

Tugas Kecil 3 IF2211 Strategi Algoritma
Semester II tahun 2024/2025

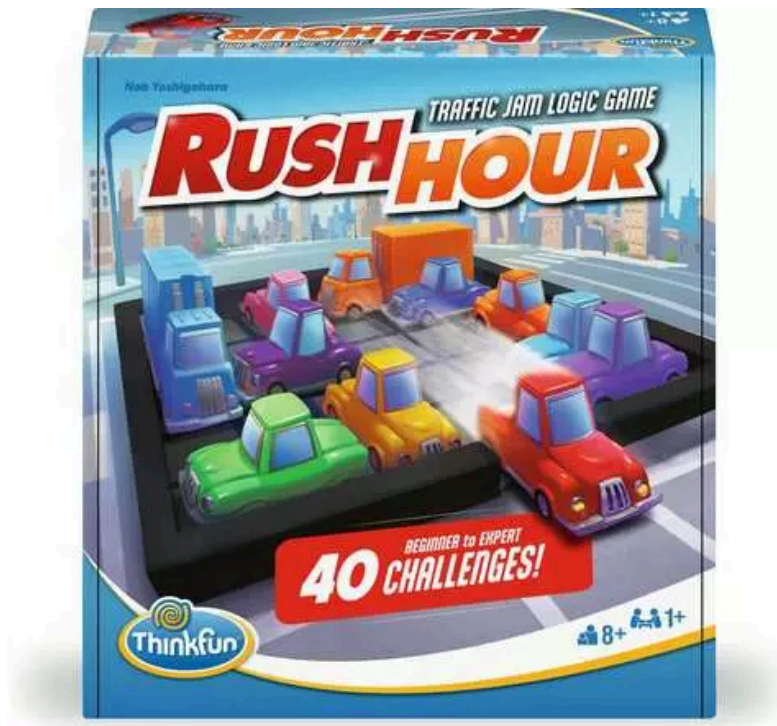
Penyelesaian Puzzle Rush Hour Menggunakan Algoritma Pathfinding

Batas pengumpulan : Hari Rabu, 21 Mei 2025 pukul 12.00 WIB

Arsip pengumpulan :

- *Source* program yang dapat dijalankan disertai README
- Laporan (*soft copy*)

Deskripsi Tugas:



Gambar 1. Rush Hour Puzzle

(Sumber: <https://www.thinkfun.com/en-US/products/educational-games/rush-hour-76582>)

Rush Hour adalah sebuah permainan puzzle logika berbasis grid yang menantang pemain untuk menggeser kendaraan di dalam sebuah kotak (biasanya berukuran 6x6) agar mobil utama (biasanya berwarna merah) dapat keluar dari kemacetan melalui pintu keluar di sisi papan. Setiap kendaraan hanya bisa bergerak lurus ke depan atau ke belakang sesuai dengan

orientasinya (horizontal atau vertikal), dan tidak dapat berputar. Tujuan utama dari permainan ini adalah memindahkan mobil merah ke pintu keluar dengan jumlah langkah seminimal mungkin.

Komponen penting dari permainan Rush Hour terdiri dari:

1. **Papan** – *Papan* merupakan tempat permainan dimainkan.

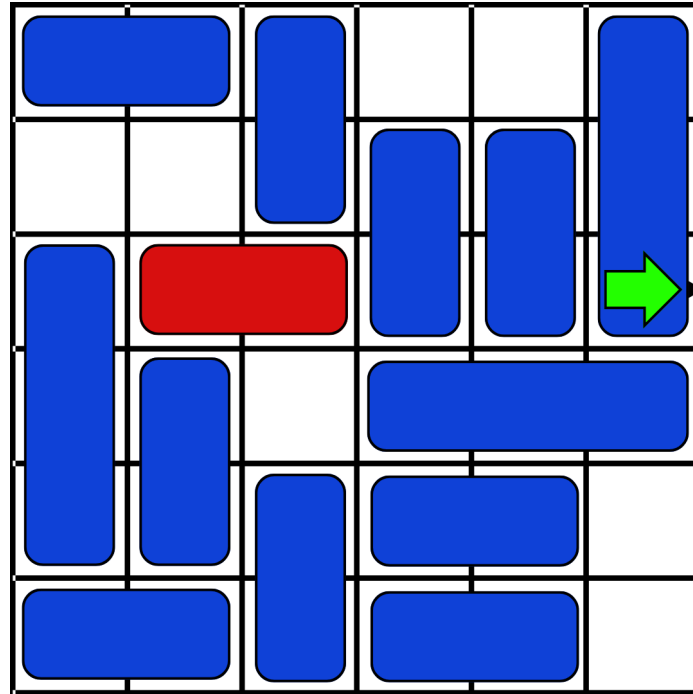
Papan terdiri atas *cell*, yaitu sebuah *singular point* dari papan. Sebuah *piece* akan menempati *cell-cell* pada papan. Ketika permainan dimulai, semua *piece* telah diletakkan di dalam papan dengan konfigurasi tertentu berupa lokasi *piece* dan *orientasi*, antara *horizontal* atau *vertikal*.

Hanya **primary piece** yang dapat digerakkan **keluar papan melewati pintu keluar**. *Piece* yang bukan *primary piece* tidak dapat digerakkan keluar papan. Papan memiliki satu *pintu keluar* yang pasti berada di *dinding papan* dan sejajar dengan orientasi *primary piece*.

2. **Piece** – *Piece* adalah sebuah kendaraan di dalam papan. Setiap *piece* memiliki *posisi*, *ukuran*, dan *orientasi*. *Orientasi* sebuah *piece* hanya dapat berupa horizontal atau vertikal–tidak mungkin diagonal. *Piece* dapat memiliki beragam *ukuran*, yaitu jumlah *cell* yang ditempati oleh *piece*. Secara standar, variasi *ukuran* sebuah *piece* adalah 2-*piece* (menempati 2 *cell*) atau 3-*piece* (menempati 3 *cell*). Suatu *piece* tidak dapat digerakkan melewati/menembus *piece* yang lain.
3. **Primary Piece** – *Primary piece* adalah kendaraan utama yang harus dikeluarkan dari *papan* (biasanya berwarna merah). Hanya boleh terdapat satu *primary piece*.
4. **Pintu Keluar** – *Pintu keluar* adalah tempat *primary piece* dapat digerakkan keluar untuk menyelesaikan permainan
5. **Gerakan** — *Gerakan* yang dimaksudkan adalah pergeseran *piece* di dalam permainan. *Piece* hanya dapat bergerak/bergeser lurus sesuai orientasinya (atas-bawah jika vertikal dan kiri-kanan jika horizontal). Suatu *piece* tidak dapat digerakkan melewati/menembus *piece* yang lain.

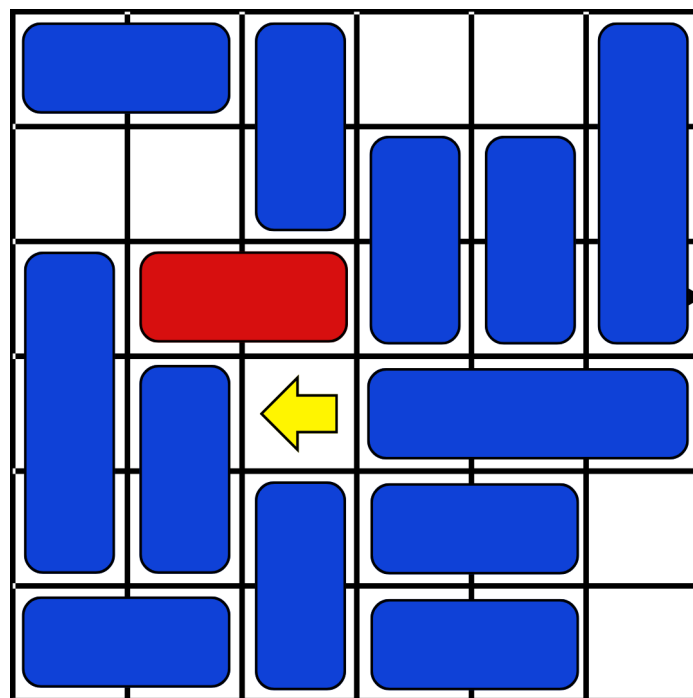
Ilustrasi kasus :

Diberikan sebuah *papan* berukuran 6 x 6 dengan 12 *piece* kendaraan dengan 1 *piece* merupakan *primary piece*. *Piece* ditempatkan pada *papan* dengan posisi dan orientasi sebagai berikut.

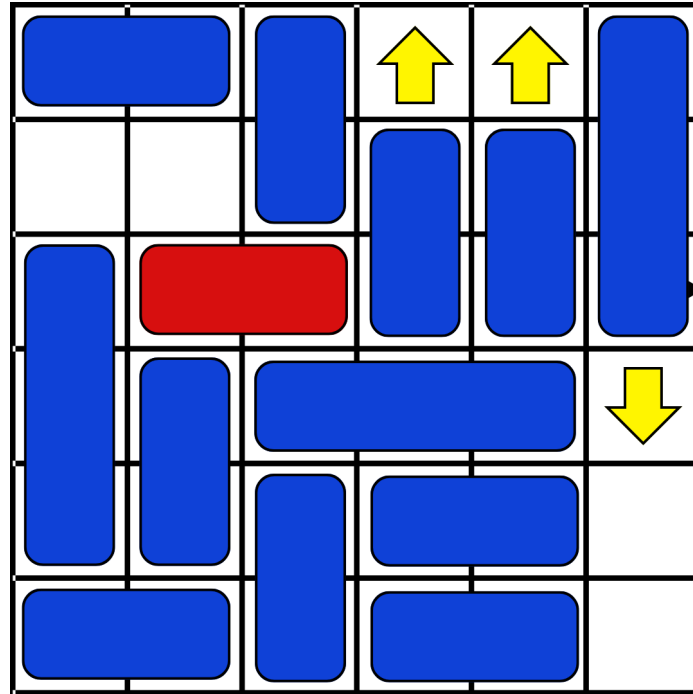


Gambar 2. Awal Permainan Game Rush Hour

Pemain dapat menggeser-geser *piece* (termasuk *primary piece*) untuk membentuk jalan lurus antara *primary piece* dan *pintu keluar*.

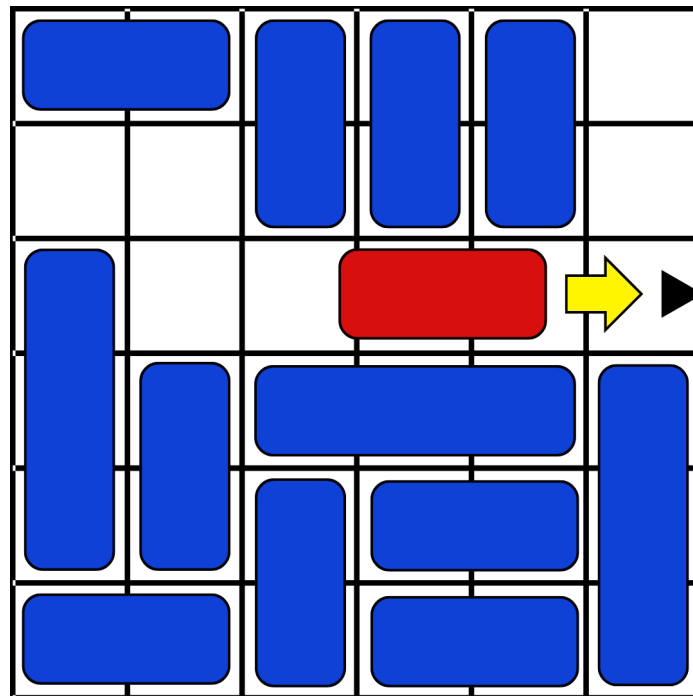


Gambar 3. Gerakan Pertama Game Rush Hour




Gambar 4. Gerakan Kedua Game Rush Hour

Puzzle berikut dinyatakan telah selesai apabila *primary piece* dapat digeser keluar papan melalui *pintu keluar*.



Gambar 5. Pemain Menyelesaikan Permainan

Agar lebih jelas, silahkan amati video cara bermain berikut:

 The New Rush Hour by ThinkFun!

Anda juga dapat melihat gif berikut untuk melihat contoh permainan [Rush Hour Solution](#).

Spesifikasi Tugas Kecil 3:

- Buatlah program sederhana dalam bahasa **C/C++/Java/Javascript** yang mengimplementasikan **algoritma *pathfinding Greedy Best First Search, UCS (Uniform Cost Search), dan A**** dalam menyelesaikan permainan Rush Hour.
- Tugas dapat dikerjakan **individu atau berkelompok** dengan anggota **maksimal 2 orang** (sangat disarankan). Boleh lintas kelas dan lintas kampus, tetapi **tidak boleh sama** dengan anggota kelompok pada **tugas kecil Strategi Algoritma sebelumnya**.
- Algoritma *pathfinding* minimal menggunakan **satu heuristic** (2 atau lebih jika mengerjakan *bonus*) yang ditentukan sendiri. Jika mengerjakan *bonus*, *heuristic* yang digunakan ditentukan berdasarkan input pengguna.
- Algoritma dijalankan secara terpisah. Algoritma yang digunakan ditentukan berdasarkan Input pengguna.
- **Alur Program:**
 1. **[INPUT] konfigurasi permainan/test case** dalam format ekstensi **.txt**. File *test case* tersebut berisi:
 1. **Dimensi Papan** terdiri atas dua buah variabel **A** dan **B** yang membentuk papan berdimensi $A \times B$
 2. **Banyak *piece* BUKAN *primary piece*** direpresentasikan oleh variabel integer **N**.
 3. **Konfigurasi papan** yang mencakup penempatan *piece* dan *primary piece*, serta lokasi *pintu keluar*. *Primary Piece* dilambangkan dengan huruf **P** dan *pintu keluar* dilambangkan dengan huruf **K**. *Piece* dilambangkan dengan huruf dan karakter selain **P** dan **K**, dan huruf/karakter berbeda melambangkan *piece* yang berbeda. *Cell* kosong dilambangkan dengan karakter **'.'** (titik). (**Catatan:** ingat bahwa *pintu keluar* pasti berada di *dinding* papan dan sejajar dengan orientasi *primary piece*)

File .txt yang akan dibaca memiliki format sebagai berikut:

```
A B
N
konfigurasi_papan
```

Contoh Input

```

6 6
11
AAB..F
..BCDF
GPPCDFK
GH.III
GHJ...
LLJMM.

```

keterangan: “K” adalah pintu keluar, “P” adalah primary piece, Titik (“.”) adalah cell kosong.

Contoh konfigurasi papan lain yang mungkin berdasarkan letak *pintu keluar* (X adalah *piece/cell random*)

K	XXX	XXX
XXX	KXXX	XXX
XXX	XXX	XXX
XXX		K

2. [INPUT] algoritma *pathfinding* yang digunakan
3. [INPUT] *heuristic* yang digunakan (**bonus**)
4. [OUTPUT] Banyaknya **gerakan** yang diperiksa (alias banyak ‘node’ yang dikunjungi)
5. [OUTPUT] Waktu eksekusi program
6. [OUTPUT] konfigurasi **papan** pada setiap tahap pergerakan/pergeseran. Output ini tidak harus diimplementasi apabila mengerjakan *bonus output GUI*. **Gunakan print berwarna** untuk menunjukkan pergerakan *piece* dengan jelas. Cukup mewarnakan **primary piece**, **pintu keluar**, dan **piece yang digerakkan** saja (boleh dengan *highlight* atau *text color*). Pastikan ketiga komponen tersebut memiliki warna berbeda.

Format sekuens adalah sebagai berikut:

```

Papan Awal
[konfigurasi_papan_awal]

Gerakan 1: [piece]-[arah gerak]
[konfigurasi_papan_gerakan_1]

Gerakan 2: [piece]-[arah gerak]

```

[konfigurasi_papan_gerakan_2]

Gerakan [N]: [piece]-[arah gerak]

[konfigurasi_papan_gerakan_N]

dst

Contoh Output

Papan Awal

AAB..F

..BCDF

GPPCDFK

GH.III

GHJ...

LLJMM.

Gerakan 1: I-kiri

AAB..F

..BCDF

GPPCDFK

GHIII.

GHJ...

LLJMM.

Gerakan 2: F-bawah

AAB..

..BCDF

GPPCDFK

GHIII

GHJ...

LLJMM.

dst

Keterangan: hanya sebagai contoh. Pastikan output jelas dan mudah dimengerti. Warna dan highlight hanya untuk menunjukkan perubahan.

7. [OUTPUT] animasi gerakan-gerakan untuk mencapai solusi (bonus GUI).

- Berkas laporan yang dikumpulkan adalah laporan dalam bentuk PDF yang setidaknya berisi:

1. Penjelasan algoritma *UCS*, *Greedy Best First Search*, dan *A** (dan **algoritma alternatif** apabila mengerjakan bonus), **bukan hanya berisi notasi pseudocode dan BUKAN IMPLEMENTASINYA melainkan algoritmanya.**
2. Analisis algoritma *UCS*, *Greedy Best First Search*, dan *A** (dan **algoritma alternatif** apabila mengerjakan bonus). **Analisis minimal memuat jawaban dari pertanyaan-pertanyaan berikut:**
 1. Definisi dari $f(n)$ dan $g(n)$, sesuai dengan salindia kuliah.
 2. Apakah heuristik yang digunakan pada algoritma *A** admissible? Jelaskan sesuai definisi admissible dari salindia kuliah.
 3. Pada penyelesaian Rush Hour, apakah algoritma *UCS* sama dengan *BFS*? (dalam artian urutan node yang dibangkitkan dan path yang dihasilkan sama)
 4. Secara teoritis, apakah algoritma *A** lebih efisien dibandingkan dengan algoritma *UCS* pada penyelesaian Rush Hour?
 5. Secara teoritis, apakah algoritma *Greedy Best First Search* menjamin solusi optimal untuk penyelesaian Rush Hour?
3. *Source* program dalam bahasa pemrograman yang dipilih (pastikan bahwa program telah dapat dijalankan).
4. Tangkapan layar yang memperlihatkan *input* dan *output* (minimal sebanyak **4** buah contoh untuk masing-masing algoritma). **Disarankan mencangkup semua kasus unik.**
5. Hasil analisis percobaan algoritma **pathfinding**. Analisis dilakukan dalam bentuk paragraf/poin dan minimal memuat mengenai analisis kompleksitas algoritma program yang telah dikembangkan.
6. Penjelasan mengenai implementasi bonus jika mengerjakan.
7. Pranala ke *repository* yang berisi kode program.

● **BONUS:**

Pastikan sudah mengerjakan spesifikasi wajib sebelum mengerjakan bonus:

1. Implementasikan **Algoritma Alternatif**
 Tambahkan minimal **1** (satu) implementasi algoritma *pathfinding* lain selain *Greedy Best First Search*, *UCS*, atau *A**. Algoritma *pathfinding* alternatif tidak boleh berupa *BFS* atau *DFS*.
2. Implementasi **Heuristic Alternatif**
 Implementasikan **2 (dua) atau lebih heuristic** (alias 1 atau lebih heuristic tambahan) yang dapat digunakan algoritma *pathfinding* dalam program kalian. Tambahkan **input** untuk memasukkan pilihan *heuristic* yang akan digunakan algoritma.

3. Graphical User Interface

Buatlah GUI untuk program yang Anda buat. Interface ini harus dapat menerima *input* secara *graphical*. Interface juga harus dapat mengeluarkan **output** berupa **animasi** gerakan-gerakan dari awal permainan sampai mencapai solusi. Kakas untuk implementasi GUI dibebaskan.

- Program disimpan dalam repository yang bernama Tucil3_NIM jika mengerjakan secara individu atau Tucil3_NIM1_NIM2 jika dikerjakan berkelompok. Berikut merupakan struktur dari isi repository tersebut:
 1. Folder **src** berisi *source code* program.
 2. Folder **bin** berisi *executable file* (Sesuaikan dengan bahasa pemrograman yang digunakan).
 3. Folder **test** berisi solusi jawaban dari data uji yang digunakan dalam laporan.
 4. Folder **doc** berisi laporan tugas kecil dalam bentuk PDF.
 5. **README** yang minimal berisi:
 - a. Penjelasan singkat program yang dibuat.
 - b. Requirement program dan instalasi tertentu bila ada.
 - c. Cara mengkompilasi program bila perlu dikompilasi (pastikan dengan langkah yang **jelas** dan **benar**).
 - d. Cara menjalankan dan menggunakan program (pastikan dengan langkah yang **jelas** dan **benar**).
 - e. Author / identitas pembuat.
- Laporan dikumpulkan hari **Rabu, 21 Mei 2025** pada alamat Google Form berikut paling lambat pukul **12.00 WIB**:
<https://forms.gle/bVRWcFhVssRjCzNx5>
- Pertanyaan terkait tugas kecil ini bisa disampaikan melalui QnA berikut:
<https://bit.ly/QnA-Stima-25>

Perhatikan:

- **Dilarang keras copy paste** program dari internet, AI, repository lain, ataupun teman. Program harus dibuat sendiri, kecurangan akan mengakibatkan nilai tugas menjadi **nol**.
- Pastikan program **dapat setidaknya dikompilasi** pada *windows* dan *linux*.
- Apabila program **tidak dapat dijalankan** maka tidak akan dinilai oleh asisten.
- Tugas dikerjakan oleh maksimal dua orang.
- Tambahkan tabel berikut yang diisi *checklist* (✓) pada bagian lampiran laporan Anda.

Poin	Ya	Tidak
1. Program berhasil dikompilasi tanpa kesalahan		

2. Program berhasil dijalankan		
3. Solusi yang diberikan program benar dan mematuhi aturan permainan		
4. Program dapat membaca masukan berkas .txt dan menyimpan solusi berupa print board tahap per tahap dalam berkas .txt		
5. [Bonus] Implementasi algoritma pathfinding alternatif		
6. [Bonus] Implementasi 2 atau lebih heuristik alternatif		
7. [Bonus] Program memiliki GUI		
8. Program dan laporan dibuat (kelompok) sendiri		

--- Selamat Mengerjakan! ---

“Jakarta ahh game”

--- Ariel ---

“Simulasi keluar macet bobotoh Persib”

--- Ciko ---