Pemecahan Persoalan *Dad's Puzzler Diagonal* Menggunakan Algoritma A*

Isabella Handayani Sumantri 13519081
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
13519081@std.stei.itb.ac.id

***Abstract*—Penentuan rute terpendek merupakan salah satu topik yang hangat dibahas dalam dunia informatika karena banyak kegunaannya pada kehidupan sehari-hari. Algoritma A* sebagai salah satu algoritma penentuan rute terpendek turut menyumbang banyak manfaat. Salah satu manfaatnya adalah untuk memecahkan persoalan *dad's puzzler* diagonal. Pada makalah ini, akan dijelaskan detail pemecahan persoalan *dad's puzzler* diagonal menggunakan algoritma A*.**

Keywords—A*; Algoritma; Rute Terpendek; Dad's Puzzler

I. PENDAHULUAN

Persoalan rute terpendek adalah persoalan yang kerap dibahas dalam dunia informatika. Topik tersebut sering sekali dibahas karena terdapat banyak manfaat yang dapat diperoleh dari pembelajaran persoalan tersebut. Adapun beberapa algoritma yang kerap digunakan untuk menyelesaikan persoalan tersebut antara lain *Depth First Search(DFS)*, *Uniform Cost Search(*UCS), *Greedy Best First Search*, A*. Dari beberapa algoritma tersebut, algoritma A* merupakan algoritma yang paling mangkus karena pendekatannya yang berbeda.

Salah satu pemanfaatan algoritma penentuan rute terpendek pada dunia nyata adalah untuk memecahkan *dad's puzzler*. *Dad's puzzler* adalah sebuah *sliding block puzzle* yang bertujuan untuk memindahkan sebuah blok spesifik ke lokasi yang sudah ditentukan. *Puzzle* ini pertama kali dipatenkan oleh Lewis. W. Hardy pada tahun 1909 di Amerika Serikat. Pada *Dad's puzzler*, pemain harus menentukan rute perpindahan sebuah blok untuk mencapai lokasi tujuan. Untuk itu, permainan ini dapat diselesaikan dengan algoritma penentuan rute seperti algoritma A*. Pada makalah ini, akan dijelaskan detail pemecahan persoalan *Dad's Puzzler* menggunakan algoritma A*.

II. LANDASAN TEORI

A. Algoritma A*

Algoritma A* merupakan algoritma penentuan rute yang kerap digunakan untuk menentukan jalur terpendek. Algoritma ini pertama kali dikemukakan oleh Peter Hart, Nils Nilsson, dan Bertram Raphael pada tahun 1968. Algoritma A* termasuk algoritma *informed search* karena terdapat *heuristic* atau informasi tambahan yang memperkirakan *cost* hingga simpul tujuan. Ide dari algoritma A* adalah menghindari melakukan ekspan pada simpul yang "mahal". *Heuristic* pada algoritma A* akan melakukan estimasi nilai dari simpul. Adapun fungsi evaluasi *heuristic* pada algoritma A* dapat dinyatakan dengan

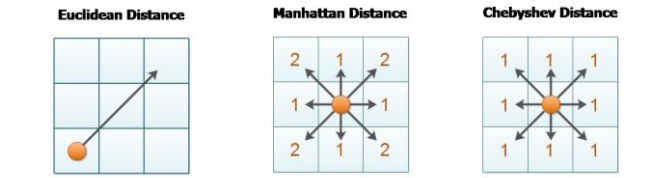
$$f(n) = g(n) + h(n)$$

Fungsi $g(n)$ adalah fungsi yang digunakan untuk mencari *cost* lintasan dari simpul asal ke simpul ke-*n*. Fungsi $h(n)$ adalah fungsi yang digunakan untuk mengestimasi *cost* dari simpul ke-*n* sampai ke simpul tujuan. Algoritma A* akan melakukan *ekspan* terhadap simpul hidup dengan nilai fungsi evaluasi terkecil.

```
Put node_start in the OPEN list with f(node_start) = h(node_start) (initialization)
while the OPEN list is not empty {
  Take from the open list the node node_current with the lowest
       f(node_current) = g(node_current) + h(node_current)
  if node_current is node_goal we have found the solution; break
  Generate each state node_successor that come after node_current
  for each node_successor of node_current {
    Set successor_current_cost = g(node_current) + w(node_current, node_successor)
    if node_successor is in the OPEN list {
      if g(node_successor) ≤ successor_current_cost continue (to line 20)
    } else if node_successor is in the CLOSED list {
      if g(node_successor) ≤ successor_current_cost continue (to line 20)
      Move node_successor from the CLOSED list to the OPEN list
    } else {
      Add node_successor to the OPEN list
      Set h(node_successor) to be the heuristic distance to node_goal
    }
    Set g(node_successor) = successor_current_cost
    Set the parent of node_successor to node_current
  }
  Add node_current to the CLOSED list
}
if(node_current != node_goal) exit with error (the OPEN list is empty)
```

Gambar 1 Pseudocode Algoritma A*
(Sumber : https://mat.uab.cat/~alseda/MasterOpt/AStar-Algorithm.pdf)

Terdapat beberapa fungsi *heurisistic* yang umum untuk digunakan,



Gambar 2 Fungsi-fungsi heuristic
(Sumber : https://iq.opengenus.org/euclidean-vs-manhattan-vs-chebyshev-distance/)

1. *Euclidian distance*
   Fungsi *heuristic euclidean distance* akan menghitung jarak(*d)* antara dua titik menggunakan fungsi berikut.
   $$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$
   Perhitungan jarak memanfaatkan koordinat kartesian kedua titik dan menghitungnya menggunakan teorema pythagoras.

2. *Manhattan distance*
   Fungsi *heuristic manhattan distance* akan menghitung jarak(*d*) antara dua titik menggunakan fungsi berikut,
   $$d = |x_1 - x_2| + |y_1 - y_2|$$
   Perhitungan jarak dilakukan dengan menjumlahkan selisih antara absis dan ordinat kedua titik.

3. *Great circle distance*
   Fungsi *heuristic great circle distance* akan menghitung jarak*(d*) antara dua titik menggunakan fungsi berikut,
   $$d = \max(|x_1 - x_2|, \quad |y_1 - y_2|)$$
   Perhitungan jarak dilakukan dengan mengambil nilai maksimal antara selisih absis dan selisih ordinat kedua titik.

Sebuah fungsi *heuristic* $h(n)$ disebut *admissible* jika untuk semua nilai *n* dipenuhi
$$h(n) \leq h^*(n)$$
$h^*(n)$ adalah nilai sesungguhnya untuk mencapai *goal state* dari *n*. Sebuah fungsi *heuristic* yang *admissible* tidak pernah *overestimate cost* untuk mencapai *goal*.

B. *Dad's Puzzler*

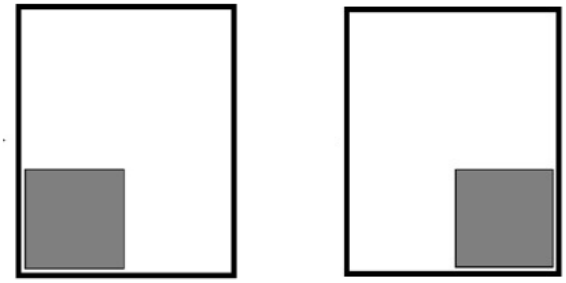
Gambar 3 Dad's Puzzler
(Sumber:https://www.cs.brandeis.edu/~storer/JimPuzzles/ZPAGES/zzzDadsPuzzler.html)

Dad's puzzler adalah sebuah *sliding block puzzle* yang bertujuan untuk memindahkan sebuah blok spesifik ke lokasi yang sudah ditentukan. *Sliding block puzzle* mulai dikenal pada tahun 1880 ketika semua orang berduyun-duyun untuk memecahkan sebuah *puzzle* yang kemudian dikenal sebagai 15 *puzzle*. Perkembangan berikutnya dari *sliding block puzzle* terjadi di tahun 1909 ketika Lewis W. Hardy menciptakan *sliding puzzle* pertama menggunakan bagian yang berbentuk *rectangular. Puzzle* tersebut diberi nama *Pennant Puzzle* yang kemudian akan dikenal sebagai *Dad's Puzzler*.

Terdapat tiga varian *dad's puzzler*,

1. *Dad's Puzzler Normal*
2. *Dad's Puzzler* Diagonal
3. *Dad's Puzzler Exhange*

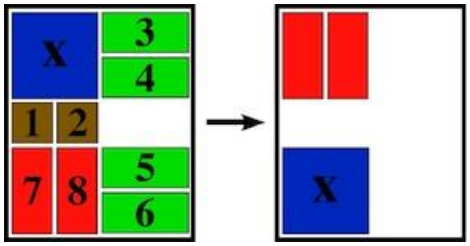
Perbedaan *dad's puzzler* normal dan *dad's puzzler* diagonal terletak pada *final state* dari *puzzle*.



Gambar 4 Final State Dad's Puzzler Normal(kiri) dan Diagonal(kanan)
(Adaptasi dari:
https://www.cs.brandeis.edu/~storer/JimPuzzles/ZPAGES/zzzDadsPuzzler.html)

Sedangkan khusus untuk *Dad's Puzzler Exchange*, terdapat batasan-batasan tambahan terhadap *final state* dari *puzzle*. *Dad's puzzler exchange* dibagi lagi menjadi tiga jenis yaitu,

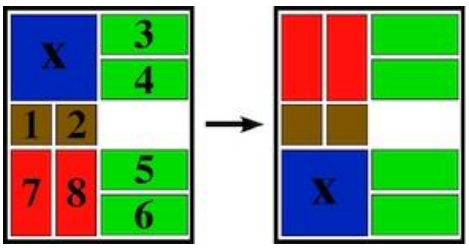
1. *Dad's Puzzler Exchange*



Gambar 5 Dad's Puzzler Exchange
(Sumber :
https://www.cs.brandeis.edu/~storer/JimPuzzles/ZPAGES/zzzDadsExchangeInfantsHospital.html)

Salah satu contoh *dad's puzzler exchange* dapat dilihat di Gambar 5. Pada contoh tersebut blok *2x2* harus dipindahkan ke bagian bawah kiri dan blok 7 dan 8 harus dipindahkan ke bagian kiri atas dan boleh ditukar.

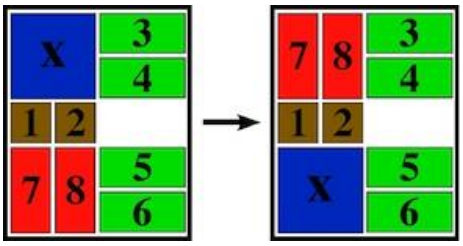
2. *Dad's Puzzler Exchange-Full*



Gambar 6 Dad's Puzzler Exchange-Full
(Sumber :
https://www.cs.brandeis.edu/~storer/JimPuzzles/ZPAGES/zzzDadsExchangeInfantsHospital.html)

Salah satu contoh *dad's puzzler exchange-full* dapat dilihat di Gambar 6. Pada contoh tersebut blok *2x2* harus dipindahkan ke bagian bawah kiri dan blok-blok lain disesuaikan dengan *final state* pada gambar.

3. *Dad's Puzzler Exchange-Full-Strict-All*

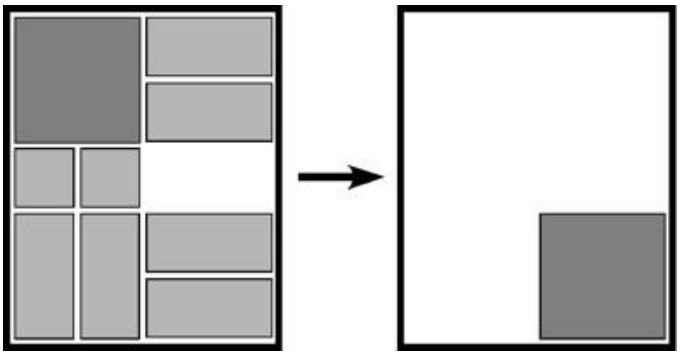


Gambar 7 Dad's Puzzler Exchange-Full-Strict-All
(Sumber :
https://www.cs.brandeis.edu/~storer/JimPuzzles/ZPAGES/zzzDadsExchangeInfantsHospital.html)

Salah satu contoh *dad's puzzler exchange-full-strict-all* dapat dilihat di Gambar 7. Pada contoh tersebut semua blok harus dipindahkan sesuai dengan *final state* pada gambar.

III. Analisis dan Pengujian

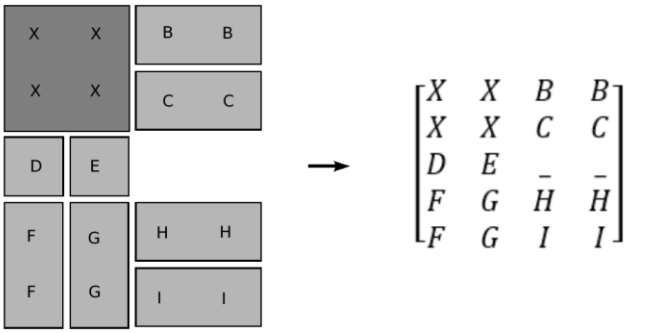
A. *Mapping* Persoalan



Gambar 8 Contoh *Start State* dan *Final State*
(Sumber:https://www.cs.brandeis.edu/~storer/JimPuzzles/ZPAGES/zzzDadsPuzzler.html)

Persoalan *Dad's Puzzler* diagonal dipecahkan dengan mencari langkah-langkah yang mungkin untuk memindahkan blok-blok yang menghalangi blok yang ingin dipindahkan(*goal block)* dan langkah-langkah tersebut memiliki *cost* yang dibuat seminimal mungkin. Contoh dari *start state* dan *final state* ditunjukkan pada Gambar 4 Contoh *Start State* dan *Final State*. *Final state* dari persoalan *dad's puzzler* tidak memedulikan posisi blok selain blok yang ingin dipindahkan.

Persoalan *dad's puzzler* dapat disederhanakan ke dalam bentuk matriks seperti berikut,

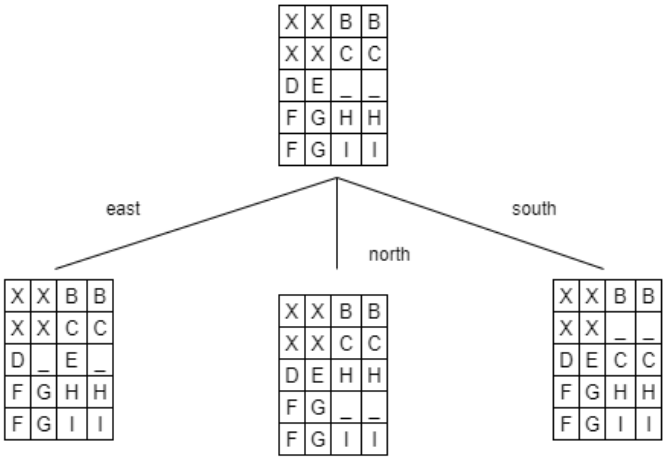


Gambar 9 Matriks Persoalan Dad's Puzzler
(Adaptasi dari :
https://www.cs.brandeis.edu/~storer/JimPuzzles/ZPAGES/zzzDadsPuzzler.html)

Karakter "X" menandakan blok yang ingin dipindahkan ke lokasi yang ditentukan. Karakter "-" menandakan posisi yang kosong pada *puzzle*. Sedangkan karakter-karakter lain akan menandakan blok-blok lain yang menghalangi blok yang ingin dipindahkan.

Setiap simpul merupakan sebuah *state* pada *puzzle*. Simpul-simpul hidup yang dihasilkan adalah *state* yang dihasilkan dari pergerakan simpul-simpul yang bertetangga

dengan tempat kosong. Urutan ekspan adalah (*east, north, west, south*).



Gambar 10 Ekspansi Simpul Akar
(Sumber : Dokumentasi Pribadi)

Pohon ruang status yang dibuat merupakan penggambaran pembangkitan dari simpul akar. Akan tetapi, pohon ruang status tersebut belum menerapkan fungsi evaluasi untuk menentukan *cost* tiap-tiap simpul. Untuk itu, akan ditentukan terlebih dahulu fungsi evaluasi yang akan digunakan pada persoalan *dad's puzzler* diagonal.

B. Fungsi Evaluasi

Fungsi evaluasi dibutuhkan untuk menentukan simpul hidup yang akan diekspan pada algoritma A*. Sebuah fungsi evaluasi $f(n)$ terdiri atas fungsi $g(n)$ dan fungsi $h(n)$. Sebelum persoalan mulai diselesaikan, akan ditentukan terlebih dahulu fungsi $g(n)$ dan fungsi $h(n)$ pada persoalan *dad's puzzler*.

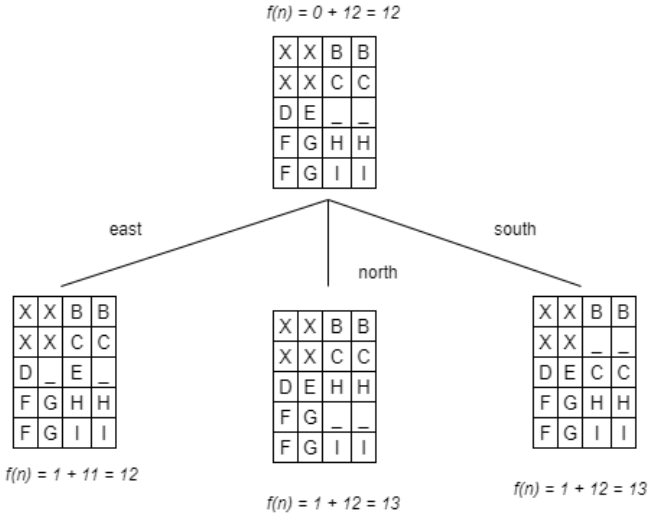
Nilai dari fungsi $g(n)$ dapat diperoleh dari jumlah langkah yang telah ditempuh sebelumnya. Untuk fungsi *heuristic* dari persoalan akan dilakukan analisis terhadap tiga fungsi *heuristic* yang umumnya digunakan. *Euclidean distance* dan *chebyshev distance* merupakan fungsi *heuristic* yang tepat untuk digunakan jika terdapat pergerakan secara diagonal. Akan tetapi, persoalan *dad's puzzler* tidak memungkinkan adanya pergerakan secara diagonal sehingga kedua cara perhitungan jarak tersebut tidak dapat digunakan sebagai fungsi *heuristic* persoalan *dad's puzzler*.

Pergerakan pada *dad's puzzler* terjadi secara horizontal dan vertikal. Oleh karena itu, fungsi *heuristic* yang cocok untuk digunakan adalah fungsi *heuristic* berbasis *manhattan distance*. Fungsi *heuristic* berbasis *manhattan distance* yang akan digunakan adalah

$$h(n) = jumlah\ tile\ yang\ menghalangi\ goal\ block$$

Nilai *heuristic* ini termasuk nilai *heuristic* yang *admissible* karena untuk mencapai *final state* langkah minimum yang harus dilakukkan sebanding dengan banyaknya *tile* yang menghalangi. Sebuah *tile* disebut menghalangi jika

1. Memiliki nilai *y* lebih kecil dari semua *tile* dari blok yang ingin dipindahkan ke lokasi spesifik(*goal block*) tetapi tidak sebaris dengan dua tile kosong yang bersebelahan lansung dengan *goal block*
2. Merupakan sebuah *tile* kosong yang tidak bersebelahan langsung dengan *goal block*



Gambar 11 Contoh Perhitungan Fungsi Evaluasi
(Sumber : Dokumentasi Pribadi)

Contoh perhitungan dari Gambar 11 pergerakan *east*, nilai $g(n) = 1$ karena langkah yang dilakukan 1. Selanjutnya nilai $h(n)$ diperoleh dengan menjumlahkan *tile-tile* yang menghalangi yaitu

$$h(n) = F + G + H + I + D + E + \_\ di\ (4,3)$$
$$h(n) = 2 + 2 + 2 + 2 + 1 + 1 + 1$$
$$h(n) = 11$$

C. Langkah-langkah Pemecahan Masalah

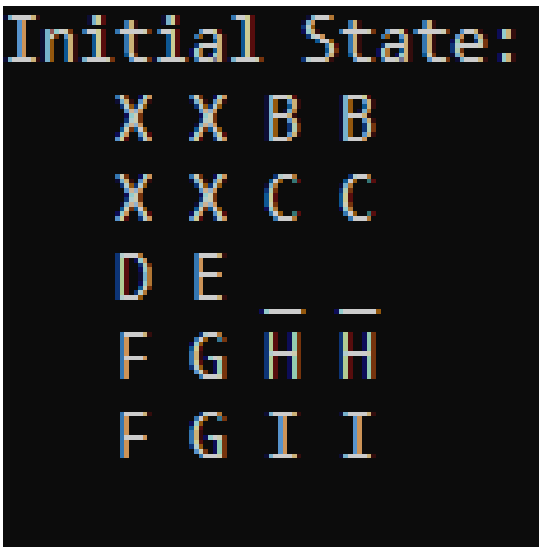
Setelah *dad's puzzler* di-*hardcode* pada program dalam bentuk matriks, dibuat terlebih dahulu algoritma A* yang akan melakukan pencarian rute. Berikut ini adalah langkah-langkah penyelesaian *dad's puzzler* diagonal,

1. Melakukan konversi pada *dad's puzzler* diagonal kemudian menyimpannya pada sebuah *list* di program sebagai *initial state* dari *puzzle*
2. Memasukkan *initial state* ke sebuah *list* bernama OPEN yang akan menyimpan semua *state* yang belum diekspan
3. Mengambil *state* pada *list OPEN* dengan *cost* terkecil, kemudian mengecek apakah *state* tersebut merupakan *final state* atau bukan
4. Jika *state merupakan* final state, aka dibunuh semua simpul hidup yang memiliki *cost* lebih besar dari *state* tersebut
5. Jika bukan, akan dibangkitkan simpul-simpul anaknya. Simpul anak yang akan dibangkitkan berdasarkan nilai terkecil fungsi evaluasi.
6. Kemudian, menghapus *state* yang sudah diekspan dan menyimpannya dalam *list CLOSE*

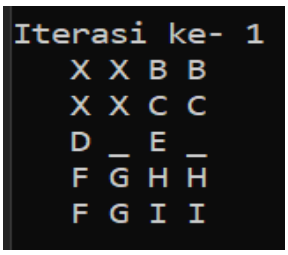
7. Mengulangi langkah 3 – langkah 6 hingga *state* pada *list OPEN* habis.
8. Mencetak ke layar rute terpendek yang dihasilkan selama penelusuran

D. Eksperimen

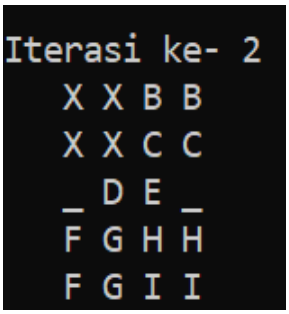
Eksperimen dilakukan dengan menggunakan *dad's puzzler* diagonal dengan *start state* dan *final state* seperti Gambar 5.

Initial State	
	
Goal Block	X
Goal Location	(3,1), (3,2), (4,1), (4,2)

Dengan memanfaatkan algoritma A* akan diperoleh rute terpendek dari persoalan *dad's puzzler* diagonal sebagai berikut,

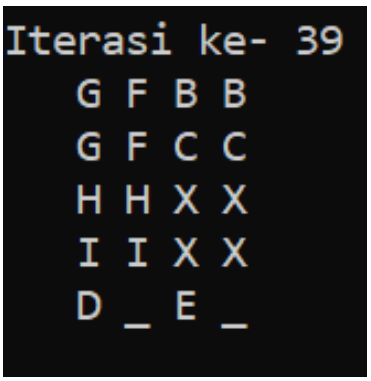


Gambar 12 Langkah Pertama Solusi
(Sumber : Dokumentasi Pribadi)

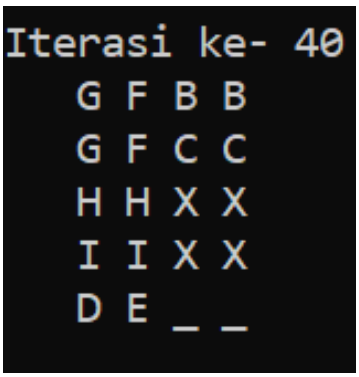


Gambar 13 Langkah Kedua Solusi
(Sumber : Dokumentasi Pribadi)

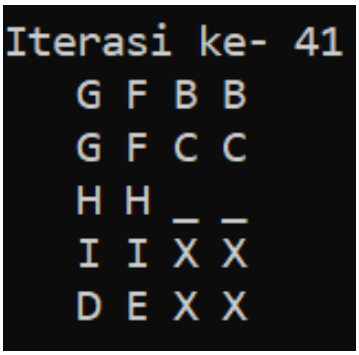
Untuk menghemat tempat, hanya akan ditampilkan 3 langkah awal dan 3 langkah akhir dari solusi



Gambar 14 Langkah ke-39 Solusi
(Sumber : Dokumentasi Pribadi)



Gambar 15 Langkah ke-40 Solusi
(Sumber : Dokumentasi Pribadi)



Gambar 16 Langkah ke-41
Solusi
(Sumber : Dokumentasi Pribadi)

Berikut adalah langkah-langkah yang dilakukan untuk memecahkan persoalan *dad's puzzler* diagonal,

```
Move tile E to East
Move tile D to East
Move tile E to East
Move tile D to East
Move tile X to South
Move tile B to West
Move tile B to West
Move tile C to North
Move tile D to North
Move tile D to East
Move tile X to East
Move tile F to North
Move tile F to North
Move tile G to West
Move tile I to West
Move tile H to West
Move tile E to South
Move tile D to South
Move tile E to South
Move tile D to South
Move tile X to East
Move tile F to East
Move tile G to North
Move tile G to North
Move tile H to West
Move tile D to West
Move tile I to West
Move tile D to South
Move tile X to South
Move tile C to South
Move tile B to East
Move tile B to East
Move tile F to North
Move tile G to North
Move tile H to North
Move tile I to North
Move tile D to West
Move tile E to West
Move tile D to West
Move tile E to West
Move tile X to South
```

IV. Kesimpulan

Algoritma A* adalah algoritma penentuan rute yang mangkus dalam melakukan pemecahan persoalan. Pada makalah ini, persoalan *dad's puzzler* diagonal telah berhasil diselesaikan menggunakan algoritma A* menggunakan fungsi *heuristic* berbasis *manhattan distance* yang menghitung jumlah *tile* yang menghalangi *goal block*. Telah diperoleh bahwa langkah-langkah yang dihasilakn mampu menyelesaikan *dad's puzzler* diagonal secara efektif.

Video Link at Youtube

Untuk memberikan memberikan gambarkan tentang persoalan yang diangkat dan solusinya terlampir pranala video penjelasan berikut,

https://www.youtube.com/watch?v=EavxgtrDZPg

Acknowledgment

Penulis ingin memanjatkan rasa syukur kepada Tuhan yang Maha Esa atas rahmat penyertaan-Nya sehingga makalah ini dapat diselesaikan. Penulis juga ingin mengucapkan terima kasih yang sebesar-besarnya atas pengajaran yang diberikan oleh Dr. Nur Ulfa Maulidevi, ST., M.Sc. selaku dosen mata kuliah IF2211 Strategi Algoritma.

References

[1] Munir, Rinaldi, Route Planning Bagian 2, https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2020-2021/Route-Planning-Bagian2-2021.pdf. Diakses tanggal 10 Mei 2021, pukul 11.29 WIB

[2] Storer, J.A., Dad's Puzzler, https://www.cs.brandeis.edu/~storer/JimPuzzles/SLIDE/DadsPuzzler/DadsPuzzler.pdf. Diakses tanggal 10 Mei 2021, pukul 12.01 WIB

[3] Patel, Amit, Heuristics for Grid Maps, http://theory.stanford.edu/~amitp/GameProgramming/Heuristics.html#heuristics-for-grid-maps. Diakses tanggal 10 Mei 2021, pukul 12.43 WIB

[4] Tim Penulis, A* Search Algorithm ,https://isaaccomputerscience.org/concepts/dsa_search_a_star#:~:text=The%20A*%20algorithm%20uses%20a,node%20and%20the%20target%20node. Diakses tanggal 10 Mei 2021, pukul 13.05 WIB

[5] Tim Penulis, Euclidean vs Manhattan vs Chebyshev Distance, https://iq.opengenus.org/euclidean-vs-manhattan-vs-chebyshev-distance/. Diakses tanggal 10 Mei 2021, pukul 15.39 WIB

[6] Kelly, John, How Sliding Puzzles Work, https://entertainment.howstuffworks.com/puzzles/sliding-puzzles1.htm. Diakses tanggal 10 Mei 2021, pukul 17.25 WIB

[7] Storer, J.A., Dad's Puzzler Exchange Version, https://www.cs.brandeis.edu/~storer/JimPuzzles/ZPAGES/zzzDadsExchangeInfantsHospital.html. Diakses tanggal 11 Mei 2021, pukul 12.46 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.

Bekasi, 11 Mei 2021

Isabella Handayani Sumantri
13519081