greedy* yang diterapkan disini berdasarkan jumlah *disease cubes* yang terdapat pada setiap kota. Strategi ini bertujuan untuk memprioritaskan tindakan yang paling cepat dan berdampak untuk mencegah penyebaran penyakit, sehingga meningkatkan peluang pemain dalam menemukan obat untuk keempat penyakit sebelum penyakit tersebut menyebar ke kota-kota lain secara tidak terkendali.

Berikut adalah *pseudocode* beberapa fungsi utama dalam mengimplementasikan algoritma *greedy* pada permainan *Pandemic* untuk menentukan aksi yang dipilih oleh pemain.

1. Fungsi *get_possible_actions*

Fungsi *get_possible_actions* bertanggung jawab untuk menentukan aksi-aksi yang dapat dilakukan berdasarkan *game state* tertentu. Himpunan aksi ini merupakan himpunan kandidat pada elemen algoritma *greedy* yang diterapkan. Fungsi ini akan mengecek kondisi-kondisi

tertentu pada *game state* sebagai kriteria penambahan setiap aksi yang dapat dilakukan sebagai berikut.

- Menambahkan aksi "treat disease" jika kota lokasi pemain memiliki minimal satu *disease cubes*
- Menambahkan aksi "drive/ferry" untuk semua kota yang terkoneksi
- Menambahkan aksi "direct flight" untuk semua kartu di tangan pemain
- Menambahkan aksi "charter flight" jika pemain memiliki kartu yang sama dengan kota lokasinya saat ini
- Menambahkan aksi "build research station" jika kota lokasi pemain belum memiliki research station dan pemain memiliki kartu kota tersebut pada tangannya
- Menambahkan aksi "share knowledge" untuk semua pemain lain yang berada di lokasi yang sama dengan pemain saat ini

```
function get_possible_actions(game_state)
{ Mengembalikan semua kemungkinan aksi yang dapat dilakukan }

  { Mengambil kota dan kartu yang ada di tangan pemain }
  player_city = game_state.current_player.location
  player_hand = game_state.current_player.hand

  { Menambahkan aksi "treat disease" }
  if (sum(disease_cubes) in player_city) > 0 then
    possible_actions.append({'type': 'treat_disease', 'location':
    player_city})
  endif

  { Menambahkan aksi "drive/ferry" }
  for (neighbor_city in game_state.map.player_city) do
    possible_actions.append({'type': 'drive_ferry', 'destination':
    neighbor_city})
  endfor

  { Menambahkan aksi "direct flight" }
  for (card in player_hand) do
    possible_actions.append({'type': 'direct_flight', 'card':
    card})
  endfor

  { Menambahkan aksi "charter flight" }
  if (player_city in player_hand) then
    possible_actions.append({'type': 'charter_flight', 'card':
    player_city})
  endif

  { Menambahkan aksi "build research station" }
  if (player_city.research_station == 0 and player_city in
  player_hand) then
    possible_actions.append({'type': 'build_research_station',
    'location': player_city})
  endif
  { Menambahkan aksi "share knowledge" }
  for (other_players in player_city) do
    possible_actions.append({'type': 'give_knowledge',
    'recipient': other_players, 'donor': current_player})

  return possible_actions
```

2. Fungsi *calculate_action_value*

Fungsi *calculate_action_value* bertanggung jawab untuk menentukan nilai dari suatu aksi pada *game state* tertentu. Fungsi ini bertindak sebagai fungsi obyektif yang menghitung nilai aksi dengan strategi *greedy* secara heuristik. Fungsi ini menghitung banyaknya *disease cubes* pada setiap kota pada himpunan kota tujuan. Kemudian, dilakukan pengecekan apakah terdapat *disease cubes* dengan warna yang sama pada kartu yang dimainkan. Jika ada, maka nilai tersebut akan disimpan. Jika tidak, maka akan bernilai 0. Selanjutnya, dilakukan perhitungan nilai aksi secara heuristik berdasarkan jumlah *disease cubes*.

Nilai aksi dihitung dengan formula jumlah *cubes* dikurangi dengan dua kali jumlah *cubes* dengan warna yang sama dengan kartu yang dimainkan. Hal ini dilakukan karena tujuan dari permainan adalah meminimalkan jumlah *disease cubes* pada papan. Dengan menggunakan formula tersebut, perhitungannya dapat memberikan nilai yang lebih besar untuk kedua aksi berikut.

- Aksi yang melibatkan penggunaan kartu dengan warna yang berbeda
- Aksi yang mengurangi jumlah *disease cubes* dengan warna yang sama

Dengan menerapkan teknik heuristik tersebut dalam penentuan nilai, aksi yang diprioritaskan adalah aksi-aksi yang memiliki dampak yang lebih besar pada pengurangan jumlah *disease cubes* dan hal ini sesuai dengan tujuan permainan secara keseluruhan.

```
function calculate_action_value(game_state, action)
{ Mengembalikan nilai dari setiap aksi yang dapat dilakukan }

  { Mengambil nilai banyaknya disease cubes untuk setiap
  warna pada kota destinasi }
  if ('destination' in action) then
    destination_city = action.destination
  endif
  destination_cubes = destination_city.disease_cubes

  { Menghitung jumlah disease cubes untuk semua warna }
  total_cubes = sum(destination_cubes.values())

  { Mengambil jumlah disease cubes dengan warna yang sama
  dengan kartu yang dimainkan (jika ada) }
  card_color = action.get('card_color')
  if (card_color) then
    same_color_cubes = destination_cubes.get(card_color, 0)
  else
    same_color_cubes = 0
  endif
```

```
    { Menghitung nilai aksi secara heuristik }
    value = total_cubes - 2 * same_color_cubes

    return value
```

3. Fungsi *select_best_action*

Fungsi *select_best_action* bertanggung jawab dalam memilih aksi terbaik pada himpunan aksi yang dapat dilakukan. Fungsi ini bertindak sebagai fungsi solusi yang bertugas untuk memilih aksi terbaik pada himpunan aksi yang dapat dilakukan. Fungsi ini memanggil fungsi *calculate_action_value* untuk melakukan perhitungan nilai untuk setiap aksi yang dapat dilakukan pada himpunan *possible_actions*. Kemudian, akan dipilih aksi dengan nilai paling besar, yaitu aksi yang memiliki dampak positif paling besar untuk mencapai tujuan permainan, yaitu meminimalkan jumlah *disease cubes* di papan.

```
function select_best_action(game_state, possible_actions)
{ Mengembalikan aksi yang dipilih berdasarkan perhitungan
algoritma greedy secara heuristik }

  { Menghitung nilai setiap aksi yang dapat dilakukan }
  action_values = []
  for (action in possible_actions) do
    value = calculate_action_value(game_state, action)
    action_values.append(value)
  endfor

  { Memilih aksi dengan nilai terbesar }
  best_action = max(action_values)

  return best_action
```

Dengan melakukan perhitungan nilai setiap aksi serta memilih aksi dengan nilai terbesar pada setiap iterasi permainan, pembuatan keputusan dengan cara ini mengikuti pendekatan secara *greedy* dengan mengambil langkah yang optimal secara lokal pada setiap pilihannya. Algoritma ini tidak memperhitungkan konsekuensi jangka panjang ataupun kemungkinan aksi di masa depan, namun hanya terfokus pada keadaan permainan saat ini.

IV. EKSPERIMEN

Untuk menguji algoritma *greedy* yang telah dijelaskan pada bagian sebelumnya, dilakukan pengujian dengan cara membuat suatu *initial game state* atau keadaan permainan awal yang telah disederhanakan dari permainan aslinya. Berikut adalah salah satu keadaan permainan awal yang dibuat.

```
Initial Game State:
CURRENT PLAYER: Player 1
RESEARCH STATIONS: {'Atlanta': 1}
OUTBREAKS: 0
DISEASE CUBES:
Atlanta
    blue: 0
    red: 1
Chicago
    blue: 0
    red: 1
Essen
    blue: 0
    red: 0
London
    blue: 2
    red: 0
PLAYER LOCATIONS:
   Player 1: Atlanta
   Player 2: Chicago
```

Selanjutnya, fungsi *get_possible_actions* dipanggil untuk menentukan himpunan aksi yang dapat dilakukan dari keadaan permainan saat itu. Dari keadaan tersebut, didapat kumpulan aksi yang valid yaitu sebagai berikut.

```
Possible Actions:

1. {'type': 'treat_disease', 'location': 'Atlanta'}
2. {'type': 'drive_ferry', 'destination': 'Chicago'}
3. {'type': 'drive_ferry', 'destination': 'Essen'}
4. {'type': 'drive_ferry', 'destination': 'London'}
5. {'type': 'direct_flight', 'card': 'Essen'}
6. {'type': 'direct_flight', 'card': 'Madrid'}
7. {'type': 'direct_flight', 'card': 'London'}
8. {'type': 'direct_flight', 'card': 'Chicago'}
```

Setiap aksi pada himpunan aksi valid tersebut akan dihitung nilainya dengan menerapkan algoritma *greedy* secara heuristik dengan memanggil fungsi *calculate_action_value*. Tabel berikut menunjukkan hasil perhitungan untuk setiap aksi pada himpunan aksi tersebut.

Aksi	Total *cubes*	Jumlah *cubes* (warna sama)	Nilai akhir
Treat disease (Atlanta)	1	0	1
Drive ferry (Chicago)	1	0	1
Drive ferry (Essen)	0	0	0
Drive ferry (London)	2	0	2
Direct flight (Essen)	0	0	0
Direct flight (Madrid)	0	0	0
Direct flight (London)	0	0	0
Direct flight (Chicago)	0	0	0

Tabel 4.1 Perhitungan nilai aksi dengan algoritma *greedy*

Pada tabel di atas, dapat dilihat bahwa aksi *drive ferry* ke London memiliki nilai paling besar. Hal ini karena *disease cubes* pada kota tersebut paling tinggi dibandingkan kota lainnya. Sehingga, menyembuhkan penyakit pada kota tersebut merupakan prioritas utama. Prioritas ini sesuai dengan objektif dari algoritma yang diterapkan yang adalah tujuan dari permainan, yaitu meminimalkan jumlah *disease cubes* yang terdapat pada papan. Dengan demikian, aksi *drive ferry* ke

London, yang berarti berpindah lokasi ke London, adalah aksi yang dipilih karena merupakan pilihan terbaik. Dengan keberadaan pemain di London, pemain tersebut dapat menyembuhkan penyakit di kota tersebut pada giliran berikutnya. Sehingga, *state* permainan selanjutnya adalah sebagai berikut.

```
Current Game State:
CURRENT PLAYER: Player 2
RESEARCH STATIONS: {'Atlanta': 1}
OUTBREAKS: 0
DISEASE CUBES:
Atlanta
    blue: 0
    red: 1
Chicago
    blue: 0
    red: 1
Essen
    blue: 0
    red: 0
London
    blue: 2
    red: 0
PLAYER LOCATIONS:
    Player 1: London
    Player 2: Chicago
```

Algoritma *greedy* ini akan diterapkan pada setiap iterasi permainan. Sehingga, aksi yang dipilih pada setiap langkah permainan selalu merupakan aksi terbaik yang dapat dilakukan sesuai dengan keadaan permainan saat itu.

V. KESIMPULAN

Algoritma *greedy* dapat diimplementasikan pada pengambilan langkah atau keputusan dalam berbagai permainan, salah satunya yaitu dalam permainan papan *Pandemic*. Strategi *greedy* yang dapat diterapkan pada permainan ini yaitu dengan mengidentifikasi kota yang paling terinfeksi, menentukan tindakan yang paling berharga, memprioritaskan tindakan tersebut di kota yang paling terinfeksi, dan mengulangi proses tersebut. Dengan mengikuti strategi ini, pemain dapat mengoptimalkan penggunaan sumber daya dan meningkatkan peluang menemukan obat untuk keempat penyakit. Sehingga, probabilitas kemenangan pemain menjadi lebih besar, meskipun algoritma ini tidak selalu mendapatkan solusi paling optimal secara global serta menjamin kemenangan.

PRANALA VIDEO YOUTUBE

Untuk memberi gambaran yang lebih jelas mengenai isi dari makalah ini, telah dibuat sebuah video penjelasan yang dapat diakses melalui tautan: https://youtu.be/WykylE_y_Pw

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tuhan Yang Maha Esa atas berkat dan rahmat-Nya sehingga makalah ini dapat diselesaikan tepat waktu. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi, M.T. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma Semester II 2022/2023 Kelas 01 yang telah memberikan bimbingan dan pengajarannya selama satu semester ini. Tidak lupa penulis juga mengucapkan terima kasih kepada orang tua yang senantiasa memberikan dukungan selama proses pendidikan penulis, serta seluruh pihak yang telah membantu dan mendukung penulis dalam menyelesaikan makalah ini.

DAFTAR REFERENSI

[1] Z-Man Games. 2020. "The World Of Pandemic" https://www.zmangames.com/en/games/pandemic/

[2] Dicebreaker. 2020. "How to play Pandemic: board game's rules, setup and how to win explained" https://www.dicebreaker.com/games/pandemic/how-to/how-to-play-pandemic-board-game

[3] Munir, Rinaldi. 2021. "Algoritma Greedy (Bagian 1)". https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Algoritma-Greedy-(2021)-Bag1.pdf

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, 22 Mei 2023

Margaretha Olivia Haryono 13521071