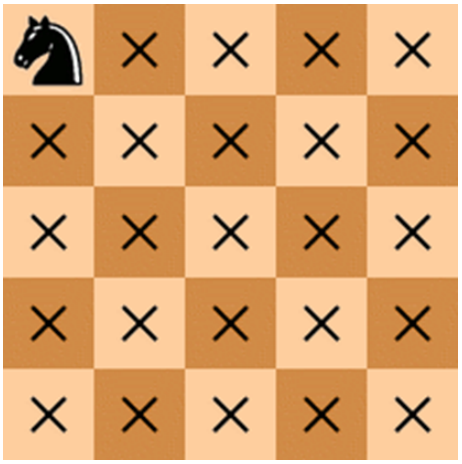
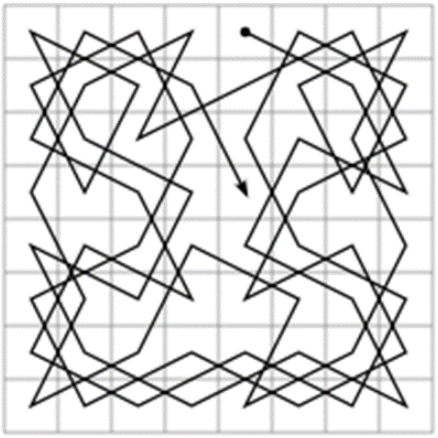
TUGAS PRAKTIKUM 2

Nama : Muhammad Daffa Rizky Sutrisno

NRP : 5025231207

Chess :





1. Jika sebuah bidak kuda diletakkan pada sebarang kotak untuk kemudian melakukan perjalanan (dengan cara pergerakan kuda) mengunjungi ke semua (8 x 8) kotak papan catur.

   Jika diinginkan situasi bahwa kuda tsb dapat:

   a. Mengakhiri perjalanan pada attacking square (closed tour);
   b. Mengakhiri perjalanan di sebarang kotak (open tour);

   Maka aplikasikan algoritma untuk menyelesaikan masalah di atas ke dalam sebuah program dengan menunjukkan rute perjalanan seperti gambar kanan bawah.

Solusi untuk Knight's Tour Problem menggunakan algoritma Warnsdorff

Solusi ini melibatkan aturan heuristik untuk menentukan langkah selanjutnya bagi sang kuda (knight) dalam catur sehingga memiliki kemungkinan terkecil untuk terjebak atau mencapai langkah buntu.

Berikut langkah-langkah implementasi algoritma Warnsdorff untuk Knight's Tour Problem:

1. Tentukan langkah awal, yaitu kotak awal kuda pada papan catur.

2. Hitung semua langkah yang mungkin dari posisi saat ini, yaitu sel-sel yang dapat dikunjungi kuda berdasarkan pola gerakannya.
3. Pilih langkah dengan jumlah kemungkinan langkah selanjutnya paling sedikit.
4. Pindahkan kuda ke posisi tersebut.
5. Perbarui daftar langkah yang mungkin, dengan menghapus sel yang telah dikunjungi.
6. Ulangi langkah 2-5 hingga seluruh papan catur (8x8 kotak) telah dikunjungi.

Jika algoritma berhasil, rute perjalanan akan ditampilkan dalam format matriks atau grafik. Jika tidak, solusi tidak ditemukan.

Metode Warnsdorff membantu mengarahkan kuda ke sekitar papan catur dengan efisien, meminimalkan peluang terjebak, dan meningkatkan kesempatan menemukan solusi untuk Knight's Tour Problem. Algoritma ini didasarkan pada ide bahwa memilih langkah yang mendekati batas catur dapat membantu mengurangi kemungkinan langkah buntu di kemudian hari.

A. Knight's Tour Closed

Berikut adalah implementasi codenya.
Program ini mencari *open knight's tour* pada papan catur 8x8. *Open tour* berarti tur kuda tidak perlu kembali ke posisi awal.

Cara Kerja

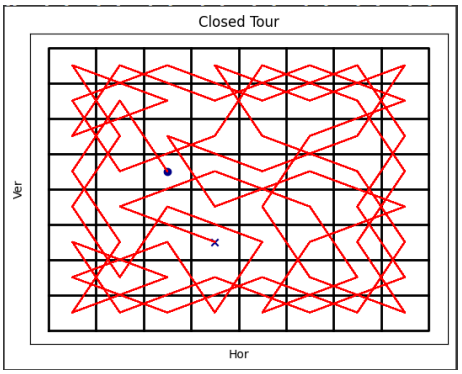
1. **Inisialisasi Papan Catur**:
    - Papan catur diwakili sebagai matriks 8x8. Nilai `-1` menandakan kotak belum dikunjungi.
2. **Validasi Langkah**:
    - Fungsi `is_valid_move` mengecek apakah langkah kuda berada dalam batas papan dan kotak belum dikunjungi.
3. **Pemilihan Langkah Berdasarkan Aksesibilitas**:
    - Fungsi `get_valid_moves` menghitung jumlah langkah valid dari setiap posisi calon langkah dan memilih langkah dengan aksesibilitas minimum (metode Warnsdorff).
4. **Algoritma Utama**:
    - Fungsi `knight_tour` menggunakan rekursi untuk mencari jalur kuda, dimulai dari posisi awal pengguna.
    - Jika tur selesai (64 langkah), algoritma berhenti.
    - Jika tur bersifat "open" (tidak ada jalur balik ke posisi awal), papan hasil dicetak.
5. **Visualisasi**:

- Langkah-langkah kuda disusun menjadi daftar koordinat untuk divisualisasikan menggunakan `matplotlib`.
- Papan catur digambar dengan batas, grid, dan jalur perjalanan kuda.

Hasil

Program menghasilkan rute perjalanan kuda yang memenuhi syarat **closed tour** di papan catur dan memvisualisasikan rute tersebut dengan diagram. Berikut adalah hasil ketika program dijalankan :

```
Enter the starting X coordinate (0-7): 3
Enter the starting Y coordinate (0-7): 2
34      31      16      49      20      29      14      51
17      2       33      30      15      50      21      28
32      35      54      19      48      27      52      13
3       18      1       62      53      56      47      22
36      63      38      55      26      61      12      57
7       4       41      64      39      58      23      46
42      37      6       9       44      25      60      11
5       8       43      40      59      10      45      24
[[3, 2], [1, 1], [3, 0], [5, 1], [7, 0], [6, 2], [5, 0], [7, 1], [6, 3], [7, 5], [6, 7], [4, 6], [2, 7], [0, 6], [1, 4], [0, 2], [1, 0], [3, 1], [2, 3], [0, 4], [1, 6], [3, 7], [5, 6], [7, 7],
[[3, 2], [1, 1], [3, 0], [5, 1], [7, 0], [6, 2], [5, 0], [7, 1], [6, 3], [7, 5], [6, 7], [4, 6], [2, 7], [0, 6], [1, 4], [0, 2], [1, 0], [3, 1], [2, 3], [0, 4], [1, 6], [3, 7], [5, 6], [7, 7],
```



B. Knight's Tour Open

Berikut adalah implementasi codenya.
Program ini mencari *closed knight's tour*, yaitu jalur kuda yang kembali ke posisi awal.

Cara Kerja

1. **Inisialisasi Papan Catur**:
   - Sama seperti Program 1, menggunakan array linear ukuran 64 untuk merepresentasikan papan.
   - Nilai `-1` menandakan kotak belum dikunjungi.
2. **Validasi Langkah**:
   - Fungsi `limits` mengecek batas papan.
   - Fungsi `isempty` mengecek apakah kotak belum dikunjungi.
3. **Pemilihan Langkah dengan Algoritma Warnsdorff**:

- Fungsi `nextMove` menentukan langkah berikutnya berdasarkan aksesibilitas minimum dari posisi calon langkah.

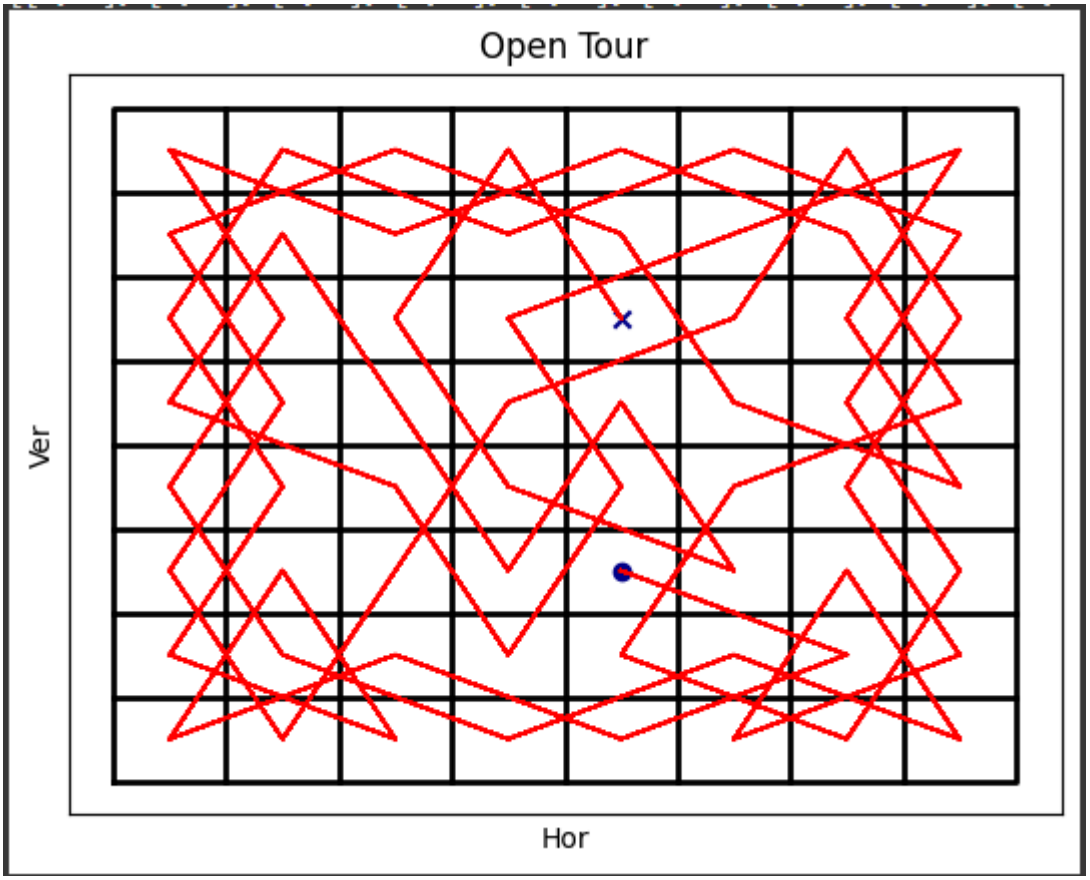
4. **Cek Kondisi Closed Tour**:
   - Setelah semua langkah selesai, fungsi `neighbour` mengecek apakah langkah terakhir dapat kembali ke posisi awal.
   - Jika iya, tur selesai dan matriks langkah dicetak.
5. **Visualisasi**:
   - Sama seperti Program 1, langkah-langkah kuda divisualisasikan menggunakan `matplotlib`.

Hasil

Program menghasilkan rute perjalanan kuda yang memenuhi syarat **open tour** di papan catur dan memvisualisasikan rute tersebut dengan diagram. Berikut adalah hasil ketika program dijalankan :

```
Masukkan koordinat x starting point (0-7): 5
Masukkan koordinat y starting point (0-7): 4

Knight's Tour Open:
[23, 10, 43, 62, 21, 12, 35, 48]
[42, 55, 22, 11, 44, 49, 20, 13]
[9, 24, 61, 50, 63, 36, 47, 34]
[54, 41, 56, 37, 58, 45, 14, 19]
[25, 8, 53, 60, 51, 18, 33, 46]
[40, 5, 38, 57, 0, 59, 30, 15]
[7, 26, 3, 52, 17, 28, 1, 32]
[4, 39, 6, 27, 2, 31, 16, 29]
[23, 10, 43, 62, 21, 12, 35, 48, 42, 55, 22, 11, 44, 49, 20, 13, 9, 24, 61, 50, 63, 36, 47, 34, 54, 41, 56, 37, 58, 45, 14, 19, 25, 8, 53, 60, 51, 18, 33, 46, 40, 5, 38, 57, 0, 59, 30, 15, 7,
[[5, 4], [6, 6], [7, 4], [6, 2], [7, 0], [5, 1], [7, 2], [6, 0], [4, 1], [2, 0], [0, 1], [1, 3], [0, 5], [1, 7], [3, 6], [5, 7], [7, 6], [6, 4], [4, 5], [3, 7], [1, 6], [0, 4], [1, 2], [0, 0],
[[5, 4], [6, 6], [7, 4], [6, 2], [7, 0], [5, 1], [7, 2], [6, 0], [4, 1], [2, 0], [0, 1], [1, 3], [0, 5], [1, 7], [3, 6], [5, 7], [7, 6], [6, 4], [4, 5], [3, 7], [1, 6], [0, 4], [1, 2], [0, 0],
```



C. Perbedaan Kedau Algoritma

Berikut adalah perbedaan utama antara kedua algoritma yang digunakan tersebut:

1. Jenis Tur

- **Program 1**: Mencari *open knight's tour* (tur kuda terbuka), di mana kuda tidak perlu kembali ke posisi awal setelah langkah terakhir.
- **Program 2**: Mencari *closed knight's tour* (tur kuda tertutup), di mana langkah terakhir kuda harus dapat kembali ke posisi awal.

2. Representasi Papan

- **Program 1**:
  - Menggunakan **matriks 2D** ukuran 8x8 (`board[x][y]`) untuk mewakili papan catur.
  - Nilai `-1` menunjukkan kotak belum dikunjungi.
- **Program 2**:
  - Menggunakan **array 1D** ukuran 64 (`a[y * N + x]`) untuk merepresentasikan papan catur.
  - Nilai `-1` juga digunakan untuk kotak yang belum dikunjungi.

3. Algoritma Pemilihan Langkah

- **Program 1**:
  - Menggunakan metode berbasis **aksesibilitas minimum**.
  - Fungsi `get_valid_moves` menghitung jumlah langkah valid dari setiap calon langkah dan memilih langkah dengan jumlah langkah valid paling sedikit (*Warnsdorff heuristic*).
  - Ada elemen *randomization* untuk mengacak pemilihan langkah dengan aksesibilitas sama.
- **Program 2**:
  - Langsung mengimplementasikan **algoritma Warnsdorff** dengan memulai dari langkah acak.
  - Tidak menghitung ulang langkah valid setelah langkah dipilih.

4. Pengecekan Keberhasilan

- **Program 1**:
  - Menggunakan fungsi `is_closed_tour` untuk memastikan tur bersifat *open*.
  - Memeriksa apakah langkah terakhir tidak memiliki jalur balik ke posisi awal.
- **Program 2**:
  - Menggunakan fungsi `neighbour` untuk memastikan tur bersifat *closed*.
  - Memeriksa apakah langkah terakhir dapat langsung kembali ke posisi awal.

5. Pencarian Solusi

- **Program 1**:
    - Mencari solusi satu kali untuk menghasilkan tur *open*.
    - Jika tur yang ditemukan bersifat tertutup, program akan mencoba lagi.
- **Program 2**:
    - Terus mencari solusi hingga menemukan *closed tour*.
    - Akan mengulang algoritma dari awal jika tidak ada solusi yang ditemukan.

6. Visualisasi Hasil

- **Program 1** dan **Program 2**:
    - Sama-sama menggunakan `matplotlib` untuk menggambar papan catur dan jalur perjalanan kuda.
    - Bedanya, **Program 1** menandai jalur untuk *open tour*, sedangkan **Program 2** menandai jalur untuk *closed tour*.

7. Fokus pada Kondisi Akhir

- **Program 1**: Tidak memerlukan syarat tambahan di langkah terakhir selain menyelesaikan 64 langkah.
- **Program 2**: Langkah terakhir harus memastikan jalur balik ke posisi awal, sehingga memenuhi syarat *closed tour*.

Ringkasan Perbedaan Utama

- **Tujuan Akhir**: *Open tour* vs *Closed tour*.
- **Representasi Data**: Matriks 2D vs Array 1D.
- **Pemilihan Langkah**: Aksesibilitas minimum dengan acak vs Algoritma Warnsdorff langsung.
- **Pengecekan Akhir**: Tidak ada jalur balik vs Harus ada jalur balik ke awal.