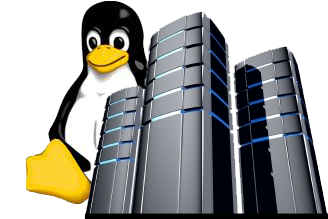
**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**



**VIỆN CÔNG NGHỆ THÔNG TIN – TRUYỀN THÔNG**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

****

**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 TẤT 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**

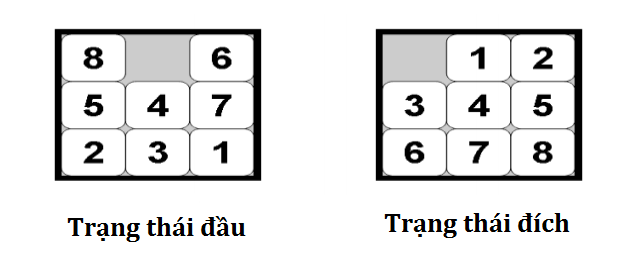
Mục lục

**Phần I:  
Giới thiệu bài toán**

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:



1. **Ý tưởng và phương pháp**
2. Xây dựng mô hình dữ liệu để mô tả 1 trạng thái của lưới vuông và vị trí các miếng ghép.

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

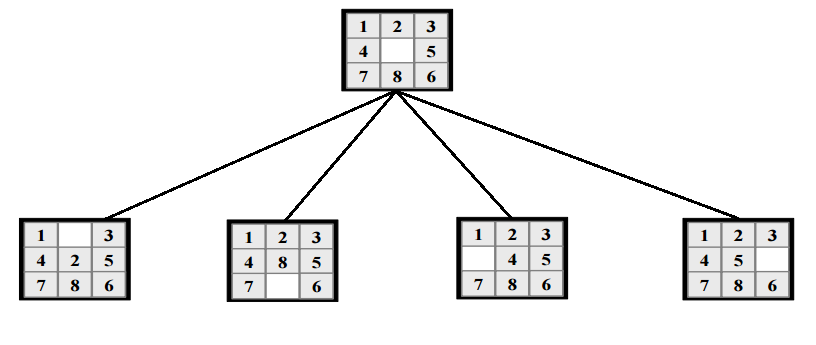
int f; // Chi phí ước tính để tới trạng thái đích

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

};

1. 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.

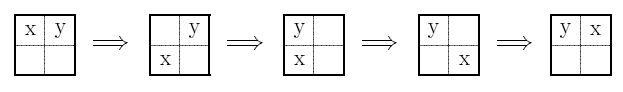
1. Một số hàm Heuristic được sử dụng

* 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 các Manhattan sẽ không lớn hơn chi phí thực tế. Do đó ược 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.

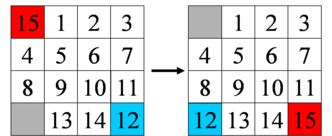


Ví dụ x và y là 2 ô Linear Conflict.

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

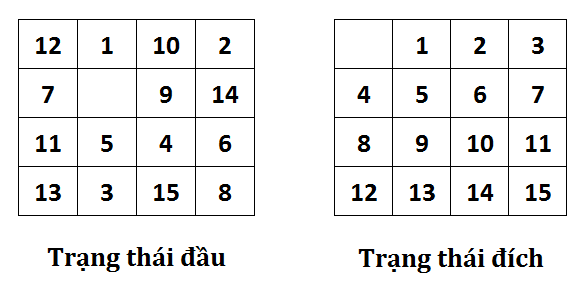
Tính chất:

- 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]] 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)



Với mỗi ô i, ta có =

Trong đó:  
 và ứng với chỉ số hàng và cột của ô i trong trạng thái đầu;  
 và ứng với chỉ số hàng và cột của ô i trong trạng thái đích.

Hàm Heuristic được định nghĩa có giá trị bằng .

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.

**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 x N.  
N dòng tiếp theo mỗi dòng có N số nguyên nằm trong , biểu diễn trạng thái ban đầu của lưới vuông, với 0 tương ứng với ô trống.