

BÁO CÁO MÔN HỌC TRÍ TUỆ NHÂN TẠO

ĐỀ TÀI
Giải quyết bài toán N-Puzzle
bằng thuật toán A* và IDA*

Nhóm sinh viên thực hiện:

HỌ VÀ TÊN	MSSV	LỚP	KHÓA
NGUYỄN NGỌC ĐÔN	20130941	CNTT2.03	K58
NGUYỄN TẮT HÒA	20131536	CNTT2.01	K58
BÙI VĂN TOẢN	20134033	CNTT2.01	K58

Giảng viên hướng dẫn: TS. Nguyễn Nhật Quang

HÀ NỘI, THÁNG 11/2015

Mục lục

Phần I: Giới thiệu bài toán.....	2
1. Mô tả bài toán thực tế.....	2
2. Ý tưởng và phương pháp	3
a. Xây dựng mô hình dữ liệu.....	3
b. Phương pháp.....	3
Phần 2: Các chức năng chính của hệ thống.....	4
Phần 3: Các vấn đề gặp phải trong quá trình làm đồ án	5
1. Tính giải được của 1 trạng thái bất kỳ	5
a. Inversion (Sự đảo ngược)	5
b. Tính chất giải được.	6
c. Chứng minh.....	6
2. Tìm kiếm hàm Heuristic tối ưu.....	8
• Manhattan Distance	8
• Linear Conflict	9
• Tiles out of row and column.....	9
• N-MaxSwap (Gaschnig's Heuristic)	10
• Pythagorean	11
Phần 4: Các đánh giá sau khi làm đồ án	12
Phần 5: Tài liệu tham khảo	16

Phần I:

Giới thiệu bài toán

1. Mô tả bài toán thực tế

Trò chơi ghép tranh (N-Puzzle) với các phiên bản thực tế thường gặp 8-Puzzle, 15-Puzzle là vấn đề cổ điển cho mô hình thuật toán liên quan đến trí tuệ nhân tạo. Trò chơi bắt đầu với 1 lưới vuông kích thước $m * m$, với $m * m = N + 1$. Trên lưới có N miếng ghép được đánh số từ 1 đến N và 1 ô vuông trống. Yêu cầu đặt ra là phải di chuyển N miếng ghép đó tới khi đúng vị trí.

Ví dụ:

8		6
5	4	7
2	3	1

Trạng thái đầu

	1	2
3	4	5
6	7	8

Trạng thái đích

2. Ý tưởng và phương pháp

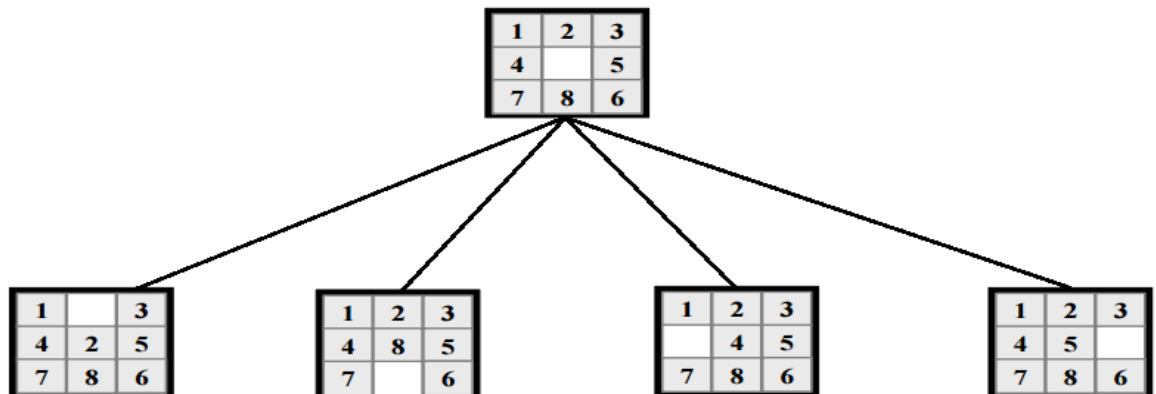
a. Xây dựng mô hình dữ liệu.

Thuật toán được sử dụng trong hệ thống là các thuật toán Informed Search, cụ thể là A* và IDA*. Do đó, tại mỗi trạng thái, ngoài việc lưu lại vị trí các ô, ta còn phải lưu lại chi phí đi từ trạng thái đầu đến trạng thái hiện tại và chi phí ước tính đến trạng thái đích. Ngoài ra, để lấy được các hành động để chuyển từ trạng thái đầu đến trạng thái đích, ta cần lưu lại trạng thái liền trước của trạng thái đang xét và hành động đã được thực hiện.

Cấu trúc dữ liệu:

```
struct Node{
    char **cell;    // Mảng 2 chiều lưu vị trí các ô
    int f;          // Chi phí ước tính để tới trạng thái đích
    int g;          // Chi phí đi từ trạng thái đầu đến trạng thái hiện tại
    Node *parent;   // Trạng thái liền trước
    int action;     // Hành động đã thực hiện
};
```

b. Phương pháp



Tại mọi thời điểm, từ 1 trạng thái của lưới, có tối đa 4 hành động để đưa lưới sang 4 trạng thái tiếp theo. Sử dụng 1 số hàm ước lượng chi phí kết hợp với thuật toán A*, IDA*, ta sẽ tìm được lời giải.

Phần 2:

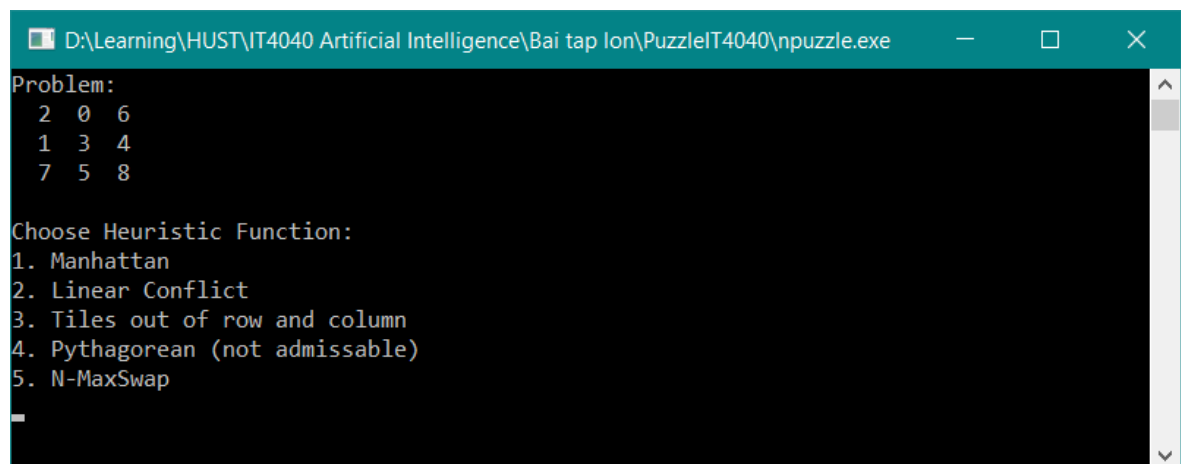
Các chức năng chính của hệ thống

Chương trình đọc dữ liệu từ file npuzzle.txt sau đó cho phép chọn hàm Heuristic sẽ được dùng làm tri thức để giải quyết bài toán.

Cấu trúc file npuzzle.txt:

Dòng 1 gồm 1 số nguyên N cho biết kích thước của lưới vuông là $N \times N$. N dòng tiếp theo mỗi dòng có N số nguyên nằm trong $[0, N]$, biểu diễn trạng thái ban đầu của lưới vuông, với 0 tương ứng với ô trống.

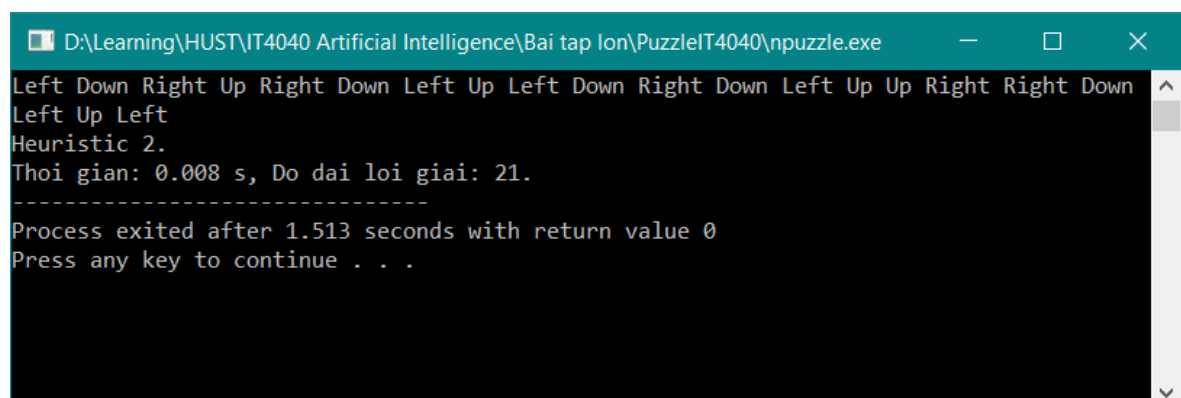
Sau khi đọc file, chương trình cho phép chọn hàm Heuristic sau đó bắt đầu tìm lời giải.



```

D:\Learning\HUST\IT4040 Artificial Intelligence\Bai tap lon\PuzzleIT4040\npuzzle.exe
Problem:
2 0 6
1 3 4
7 5 8

Choose Heuristic Function:
1. Manhattan
2. Linear Conflict
3. Tiles out of row and column
4. Pythagorean (not admissable)
5. N-MaxSwap
  
```



```

D:\Learning\HUST\IT4040 Artificial Intelligence\Bai tap lon\PuzzleIT4040\npuzzle.exe
Left Down Right Up Right Down Left Up Left Down Right Down Left Up Up Right Right Down
Left Up Left
Heuristic 2.
Thời gian: 0.008 s, Độ dài lời giải: 21.
-----
Process exited after 1.513 seconds with return value 0
Press any key to continue . . .
  
```

Phần 3:

Các vấn đề gặp phải trong quá trình làm đồ án

Trong quá trình làm đồ án, nhóm em đã gặp phải một số khó khăn sau:

1. Tính giải được của 1 trạng thái bất kỳ

a. Inversion (Sự đảo ngược)

Nếu trải dàn lưới vuông $N \times N$ thành 1 hàng ngang, ta có 1 inversion khi 1 ô nằm trước 1 ô khác nhưng giá trị ô nằm trước lớn hơn giá trị ô nằm sau (không tính ô trống).

Ví dụ:

2		6
1	3	4
7	5	8

2		6	1	3	4	7	5	8
---	--	---	---	---	---	---	---	---

Hình bên phải là trạng thái lưới vuông ở bên phải nhưng viết trên cùng 1 hàng ngang.

Số inversions của ô 2 là 1. (1)

Số inversions của ô 6 là 4. (1, 3, 4, 5)

...

Tổng số Inversion của trạng thái này là: 6.

b. Tính chất giải được.

	1	2
3	4	5
6	7	8

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Với trạng thái đích là khi lưới ở vị trí như hình trên (ô trống ở góc trái trên cùng, giá trị các ô tăng từ trái qua phải, từ trên xuống dưới), ta có 2 định lý sau:

Theorem 1: Mọi trạng thái giải được (Solvable State) của lưới $N \times N$ đều có tính chất P như sau:

- + Nếu N lẻ, số inversions chẵn.
- + Nếu N chẵn, số inversions của lưới và số thứ tự hàng của ô trống (tính từ trên xuống, bắt đầu từ hàng 0) có cùng tính chẵn lẻ.

Theorem 2: Mọi trạng thái có tính chất P đều giải được.

c. Chứng minh

• Theorem 1:

+ Fact 0: Trạng thái đích của lưới vuông $N \times N$ có tính chất P.

+ Fact 1: Với lưới vuông kích thước $N \times N$, nếu N lẻ, mọi bước di chuyển hợp lệ sẽ giữ nguyên tính chẵn lẻ của số inversions.

- Với các cách di chuyển ô trống sang trái hoặc sang phải, số inversions được giữ nguyên.
- Với các cách di chuyển ô trống lên trên hoặc xuống dưới:

2	3	6			2	3	6
1	5	4	\Rightarrow		1		4
7		8			7	5	8

2	3	6	1	5	4	7		8
---	---	---	---	---	---	---	--	---

2	3	6	1		4	7	5	8
---	---	---	---	--	---	---	---	---

Khi ô trống di chuyển xuống, nó đi qua 1 số chẵn các ô khác (N-1). Trong hình trên, nó đi qua 2 ô 4 và 7. Ô trống đổi chỗ cho ô 5 ngay dưới nó. Do đó có $(N - 1) + 1 = N$ ô bị thay đổi số inversion.

Với các ô mà nó đi qua, lượng thay đổi của giá trị inversion của mỗi ô sẽ là tăng hoặc giảm 1 đơn vị tùy vào giá trị ô đó. Do đó lượng thay đổi inversion của N-1 ô này là:

$\Delta I = \pm 1 \pm 1 \pm 1 \dots \pm 1$, giá trị này là 1 số chẵn do N-1 chẵn.

Với ô mà nó trực tiếp đổi chỗ, lượng thay đổi giá trị inversion của ô này cũng bằng giá trị ΔI trên. Do đó tính chẵn lẻ của số inversion của trạng thái được bảo toàn.

+ Fact 2: Với lưới vuông kích thước NxN, nếu N chẵn, mọi bước di chuyển hợp lệ sẽ đảm bảo số inversions của lưới và số thứ tự hàng của ô trống luôn cùng tính chẵn lẻ.

Chứng minh tương tự Fact 1 ta có:

- Với các cách di chuyển ô trống sang trái hoặc sang phải, số inversions và số thứ tự hàng của ô trống cũng được giữ nguyên.

- Với các cách di chuyển ô trống lên trên hoặc xuống dưới:

ΔI của các ô mà ô trống đi qua có tổng giá trị là 1 số lẻ do N-1 lẻ.

ΔI của ô mà ô trống trực tiếp đổi chỗ cũng là 1 số lẻ.

Do đó, số inversion của lưới sẽ thay đổi tính chẵn lẻ. Mặt khác, ô trống sẽ tiến lên trên hoặc xuống dưới đúng 1 hàng nên số thứ tự hàng của nó cũng thay đổi tính chẵn lẻ. Vậy, số inversions của lưới và số thứ tự hàng của ô trống được đảm bảo luôn cùng tính chẵn lẻ.

+ Fact 3: Mọi trạng thái giải được của lưới đều có thể đạt được bằng cách thực hiện 1 số lượng hữu hạn các bước di chuyển hợp lệ từ trạng thái đích.

Từ Fact 0, Fact 1, Fact 2 và Fact 3 ta suy ra Theorem 1 là đúng.

- **Theorem 2:**

Để chứng minh mọi lưới vuông 3×3 có tính chất P đều có thể giải được, nhóm em sinh tất cả các trường hợp đó, sau đó tìm lời giải cho từng trường hợp. Với lưới 4×4 hoặc lớn hơn, do số lượng trường hợp quá lớn và với 1 số lượng các trường hợp được sinh ra, hệ thống đều đưa ra được lời giải nên nhóm em chứng minh định lý này đối với lưới kích thước lớn.

Chương trình dùng để chứng minh định lý này nằm trong file `theorem2.cpp`. Kết quả được lưu trong 2 file:

- + `theorem2.txt` lưu tất cả các trạng thái của lưới 3×3 thỏa mãn tính chất P.
- + `theorem2_result.txt` lưu lời giải tương ứng cho các trạng thái đó.

2. Tìm kiếm hàm Heuristic tối ưu.

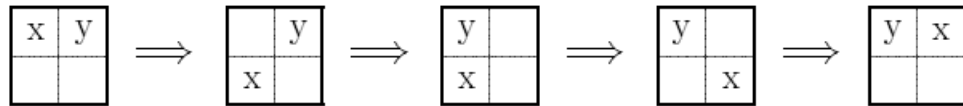
Khó khăn lớn nhất nhóm em gặp phải trong quá trình làm đồ án là tìm kiếm hàm Heuristic tốt và chứng minh hàm đó là hàm chấp nhận được. Sau khi tìm kiếm từ nhiều nguồn, nhóm em đã tích hợp 5 hàm Heuristic dưới đây vào chương trình.

- **Manhattan Distance**

Giả sử mỗi ô có thể di chuyển tự do trên lưới vuông. Khi đó, khoảng cách Manhattan được định nghĩa bằng tổng các bước di chuyển trực tiếp (theo hàng ngang, cột dọc) từ vị trí hiện tại của mỗi ô tới vị trí của ô đó trong trạng thái đích. Do thực tế, các ô không thể tự do di chuyển mà phải phụ thuộc vào ô trống nên khoảng cách Manhattan sẽ không lớn hơn chi phí thực tế. Do đó ước lượng Manhattan Distance là ước lượng chấp nhận được.

• Linear Conflict

Hai ô x và y gọi là Linear Conflict nếu x và y cùng nằm trên 1 đường thẳng (cùng hàng hoặc cùng cột) cả trong trạng thái hiện tại và trạng thái đích. Ở trạng thái hiện tại, ô x nằm bên trái (trên) ô y, nhưng ở trạng thái đích, ô x phải nằm bên phải (dưới) ô y.

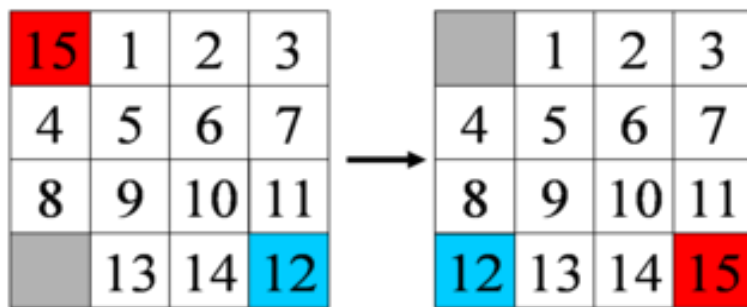


Ví dụ x và y là 2 ô Linear Conflict.

Tổng khoảng cách Manhattan của 2 ô x, y là 2. Nhưng trong thực tế, không thể đưa x và y về vị trí đúng chỉ với 2 bước di chuyển. Thay vào đó, cần có ít nhất 4 bước như trong hình để chuyển đổi vị trí 2 ô đó. Do đó, hàm Heuristic sẽ cộng thêm 2 chi phí trên mỗi cặp Linear Conflict và khoảng cách Mahattan. Trên thực tế cũng không có sẵn 2 ô trống ngay cạnh x và y để có thể tự do di chuyển 4 bước như trong hình nên hàm Heuristic ước lượng ra chi phí luôn nhỏ hơn chi phí thực. Như vậy hàm Heuristic là chấp nhận được.

• Tiles out of row and column

Hàm Heuristic được định nghĩa bằng tổng số ô nằm ngoài hàng của nó trong trạng thái đích và tổng số ô nằm ngoài cột của nó trong trạng thái đích.



Ví dụ như trong hình, ô 12 nằm đúng hàng như sai cột, ô 15 nằm ngoài cả hàng và cột, các ô khác (trừ ô trống) nằm đúng vị trí. Hàm Heuristic có giá trị bằng $1 + 2 = 3$. Rõ ràng chi phí này thấp hơn chi phí thực tế rất nhiều nên hàm Heuristic là chấp nhận được.

• N-MaxSwap (Gaschnig's Heuristic)

Hàm Heuristic được định nghĩa bằng số bước phải thực hiện để đưa lưới vuông về trạng thái đích nếu có thể đổi chỗ 1 ô bất kỳ với ô trống. Tương tự, hàm Heuristic này cũng là hàm chấp nhận được.

Hàm Heuristic này có thể được cài đặt bằng cách sử dụng 2 mảng:

$P[N * N]$ – biểu diễn trạng thái hiện tại của lưới vuông.

$B[N * N]$ – vị trí của ô i trong P . $B[i] = j$ nghĩa là $P[j] = i$

Tính chất:

- $B[P[i]] = i$.
- Khi đổi chỗ $P[i]$ và $P[j]$, ta phải đổi chỗ $B[P[i]]$ và $B[P[j]]$ để giữ nguyên ý nghĩa của mảng B .

Thuật toán:

- B1. Khởi tạo biến $count = 0$
- B2. Nếu lưới đã ở trạng thái đích thì gán $Heuristic = count$ và kết thúc.
- B3. $B[0]$ là vị trí hiện tại của ô trống.
- B4. Nếu $P[B[0]] \neq 0$, nghĩa là đang có 1 ô khác ô trống ở vị trí đó, đổi chỗ ô trống và ô đó ($swap(P[B[0]], P[B[B[0]]])$). Cập nhật mảng B . ($swap(B[0], B[B[0]])$).
- B5. Nếu $P[B[0]] = 0$, nghĩa là ô trống đang ở đúng vị trí, đổi chỗ ô trống và 1 ô bất kỳ đang nằm sai vị trí ($swap(P[i], P[B[0]])$). Cập nhật mảng B . ($swap(B[P[i]], B[0])$).
- B6. Tăng biến $count$ và lặp lại bước 2.

- **Pythagorean (không chấp nhận được)**

12	1	10	2
7		9	14
11	5	4	6
13	3	15	8

Trạng thái đầu

	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15

Trạng thái đích

Với mỗi ô i , ta có $d_i = |row_s - row_g|^2 + |column_s - column_g|^2$

Trong đó:

row_s và $column_s$ ứng với chỉ số hàng và cột của ô i trong trạng thái đầu;

row_g và $column_g$ ứng với chỉ số hàng và cột của ô i trong trạng thái đích.

Hàm Heuristic được định nghĩa có giá trị bằng $\sum d_i$.

Trong trường hợp trên hàm ước lượng có giá trị bằng 77 nhưng chi phí thực tế chỉ bằng 68. Do đó hàm Heuristic là không chấp nhận được. Tuy nhiên trong thực tế, khi dùng hàm ước lượng này, chương trình luôn tìm được lời giải rất nhanh và cũng tối ưu nên nhóm em cho vào để so sánh hiệu năng với các hàm khác.

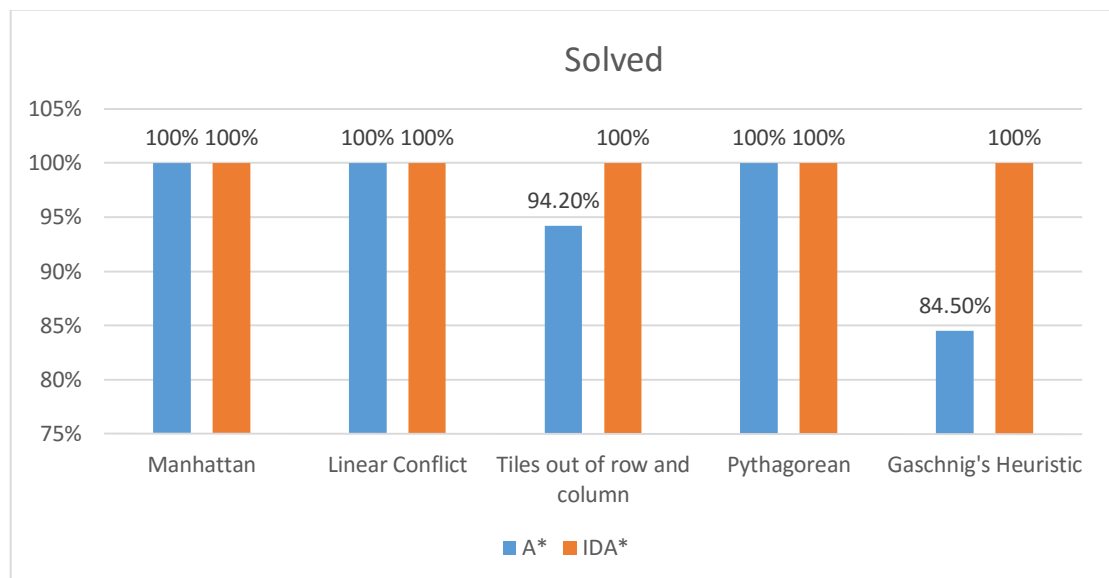
Cụ thể, chương trình tích hợp hàm ước lượng này giải được tất cả mọi trường hợp có thể giải của 8-Puzzle, ($9! / 2 = 181440$ trường hợp) trong thời gian trung bình 0.002s với thuật toán IDA*, giải 1000 trường hợp ngẫu nhiên của 16-Puzzle trong thời gian trung bình 0.3787s

Phần 4:

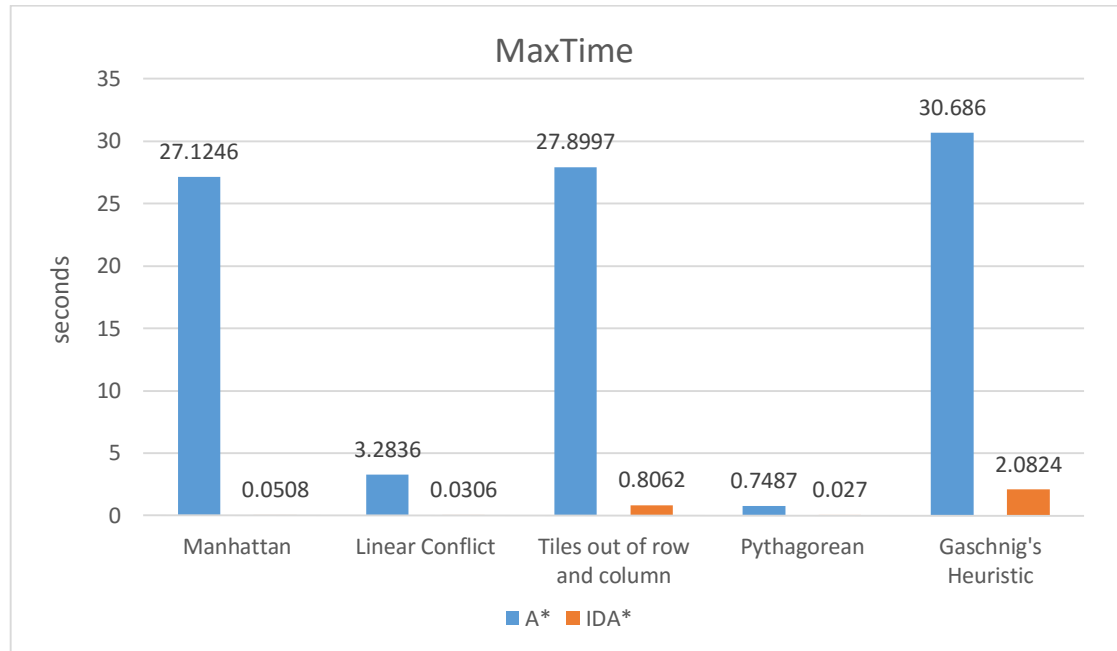
Các đánh giá sau khi làm đồ án

Đánh giá hiệu năng của 2 thuật toán tìm kiếm A* và IDA* với các hàm Heuristic.

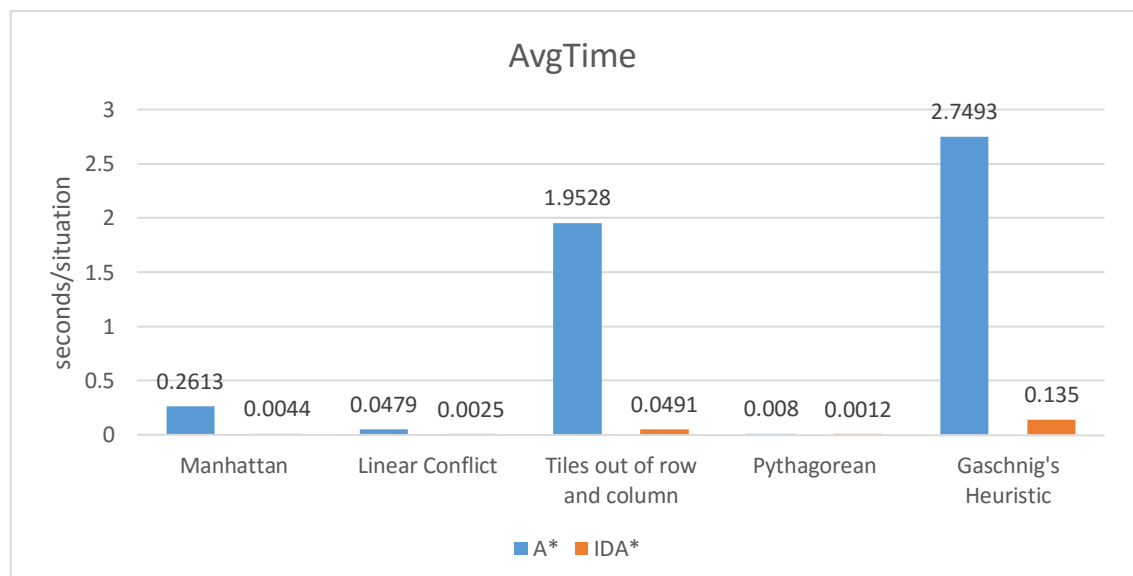
Số trường hợp tìm được lời giải / 1000 mẫu thử:



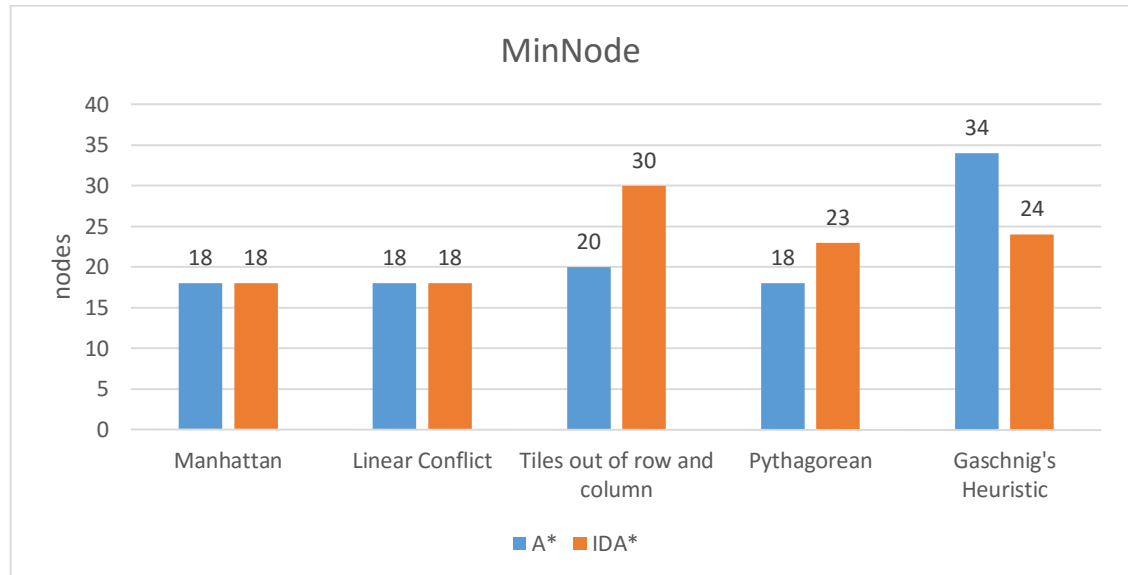
Thời gian lớn nhất để tìm ra lời giải:



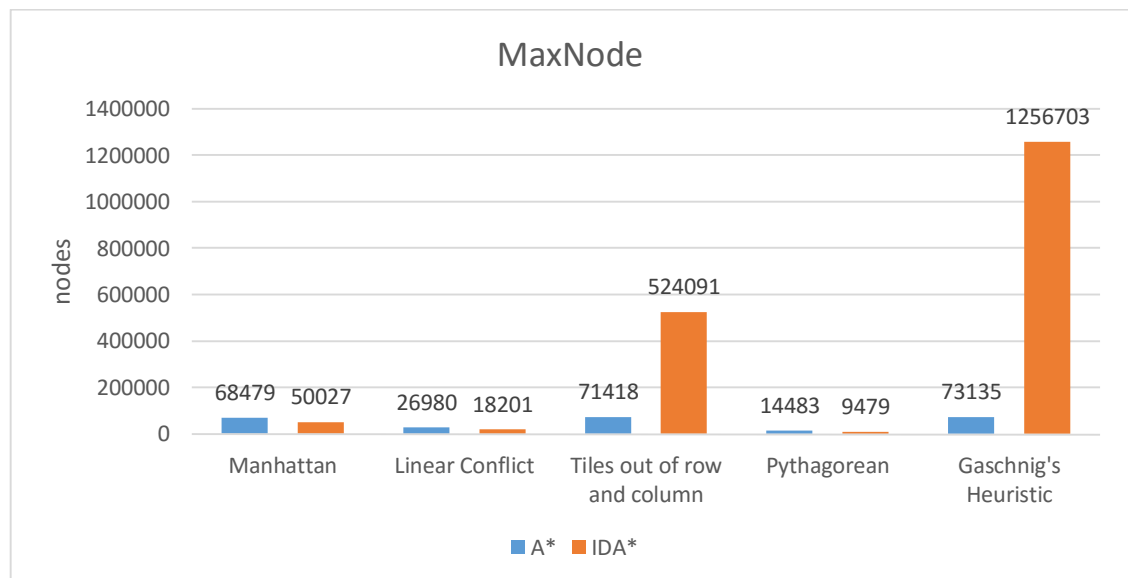
Thời gian trung bình để tìm ra lời giải:



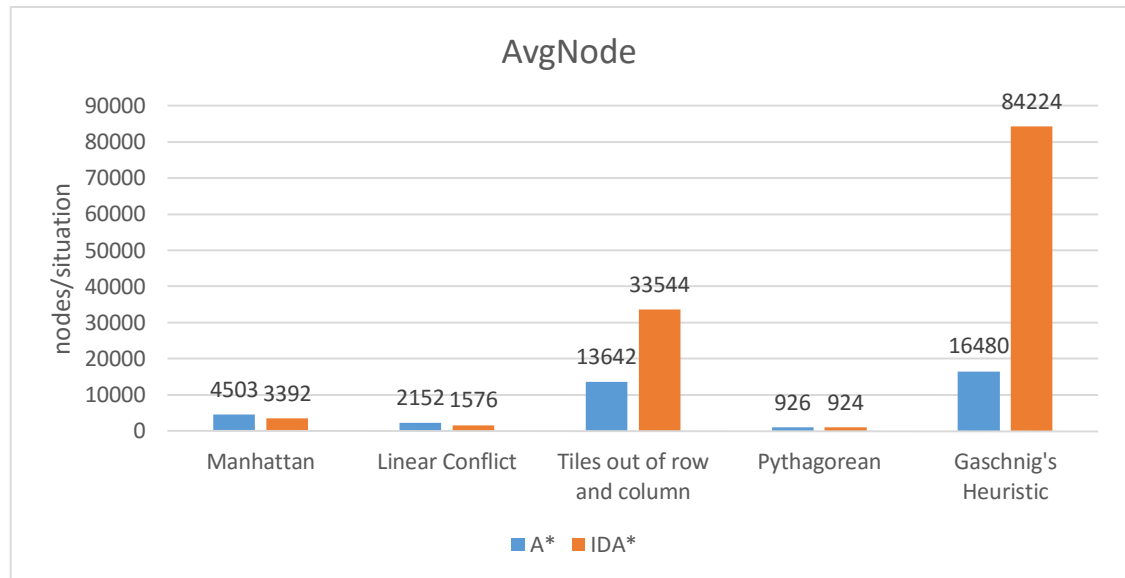
Số lượng Node ít nhất được sinh ra:



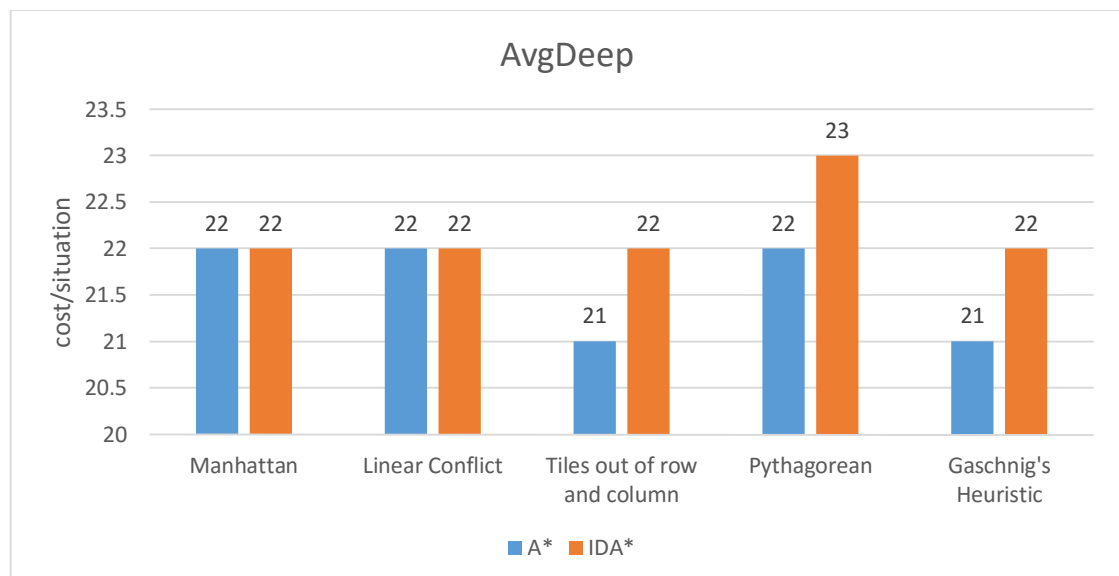
Số lượng Node lớn nhất được sinh ra:



Số lượng Node trung bình được sinh ra:



Độ sâu trung bình của lời giải:



Phần 5:

Tài liệu tham khảo

Trong quá trình làm bài tập, nhóm em có tham khảo một số tài liệu sau:

- Slide bài giảng môn học “Trí tuệ nhân tạo” của Tiến sỹ Nguyễn Nhật Quang, giảng viên trường Đại học Bách Khoa Hà Nội.
- Thư viện Timer (dùng để đo thời gian tìm kiếm lời giải) của Tiến sỹ Phạm Quang Dũng, giảng viên trường Đại học Bách Khoa Hà Nội.
- Tài liệu về các hàm Heuristic trên trang web:
<http://heuristicswiki.wikispaces.com/N++Puzzle>

Tài liệu về bài toán N-Puzzle từ nhiều nguồn khác:

<http://www.cs.unb.ca/profs/hzhang/CS4725/assignments/assgn2-06W-sol.htm>

<http://ethesis.nitrkl.ac.in/5575/1/110CS0081-1.pdf>

http://disi.unitn.it/~montreso/asd/progetti/2007-08/progetto2/The_Manchattan_Pair_Distance_Heuristic_for_the_15-puzzle.pdf

http://cseweb.ucsd.edu/~ccalabro/essays/15_puzzle.pdf
https://www.scribd.com/fullscreen/103698298?access_key=key-anvyh6p0mh6yxt4pq40

<http://ai.stanford.edu/~latombe/cs121/2011/slides/D-heuristic-search.pdf>

<https://www.cs.bham.ac.uk/~mdr/teaching/modules04/java2/TilesSolvability.html>

http://www.academia.edu/4167099/Tham_khao_Bao_cao_bai_tap_lon_du_an_cac_nhom_and_thao_luan_Tong_ket_noi_dung_AI_da_hoc