Cơ sở trí tuệ nhân tạo

Bài tập về tìm kiếm

Nguyễn Minh Đức

1712358

1/

Một phân xưởng gia công chi tiết máy có 3 máy, ký hiệu là P1, P2, P3 có khả năng gia công các chi tiết như nhau và sáu công việc với thời gian gia công (tính theo giờ) là t1 = 2, t2 = 5, t3 = 8, t4 = 1, t5 = 5, t6 = 1. Nêu phương pháp để giải bài toán trên theo nguyên lý thứ tự để thời gian hoàn thành các công việc kể trên của phân xưởng có thể đảm bảo thời gian ngắn nhất?

Em đề xuất một thuật giải heuristic với nguyên lý thứ tự như sau:

- Lần lượt đưa các công việc vào các máy theo nguyên tắc: Đưa công việc với thời gian dài nhất hiện tại vào máy có tổng thời gian gia công ngắn nhất.

Ban đầu công việc có thời gian dài nhất là t3 = 8, các máy hiện tại đều có tổng thời gian gia công bằng 0 → ta giao t3 cho P1

P1	t3	t3	t3	t3	t3	t3	t3	t3
P2								
P3								

Hiện tại công việc có thời gian dài nhất là t2 = 5, máy P2 P3 đều có tổng thời gian gia công bằng 0 → ta giao t2 cho P2

P1	t3	t3	t3	t3	t3	t3	t3	t3
P2	t2	t2	t2	t2	t2			
P3								

Hiện tại công việc có thời gian dài nhất là t5 = 5, máy P3 có tổng thời gian gia công ngắn nhất)→ ta giao t5 cho P3

P1	t3	t3	t3	t3	t3	t3	t3	t3
P2	t2	t2	t2	t2	t2			
P3	t5	t5	t5	t5	t5			

Hiện tại công việc có thời gian dài nhất là t1 = 2, máy P2 có tổng thời gian gia công ngắn nhất → ta giao t1 cho P2

P1	t3	t3	t3	t3	t3	t3	t3	t3
P2	t2	t2	t2	t2	t2	t1	t1	
P3	t5	t5	t5	t5	t5			

Tương tự ta giao t4 cho P3

P1	t3	t3	t3	t3	t3	t3	t3	t3
P2	t2	t2	t2	t2	t2	t1	t1	
P3	t5	t5	t5	t5	t5	t4		

Giao t6 cho P3

P1	t3	t3	t3	t3	t3	t3	t3	t3
P2	t2	t2	t2	t2	t2	t1	t1	
P3	t5	t5	t5	t5	t5	t4	t6	

Thuật giải cho kết quả thời gian tìm được bằng 8, đúng bằng t3 → đây là kết quả optimal (do không thể gia công với thời gian nhỏ hơn thời gian của một công việc).

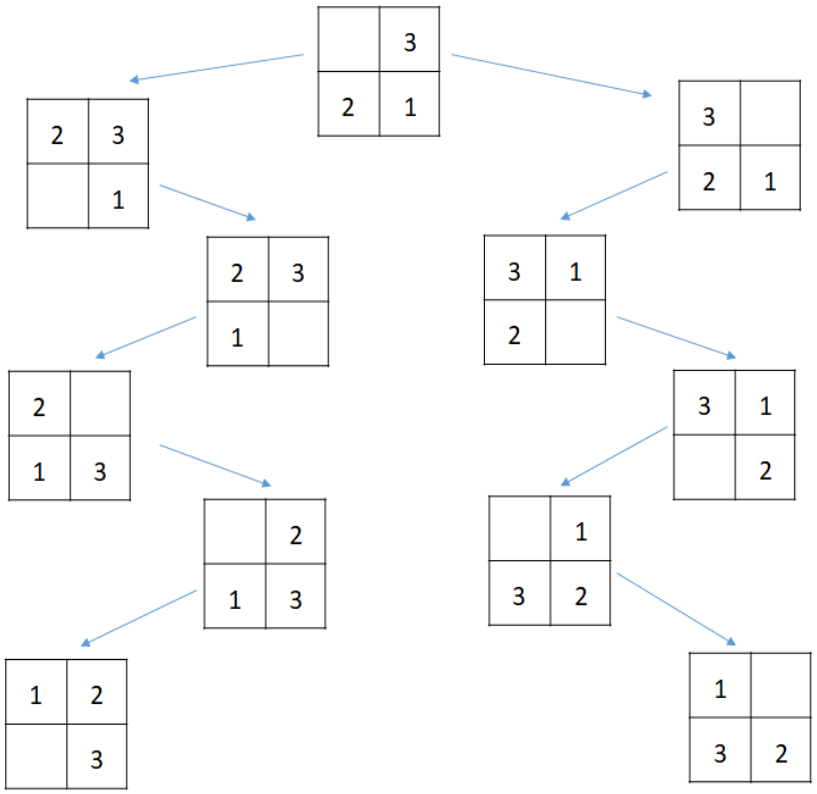
Trong thực tế thì nguyên lý thứ tự này không đưa ra được kết quả optimal nhưng thuật giải chạy với tốc độ rất nhanh. Nếu bước sắp xếp các công việc ban đầu mất thời gian cỡ O(n) (dùng counting sort hoặc radix sort), và bước sếp các công việc vào máy mất thời gian O($n \log_2 m$) (dùng một priority queue để lưu thời gian gia của các máy hiện hành). → Độ phức tạp của thuật giải vào cỡ O($n \log_2 m$), gần tuyến tính theo n. Vì bài toán được chứng minh là thuộc lớp NP-hard nên các thuật toán optimal tốt nhất hiện tại chắc chắn chạy trong thời

gian bằng hoặc lớn hơn hàm mũ. Vậy nên thuật giải heuristic ở trên khá lý tưởng vì thời gian chạy rất nhanh mà lại có kết quả gần với kết quả tối ưu.

2/

a/ tìm kiếm theo chiều rộng (BFS)

Mặc dù có sự ưu tiên giữa các phép dịch chuyển nhưng do tại một trạng thái thì chỉ có thể có 2 cách dịch chuyển trái ngược nhau, mà ta không thể đi ngược lại bước dịch chuyển trước đó (chính xác là không nên) → chỉ còn 1 cách dịch chuyển duy nhất (trừ trạng thái đầu tiên) nên các bước đi của thuật toán tách thành 2 nhánh riêng biệt (chung gốc) như sau:

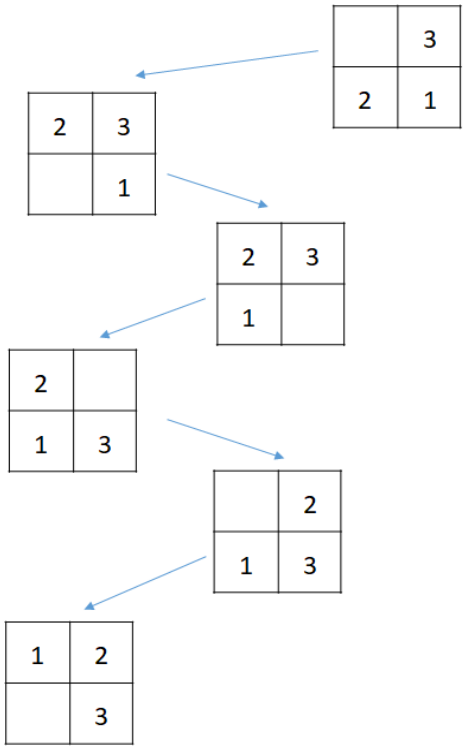


Các trạng ở cùng hàng thuộc frontier tại một thời điểm nào đó của thuật toán.

Kết quả nhận được đáp án optimal: cần ít nhất 5 bước di chuyển để tới trạng thái đích (các bước cụ thể được biểu diễn ở nhánh bên trái).

b/ tìm kiếm theo chiều sâu (DFS)

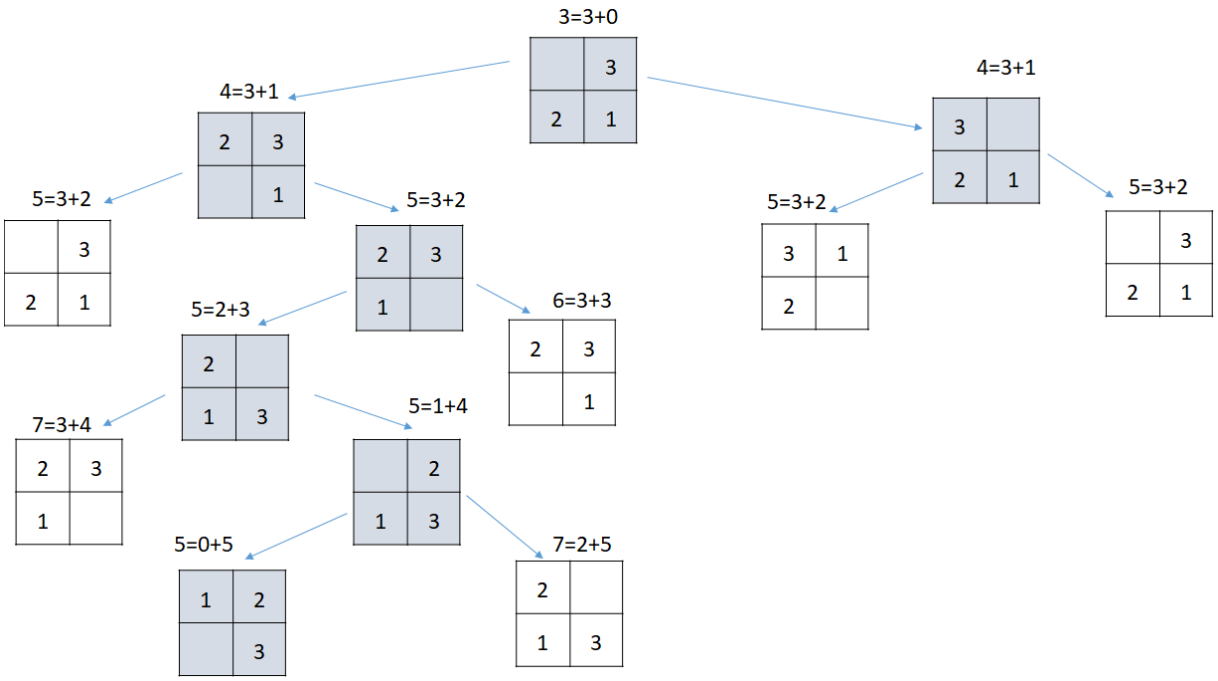
Cũng với lập luận như trong câu a), tại mỗi trạng thái bất kỳ (trừ trạng thái đầu tiên) chỉ tồn tại một nước đi kế tiếp duy nhất. Tại trạng thái đầu tiên, ta ưu tiên bước đi "up" thì được kết quả như sau. Nếu sau khi đi tất cả những bước có thể mà vẫn không tới được đích thì ta sẽ quay lại trạng thái đầu và đi "left".



May mắn là ta đã tìm được trạng thái đích với kết quả là optimal (vì cùng kết quả với BFS – 5 bước di chuyển).

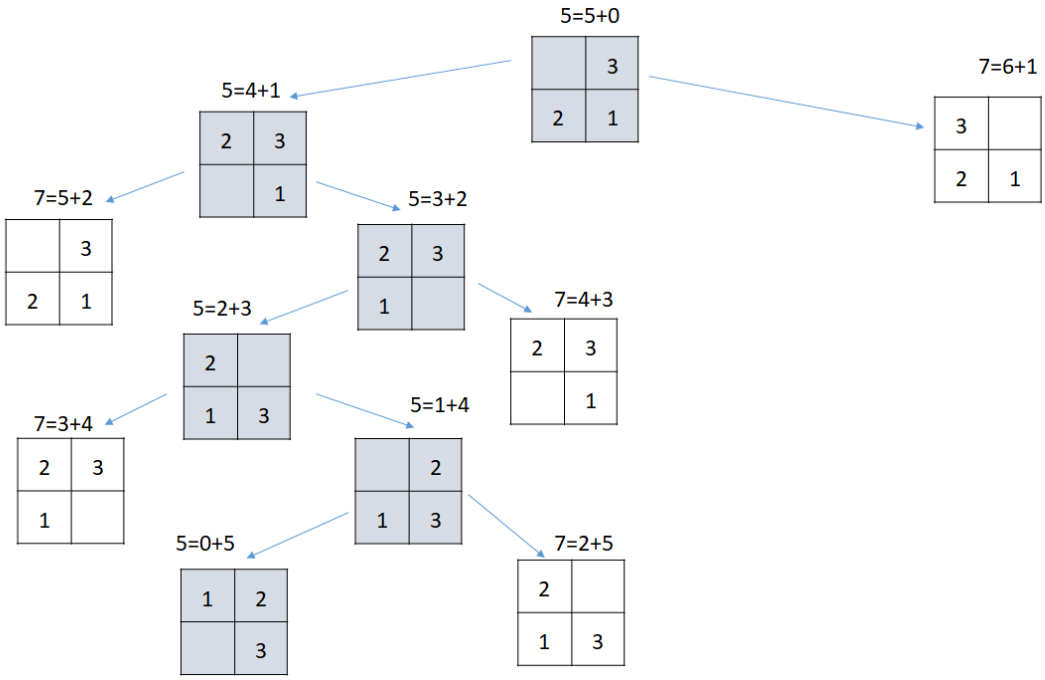
c/ A*

Dùng hàm lượng giá h1: Tổng số mảnh bị sai vị trí



Các trạng thái tô màu xanh là các trạng được chọn để đi tiếp tại một thời điểm nào đó của thuật toán. Ta thấy rằng A* với hàm lượng giá h1 đã cho kết quả optimal – 5 bước di chuyển để tới được trạng thái đích.

Dùng hàm lượng giá h2: Tổng khoảng cách Manhattan của các miếng giữa trạng thái đầu và trạng thái đích



Hàm h2 có vẻ hiệu quả hơn h1 khi không cần mở 1 trạng thái phía nhánh bên phải. A* cùng h2 cũng đưa ra kết quả optimal.

3/

Áp dụng thuật toán các thuật toán sau

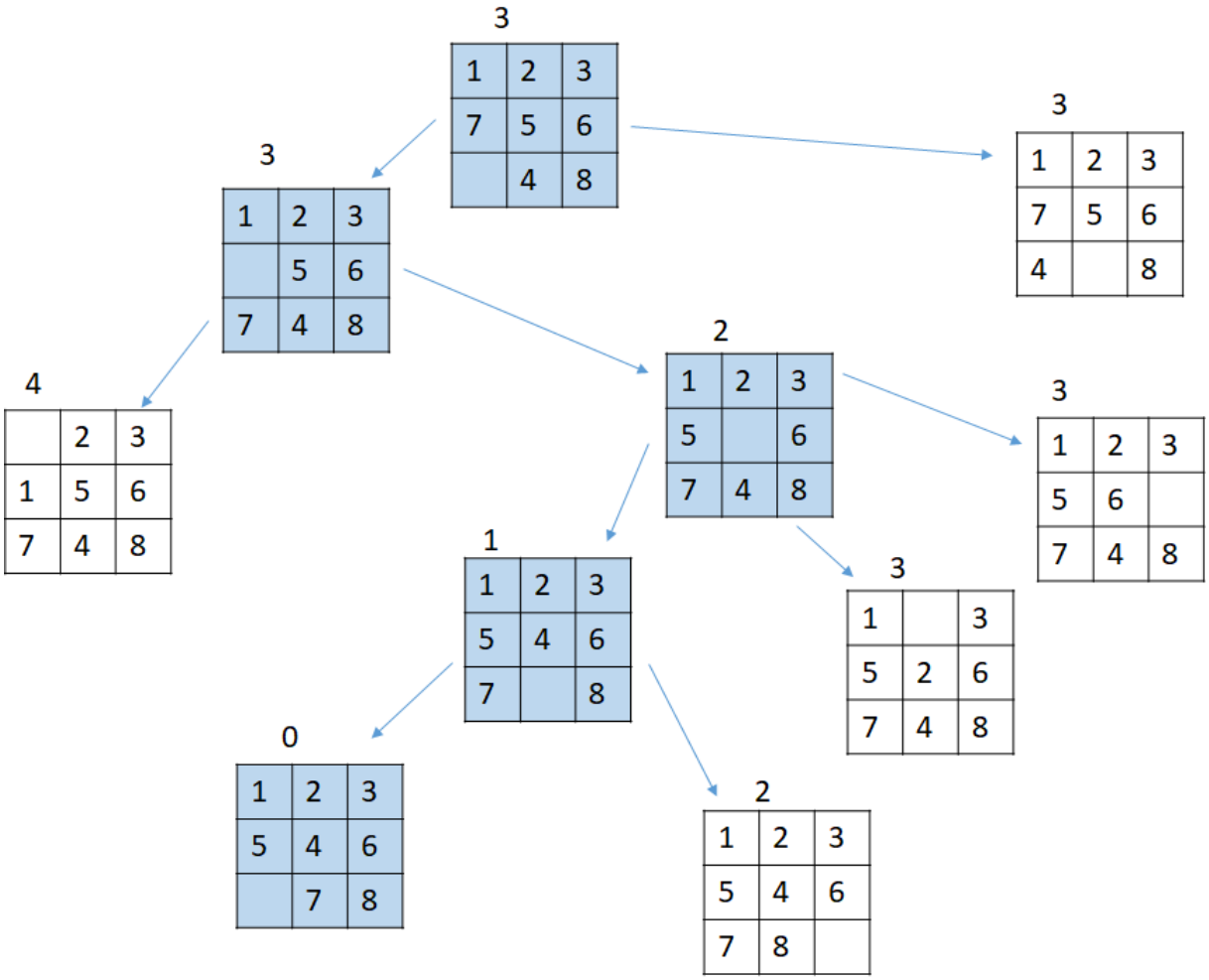
a) tìm kiếm tốt nhất tham lam với các hàm heuristic khác nhau (đã học)

b) A* với các hàm heuristic như trên

để giải bài toán 8-puzzle với độ ưu tiên cho các di chuyển: up, down, left, right

a/

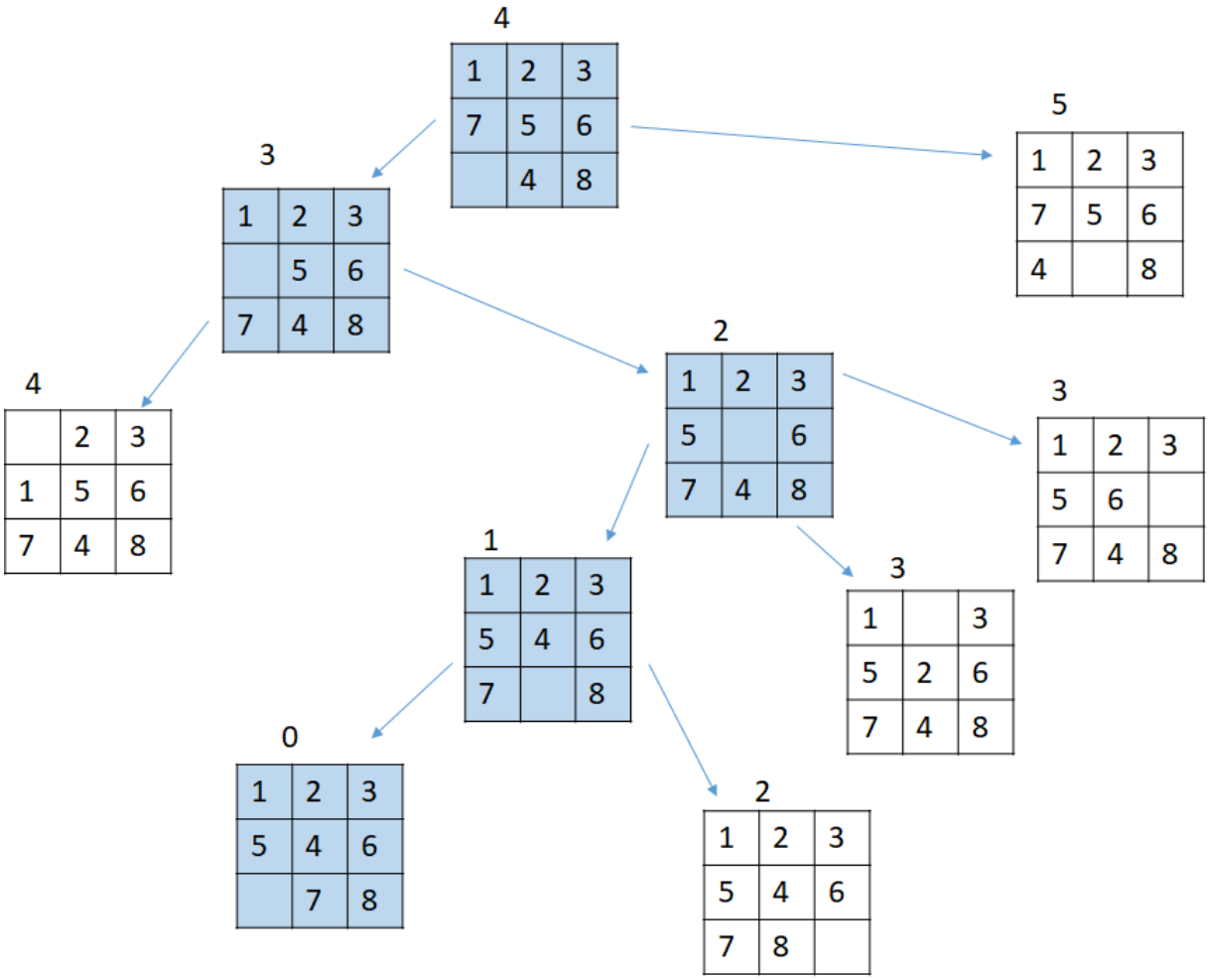
Dùng h1: số lượng mảnh sai vị trí



Thuật toán đưa ra kết quả: cần 4 bước đi để tới trạng thái đích.

Để không rơi vào loop em đã search theo kiểu graph search (tức là không đi vào những trạng thái đã đi qua). Độ phân nhánh của thuật toán khá thấp → chạy nhanh, nhưng kết quả có thể không optimal

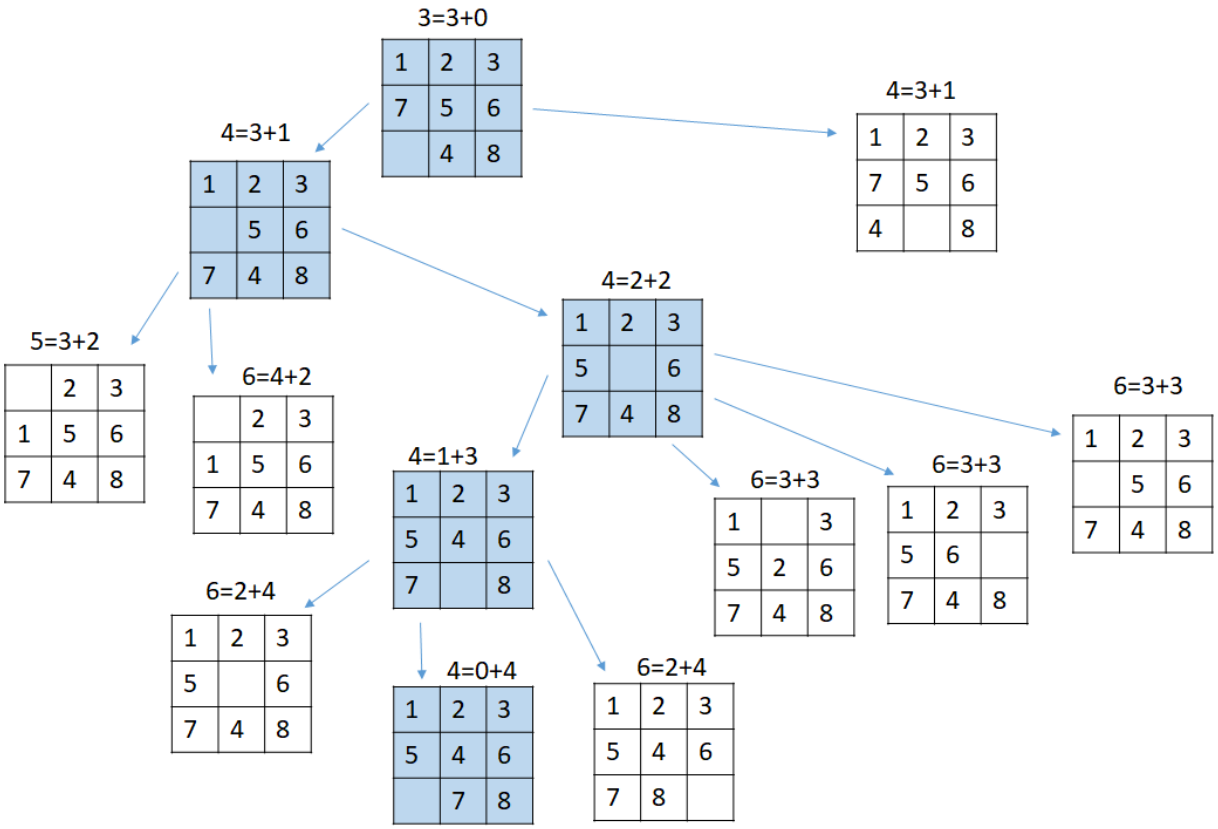
Dùng h2: Tổng khoảng cách Manhattan của các miếng giữa trạng thái đầu và trạng thái đích



Thuật toán cho kết quả giống với khi dùng h1. Nhưng để ý kỹ ta sẽ thấp các giá trị f khác biệt nhau nhiều hơn và ít bị trùng nhau nên độ phân nhánh sẽ còn nhỏ hơn khi dùng h1 → có tốc độ nhanh hơn, và cũng như h1, đây là greedy best first search nên không đảm bảo một kết quả optimal

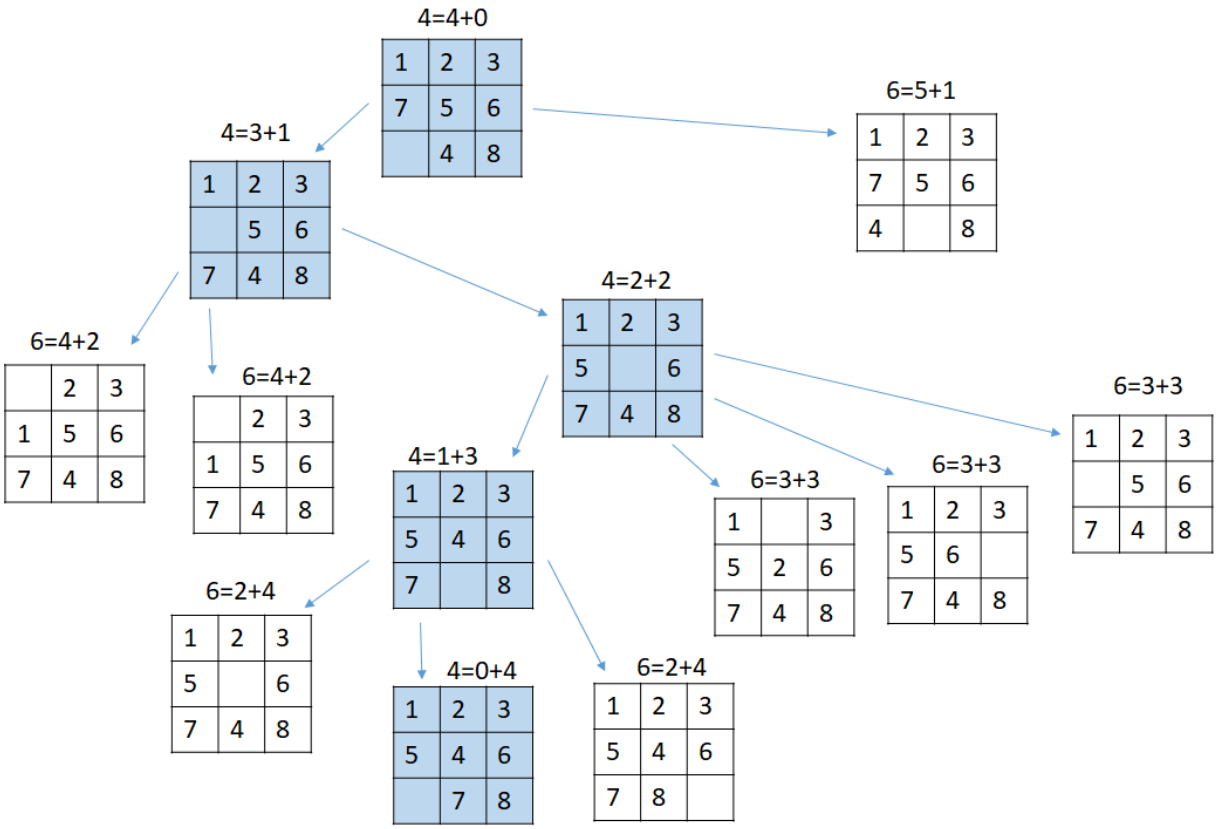
b/

Dùng h1: số lượng mảnh sai vị trí



Ta được kết quả: cần 4 bước đi để tới được trạng thái đích. Ở đây ta thấy thuật toán chạy khá may mắn khi không mở trạng thái bên nhánh phải ngay từ đầu dù cho trạng thái này có f=4, bởi ta đã ưu tiên bước đi "down" hơn bước đi "left".

Dùng h2: Tổng khoảng cách Manhattan của các miếng giữa trạng thái đầu và trạng thái đích



Thuật toán cũng cho ra kết quả optimal : cần 4 bước di chuyển để tới trạng thái đích.

Và rõ ràng hàm lượng giá h2 tốt hơn h1 khi trạng thái ở nhánh phải trên cùng có f=6, lớn hơn hẳn so với nhánh trái, nên thuật toán sẽ có sự lựa chắc chắn hơn mà không phải dựa vào may mắn.