

```
1 # 🧩 Đồ án cá nhân: 8-Puzzle Solver
2
3 ## 🎯 Mục tiêu
4 Xây dựng một chương trình giải bài toán **8-Puzzle** sử dụng nhiều thuật ➤
   toán tìm kiếm khác nhau trong lĩnh vực Trí tuệ nhân tạo.
5
6 ---
7
8 ## 🧠 Các thuật toán được triển khai
9 ## Uninformed search algorithms
10 ---
11 ### 1. **Khái niệm chung về Uninformed Search Algorithms**
12 - **Uninformed Search** (tìm kiếm mù) là các thuật toán tìm kiếm không sử ➤
   dụng thông tin heuristic (thông tin bổ sung về chi phí ước lượng đến ➤
   mục tiêu). Chúng dựa vào cấu trúc của không gian tìm kiếm và các quy ➤
   tắc cố định để khám phá các trạng thái.
13 - **Các thành phần chính**:
```

- 14 - **Không gian trạng thái (State Space)**: Tập hợp tất cả các trạng ➤
 thái có thể có của bài toán.
- 15 - **Trạng thái ban đầu (Initial State)**: Điểm bắt đầu của bài toán.
- 16 - **Trạng thái mục tiêu (Goal State)**: Trạng thái cần đạt được.
- 17 - **Hành động (Actions)**: Các thao tác có thể thực hiện để chuyển từ ➤
 trạng thái này sang trạng thái khác.
- 18 - **Chi phí đường đi (Path Cost)**: Chi phí liên quan đến mỗi hành động ➤
 hoặc đường đi (nếu có).
- 19 - **Cấu trúc dữ liệu**: Thường sử dụng hàng đợi (queue), ngăn xếp ➤
 (stack) hoặc hàng đợi ưu tiên (priority queue) để quản lý các trạng ➤
 thái cần khám phá.

```
20 ---
21 ### 2. **Các thuật toán Uninformed Search**
22
23 ##### a. **Breadth-First Search (BFS - Tìm kiếm theo chiều rộng)**
24 - **Mô tả**: Khám phá tất cả các trạng thái ở độ sâu hiện tại trước khi ➤
   chuyển sang độ sâu tiếp theo. Sử dụng **hàng đợi (queue)** để lưu trữ ➤
   các trạng thái.
25 - **Cách hoạt động**:
```

- 26 1. Bắt đầu từ trạng thái ban đầu, thêm vào hàng đợi.
- 27 2. Lấy trạng thái đầu tiên trong hàng đợi, kiểm tra xem có phải trạng ➤
 thái mục tiêu không.
- 28 3. Nếu không, sinh tất cả các trạng thái con (successors) và thêm chúng ➤
 vào cuối hàng đợi.
- 29 4. Lặp lại cho đến khi tìm thấy mục tiêu hoặc hàng đợi rỗng.

```
30 - **Đặc điểm**:
```

- 31 - **Hoàn chỉnh (Complete)**: Tìm được giải pháp nếu tồn tại, với không ➤
 gian trạng thái hữu hạn.
- 32 - **Tối ưu (Optimal)**: Tìm đường đi ngắn nhất nếu chi phí hành động ➤
 đồng nhất.
- 33 - **Độ phức tạp**:
- 34 - Thời gian: $O(b^d)$, với b là số nhánh trung bình, d là độ sâu của ➤

giải pháp.

- 35 - Không gian: $O(b^d)$, do lưu trữ nhiều trạng thái ở mỗi mức.
- 36 - **Ứng dụng**: Tìm đường đi ngắn nhất trong đồ thị không trọng số, như [mê cung](#).

37

38 ##### b. **Depth-First Search (DFS - Tìm kiếm theo chiều sâu)**

- 39 - **Mô tả**: Khám phá trạng thái theo chiều sâu tối đa trước khi quay lại (backtrack). Sử dụng **ngăn xếp (stack)** hoặc đệ quy.
- 40 - **Cách hoạt động**:
 - 41 1. Bắt đầu từ trạng thái ban đầu, khám phá trạng thái con đầu tiên.
 - 42 2. Tiếp tục đi sâu vào một nhánh cho đến khi gặp ngõ cụt hoặc tìm thấy [mục tiêu](#).
 - 43 3. Quay lại (backtrack) để khám phá các nhánh khác.
- 44 - **Đặc điểm**:
 - 45 - **Hoàn chỉnh**: Không đảm bảo trong không gian vô hạn hoặc có chu kỳ, [trừ khi có cơ chế kiểm tra chu kỳ](#).
 - 46 - **Tối ưu**: Không tối ưu, vì có thể tìm đường đi dài hơn.
 - 47 - **Độ phức tạp**:
 - 48 - Thời gian: $O(b^m)$, với m là độ sâu tối đa của không gian trạng [thái](#).
 - 49 - Không gian: $O(bm)$, do chỉ lưu một đường đi tại một thời điểm.
- 50 - **Ứng dụng**: Tìm kiếm trong không gian lớn, như giải câu đố, khi không [cần đường đi tối ưu](#).

51

52 ##### c. **Uniform Cost Search (UCS - Tìm kiếm chi phí đồng nhất)**

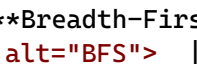
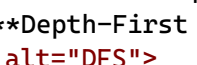
- 53 - **Mô tả**: Khám phá trạng thái theo chi phí đường đi tăng dần. Sử dụng [hàng đợi ưu tiên \(priority queue\)](#) dựa trên chi phí.
- 54 - **Cách hoạt động**:
 - 55 1. Bắt đầu từ trạng thái ban đầu, thêm vào hàng đợi ưu tiên với chi phí [0](#).
 - 56 2. Lấy trạng thái có chi phí thấp nhất từ hàng đợi, kiểm tra xem có [phải mục tiêu không](#).
 - 57 3. Sinh các trạng thái con, tính chi phí đường đi từ gốc, thêm vào hàng [đợi theo thứ tự chi phí](#).
 - 58 4. Lặp lại cho đến khi tìm thấy mục tiêu hoặc hàng đợi rỗng.
- 59 - **Đặc điểm**:
 - 60 - **Hoàn chỉnh**: Nếu chi phí hành động lớn hơn 0.
 - 61 - **Tối ưu**: Tìm đường đi có chi phí thấp nhất.
 - 62 - **Độ phức tạp**:
 - 63 - Thời gian: $O(b^{(C*/\epsilon)})$, với C^* là chi phí tối ưu, ϵ là chi phí hành [động nhỏ nhất](#).
 - 64 - Không gian: $O(b^{(C*/\epsilon)})$.
- 65 - **Ứng dụng**: Tìm đường đi tối ưu trong đồ thị có trọng số, như định [tuyến đường đi](#).

66

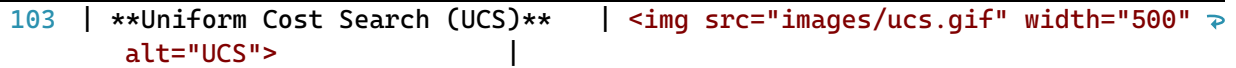
67 ##### d. **Iterative Deepening Search (IDS - Tìm kiếm sâu dần)**

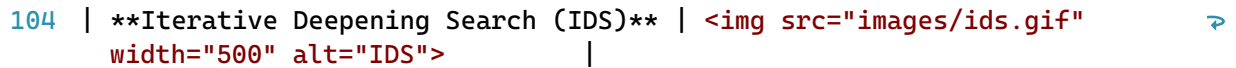
- 68 - **Mô tả**: Kết hợp ưu điểm của BFS và DFS, thực hiện DFS với giới hạn [độ sâu tăng dần](#).
- 69 - **Cách hoạt động**:

```

70 1. Thực hiện DFS với giới hạn độ sâu (depth limit) là 0.
71 2. Nếu không tìm thấy mục tiêu, tăng giới hạn độ sâu lên 1 và lặp lại.
72 3. Tiếp tục tăng giới hạn độ sâu cho đến khi tìm thấy mục tiêu.
73 - Đặc điểm:
74   - Hoàn chỉnh: Nếu không gian trạng thái hữu hạn.
75   - Tối ưu: Tìm đường đi ngắn nhất nếu chi phí hành động đồng nhất.
76   - Độ phức tạp:
77     - Thời gian:  $O(b^d)$ , tương tự BFS nhưng lặp lại nhiều lần.
78     - Không gian:  $O(bd)$ , tương tự DFS.
79 - Ứng dụng: Khi cần kết hợp ưu điểm của BFS (tối ưu) và DFS (tiết
    kiểm bộ nhớ).
80 ---
81 ### 3. So sánh tổng quát
82 | Thuật toán | Hoàn chỉnh | Tối ưu | Độ phức tạp thời gian | Độ phức tạp
    không gian | Ứng dụng chính |
83 |-----|-----|-----|-----|-----
    -----|-----|
84 | BFS | Có (nếu hữu hạn) | Có (nếu chi phí đồng nhất) |  $O(b^d)$  |  $O(b^d)$  | Đường đi ngắn nhất (không trọng số) |
85 | DFS | Không (nếu có chu kỳ) | Không |  $O(b^m)$  |  $O(bm)$  | Không
    gian lớn, không cần tối ưu |
86 | UCS | Có (nếu chi phí  $> 0$ ) | Có |  $O(b^{(C/\epsilon)})$  |  $O(b^{(C/\epsilon)})$  |
    Đường đi tối ưu (có trọng số) |
87 | IDS | Có (nếu hữu hạn) | Có (nếu chi phí đồng nhất) |  $O(b^d)$  |  $O(bd)$  | Kết hợp BFS và DFS |
88 ---
89 ### 4. Giải pháp tổng quát của Uninformed Search
90 - Quy trình chung:
91 1. Xác định trạng thái ban đầu và mục tiêu.
92 2. Xây dựng không gian trạng thái và các hành động có thể thực hiện.
93 3. Sử dụng cấu trúc dữ liệu (queue, stack, priority queue) để quản lý
    các trạng thái cần khám phá.
94 4. Áp dụng chiến lược chọn trạng thái (theo chiều rộng, chiều sâu, chi
    phí, hoặc sâu dần) để tìm đường đi từ trạng thái ban đầu đến mục
    tiêu.
95 - Ưu điểm: Đơn giản, không cần thông tin bổ sung (heuristic), phù hợp
    với các bài toán không có thông tin về chi phí ước lượng.
96 - Nhược điểm: Hiệu quả thấp trong không gian trạng thái lớn hoặc phức
    tạp, đặc biệt khi không có heuristic hỗ trợ.
97
98 ### 🖼️ Hình ảnh các thuật toán được áp dụng trong trò chơi
99 | Thuật Toán | Minh Họa GIF
100 |-----|-----
    -----|
101 | Breadth-First Search (BFS) | 
102 | Depth-First Search (DFS) | 

```

103 | ****Uniform Cost Search (UCS)**** |  **** ↗

104 | ****Iterative Deepening Search (IDS)**** |  **** ↗

105

106 ### 🔍 So sánh các thuật toán tìm kiếm không thông tin (Uninformed Search Algorithms) ↗

107

108 | ****Thuật toán**** | ****Hoàn chỉnh**** | ****Tối ưu**** | ****Bộ nhớ**** | ↗
****Thời gian**** | ****Phù hợp với 8-puzzle**** ↗

109 | ----- | ----- | ----- | ----- | ----- ↗
 ----- | ----- | ----- | ----- | ----- ↗
 ---- |

110 | ****BFS**** | Có | Có | Cao **'O(b^d)'** | Cao ↗
'O(b^d)' | ✓ Phù hợp nếu lời giải nông, nhưng tiêu tốn nhiều bộ nhớ ↗

111 | ****DFS**** | Không | Không | Thấp **'O(bm)'** | Biến ↗
 động **'O(b^m)'** | ✗ Không phù hợp, dễ bị kẹt và không tối ưu ↗

112 | ****UCS**** | Có | Có | Cao **'O(b^d)'** | Cao ↗
'O(b^d)' | ✓ Tìm giải pháp tối ưu, nhưng tốn tài nguyên ↗

113 | ****IDS**** | Có | Có | Thấp **'O(bd)'** | Cao ↗
'O(b^d)' | ✓ Thích hợp khi bộ nhớ hạn chế, nhưng chậm hơn BFS ↗

114

115 ****Chú thích:****

116 - **'b'**: số nhánh trung bình (branching factor)

117 - **'d'**: độ sâu của lời giải tối ưu

118 - **'m'**: độ sâu tối đa của cây tìm kiếm

119 ---

120 ### 📝 Nhận xét chung:

121

122 Các thuật toán tìm kiếm không thông tin (Uninformed Search) đều không có kiến thức cụ thể về vị trí đích, do đó phải duyệt toàn bộ không gian trạng thái một cách mù mờ. Mỗi thuật toán có đặc điểm riêng: ↗

123

124 * ****BFS**** thích hợp khi lời giải nằm ở độ sâu nhỏ, đảm bảo tìm được lời giải ngắn nhất nhưng ****tốn nhiều bộ nhớ****. ↗

125 * ****DFS**** có ưu điểm tiết kiệm bộ nhớ, nhưng ****dễ rơi vào vòng lặp vô tận**** và không đảm bảo tối ưu. ↗

126 * ****UCS**** mở rộng BFS bằng cách tính đến chi phí, cho phép tìm lời giải tối ưu khi chi phí không đồng đều, nhưng ****hiệu năng giảm nếu không gian tìm kiếm lớn****. ↗

127 * ****IDS**** kết hợp ưu điểm của BFS và DFS: đảm bảo tối ưu, tiết kiệm bộ nhớ, nhưng ****thời gian chạy lâu hơn do phải lặp lại nhiều lần****. ↗

128 Với bài toán như ****8-puzzle****, nơi không gian trạng thái lớn và cần lời giải tối ưu, ****BFS, UCS hoặc IDS**** là lựa chọn phù hợp. Tuy nhiên, khi ↗

bộ nhớ hạn chế, **IDS** thường là phương án an toàn hơn.

129 ---

130 ## Informed Search Algorithms

131 ---

132 ### 1. **Khái niệm chung về Informed Search Algorithms**

- 133 - **Informed Search** (tìm kiếm có thông tin) sử dụng **hàm heuristic** để ước lượng chi phí từ trạng thái hiện tại đến trạng thái mục tiêu, giúp định hướng tìm kiếm hiệu quả hơn so với Uninformed Search (BFS, DFS, UCS, IDS). ➤
- 134 - **Các thành phần chính**: ➤
- 135 - **Không gian trạng thái (State Space)**: Tập hợp tất cả các trạng thái có thể có của bài toán (ví dụ: các hoán vị của ô trong 8-puzzle). ➤
- 136 - **Trạng thái ban đầu (Initial State)**: Điểm xuất phát của bài toán. ➤
- 137 - **Trạng thái mục tiêu (Goal State)**: Trạng thái cần đạt được. ➤
- 138 - **Hành động (Actions)**: Các thao tác hợp lệ để chuyển đổi giữa các trạng thái (ví dụ: di chuyển ô trống lên, xuống, trái, phải). ➤
- 139 - **Chi phí đường đi (Path Cost, $g(n)$)**: Tổng chi phí từ trạng thái ban đầu đến trạng thái hiện tại (thường là số bước hoặc chi phí cụ thể của hành động). ➤
- 140 - **Hàm heuristic ($h(n)$)**: Hàm ước lượng chi phí từ trạng thái hiện tại đến mục tiêu. Hàm này phải **admissible** (không overestimated) và lý tưởng là **consistent** (đáp ứng bất đẳng thức tam giác) để đảm bảo tính tối ưu. ➤
- 141 - **Cấu trúc dữ liệu**: Thường sử dụng hàng đợi ưu tiên (priority queue) để ưu tiên trạng thái có chi phí thấp nhất hoặc giá trị heuristic nhỏ nhất. ➤

142 ---

143 ### 2. **Các thuật toán Informed Search**

144

145 #### a. **A* Search**

- 146 - **Mô tả**: A* sử dụng hàm đánh giá **$f(n) = g(n) + h(n)$** : ➤
- 147 - **$g(n)$** : Chi phí thực tế từ trạng thái ban đầu đến trạng thái hiện tại. ➤
- 148 - **$h(n)$** : Chi phí ước lượng từ trạng thái hiện tại đến mục tiêu (ví dụ: khoảng cách Manhattan trong 8-puzzle). ➤
- 149 - A* ưu tiên khám phá trạng thái có **$f(n)$** nhỏ nhất, đảm bảo đường đi tối ưu nếu heuristic là admissible. ➤
- 150 - **Cách hoạt động**: ➤
- 151 1. Bắt đầu từ trạng thái ban đầu, thêm vào hàng đợi ưu tiên với chi phí **$f(n) = g(n) + h(n)$** . ➤
- 152 2. Lấy trạng thái có **$f(n)$** nhỏ nhất từ hàng đợi, kiểm tra xem có phải trạng thái mục tiêu không. ➤
- 153 3. Sinh các trạng thái con, tính **$g(n)$** và **$h(n)$** cho mỗi trạng thái, thêm vào hàng đợi. ➤
- 154 4. Lặp lại cho đến khi tìm thấy mục tiêu hoặc hàng đợi rỗng. ➤
- 155 - **Đặc điểm**: ➤
- 156 - **Hoàn chỉnh**: Có, nếu không gian trạng thái hữu hạn và chi phí hành động lớn hơn 0. ➤

- 157 - ****Tối ưu****: Có, nếu heuristic là admissible ($h(n) \leq$ chi phí thực tế đến mục tiêu). ➤
- 158 - ****Độ phức tạp****:
- 159 - Thời gian: $O(b^d)$, nhưng thường nhanh hơn BFS/UCS nhờ heuristic định hướng. ➤
- 160 - Không gian: $O(b^d)$, do lưu trữ nhiều trạng thái trong hàng đợi ưu tiên. ➤
- 161 - ****Ứng dụng****: Tìm đường đi tối ưu trong các bài toán như 8-puzzle, định tuyến, hoặc lập kế hoạch, khi cần đảm bảo chi phí thấp nhất. ➤
- 162
- 163 ##### b. ****Iterative Deepening A* (IDA*)****
- 164 - ****Mô tả****: IDA* kết hợp ý tưởng của A* và Iterative Deepening Search (IDS). Nó sử dụng hàm $f(n) = g(n) + h(n)$ nhưng giới hạn tìm kiếm theo ngưỡng $f(n)$ tăng dần, thực hiện tìm kiếm theo chiều sâu (DFS) trong mỗi lần lặp. ➤
- 165 - ****Cách hoạt động****:
- 166 1. Bắt đầu với ngưỡng ban đầu là $f(n) = h(n)$ của trạng thái ban đầu. ➤
- 167 2. Thực hiện DFS, chỉ khám phá các trạng thái có $f(n) \leq$ ngưỡng. ➤
- 168 3. Nếu không tìm thấy mục tiêu, tăng ngưỡng lên giá trị $f(n)$ nhỏ nhất vượt ngưỡng hiện tại, lặp lại. ➤
- 169 4. Tiếp tục cho đến khi tìm thấy mục tiêu hoặc không còn trạng thái để khám phá. ➤
- 170 - ****Đặc điểm****:
- 171 - ****Hoàn chỉnh****: Có, nếu không gian trạng thái hữu hạn. ➤
- 172 - ****Tối ưu****: Có, nếu heuristic là admissible. ➤
- 173 - ****Độ phức tạp****:
- 174 - Thời gian: $O(b^d)$, nhưng có thể chậm hơn A* do lặp lại nhiều lần. ➤
- 175 - Không gian: $O(bd)$, tiết kiệm bộ nhớ hơn A* vì chỉ lưu một đường đi tại mỗi lần lặp. ➤
- 176 - ****Ứng dụng****: Phù hợp cho các bài toán như 8-puzzle khi bộ nhớ hạn chế, nhưng cần giải pháp tối ưu. ➤
- 177
- 178 ##### c. ****Greedy Best-First Search (Greedy)****
- 179 - ****Mô tả****: Greedy ưu tiên khám phá trạng thái có giá trị **heuristic** $h(n)$ nhỏ nhất, bỏ qua chi phí đường đi $g(n)$. Nó tập trung vào việc tiến gần trạng thái mục tiêu nhanh nhất có thể. ➤
- 180 - ****Cách hoạt động****:
- 181 1. Bắt đầu từ trạng thái ban đầu, thêm vào hàng đợi ưu tiên với giá trị $h(n)$. ➤
- 182 2. Lấy trạng thái có $h(n)$ nhỏ nhất, kiểm tra xem có phải mục tiêu không. ➤
- 183 3. Sinh các trạng thái con, tính $h(n)$ cho mỗi trạng thái, thêm vào hàng đợi. ➤
- 184 4. Lặp lại cho đến khi tìm thấy mục tiêu hoặc hàng đợi rỗng. ➤
- 185 - ****Đặc điểm****:
- 186 - ****Hoàn chỉnh****: Không, có thể bị kẹt trong các vòng lặp hoặc bỏ sót giải pháp. ➤
- 187 - ****Tối ưu****: Không, vì không xem xét chi phí đường đi $g(n)$, có thể dẫn đến đường đi dài hơn. ➤

```

188 - **Độ phức tạp**:
189 - Thời gian:  $O(b^m)$ , với  $m$  là độ sâu tối đa, nhưng thường nhanh hơn ↗
    A* do chỉ dựa vào 'h(n)'.
190 - Không gian:  $O(b^m)$ , tùy thuộc vào số trạng thái được lưu trữ.
191 - **Ứng dụng**: Dùng khi cần tìm giải pháp nhanh, không yêu cầu tối ưu, ↗
    như trong một số bài toán tìm kiếm đơn giản hoặc khi thời gian thực thi ↗
    là ưu tiên.
192
193 ---
194
195 ### 3. **So sánh tổng quát**
196 | Thuật toán | Hoàn chỉnh | Tối ưu | Độ phức tạp thời gian | Độ phức tạp ↗
    không gian | Ứng dụng chính |
197 |-----|-----|-----|-----|----- ↗
    -----|-----|
198 | **A** | Có | Có |  $O(b^d)$  |  $O(b^d)$  ↗
    | Tìm đường đi tối ưu (8-puzzle, định tuyến) |
199 | **IDA** | Có | Có |  $O(b^d)$  |  $O(bd)$  ↗
    | Tìm đường đi tối ưu, tiết kiệm bộ nhớ |
200 | **Greedy** | Không | Không |  $O(b^m)$  |  $O(b^m)$  ↗
    | Tìm giải pháp nhanh, không cần tối ưu |
201
202 ---
203
204 ### 4. **Giải pháp tổng quát của Informed Search**
205 - **Quy trình chung**:
206 1. Xác định trạng thái ban đầu, trạng thái mục tiêu, và các hành động ↗
    có thể thực hiện.
207 2. Xây dựng hàm heuristic (ví dụ: khoảng cách Manhattan cho 8-puzzle) ↗
    để ước lượng chi phí.
208 3. Sử dụng hàng đợi ưu tiên hoặc chiến lược DFS với ngưỡng để quản lý ↗
    các trạng thái cần khám phá.
209 4. Áp dụng chiến lược chọn trạng thái:
210 - **A**: Dựa trên 'f(n) = g(n) + h(n)'.
211 - **IDA**: DFS với ngưỡng 'f(n)' tăng dần.
212 - **Greedy**: Dựa trên 'h(n)' nhỏ nhất.
213 5. Tìm đường đi từ trạng thái ban đầu đến mục tiêu, ưu tiên các trạng ↗
    thái có chi phí hoặc heuristic thấp.
214 - **Ưu điểm**:
215 - Hiệu quả hơn Uninformed Search nhờ heuristic định hướng.
216 - A* và IDA* đảm bảo tối ưu nếu heuristic là admissible.
217 - IDA* tiết kiệm bộ nhớ, phù hợp cho các bài toán lớn.
218 - Greedy nhanh, phù hợp khi không cần tối ưu.
219 - **Nhược điểm**:
220 - A* tốn bộ nhớ do lưu trữ nhiều trạng thái.
221 - IDA* có thể chậm do lặp lại nhiều lần.
222 - Greedy không đảm bảo hoàn chỉnh hoặc tối ưu, dễ bị kẹt trong các cực ↗
    trị cục bộ.
223 - **Yêu cầu**:

```

```

...m_23110250_DoAnCaNhanTriTueNhanTao_8_puzzle\README.md 8
224 - Cần thiết kế hàm heuristic phù hợp (admissible và consistent cho A* ➤
    và IDA*).
225 - Kiểm tra chu kỳ hoặc trạng thái lặp để tránh vòng lặp vô hạn.
226 ---
227
228 ### 📷 **Hình ảnh các thuật toán được áp dụng trong trò chơi**
229 | **Thuật Toán** | **Minh Họa GIF** | ➤
    |
230 |-----|-----| ➤
    |
231 | **A* Search (A-Star)** | 20) |
241 | **IDA*** | Có | Có | 'O(b^d)' | ➤
    | 'O(b^d)' | ✓ Tiết kiệm bộ nhớ, phù hợp cho | ➤
    hệ thống hạn chế tài nguyên. Chậm hơn A* ở độ sâu lớn. | ✓ | ➤
    Tối ưu, tiết kiệm bộ nhớ | ✗ Chậm hơn A* do | ➤
    phải lặp lại nhiều lần |
242 | **Greedy Best-First** | Không | Không | 'O(b^m)' | ➤
    | 'O(b^m)' | ✓ Nhanh, nhưng dễ bị kẹt hoặc | ➤
    tìm đường không tối ưu. Phù hợp khi cần kết quả nhanh. | ✓ | ➤
    Nhanh, đơn giản | ✗ Không tối ưu, có | ➤
    thể bỏ sót lời giải tốt hơn |
243
244 ### **Chú thích:**

```



245 * ``b``: Số nhánh trung bình (trong 8-puzzle, thường $\approx 2-4$ tùy vị trí ô trống).

246 * ``d``: Độ sâu của lời giải tối ưu.

247 * ``m``: Độ sâu tối đa của không gian trạng thái.

248 * **Heuristic sử dụng**: *Khoảng cách Manhattan* là heuristic **admissible** và **consistent**, đảm bảo tính tối ưu cho thuật toán **A*** và **IDA***.

249 ---

250  **Nhận xét chung**:

251

252 Các thuật toán **tìm kiếm có thông tin (Informed Search)** như **A***, **IDA*** và **Greedy Best-First Search** tận dụng heuristic để hướng dẫn quá trình tìm kiếm hiệu quả hơn so với các thuật toán không thông tin.


253

254 * **A*** là lựa chọn **tối ưu nhất** nếu hệ thống có đủ bộ nhớ, nhờ vào tính chất tối ưu và nhanh nhờ sử dụng heuristic tốt (ví dụ: Manhattan).

255 * **IDA*** phù hợp cho các môi trường **hạn chế tài nguyên** (như thiết bị nhúng, bộ nhớ thấp), vẫn đảm bảo tối ưu nhưng **hy sinh tốc độ** vì phải lặp lại nhiều lần.

256 * **Greedy Best-First Search** hoạt động **nhanh và đơn giản**, tuy nhiên **thiếu tính tối ưu**, dễ rơi vào bẫy cực bộ (local minima) nếu heuristic không tốt.

257

258  **Tóm lại**:

259

260 * Nếu **ưu tiên chất lượng lời giải** và **có đủ tài nguyên**, hãy chọn **A***.

261 * Nếu **ưu tiên tiết kiệm bộ nhớ**, chọn **IDA***.

262 * Nếu **cần kết quả nhanh** và **không quá quan tâm tối ưu**, có thể thử **Greedy**.

263 ---

264

265 **## Local Search Algorithms**

266 ---

267 **### 1. Khái niệm chung về Local Search Algorithms**

268 - **Local Search** (tìm kiếm cục bộ) tập trung vào việc cải thiện một giải pháp hiện tại bằng cách khám phá các trạng thái lân cận, thay vì khám phá toàn bộ không gian trạng thái như các thuật toán Informed/Uninformed Search.

269 - Không duy trì một cây tìm kiếm hoặc hàng đợi các trạng thái, mà chỉ làm việc với trạng thái hiện tại và các trạng thái lân cận của nó.

270 - Thường sử dụng trong các bài toán tối ưu, khi không gian trạng thái lớn và mục tiêu là tìm giải pháp tốt (không nhất thiết tối ưu toàn cục).

271 - **Các thành phần chính**:

272 - **Không gian trạng thái (State Space)**: Tập hợp tất cả các trạng thái có thể có (ví dụ: các hoán vị của ô trong 8-puzzle).

273 - **Trạng thái ban đầu (Initial State)**: Một giải pháp khởi đầu, thường được chọn ngẫu nhiên hoặc cố định.

- 274 - **Trạng thái mục tiêu (Goal State)**: Trạng thái lý tưởng hoặc tiêu chí tối ưu (ví dụ: trạng thái mục tiêu trong 8-puzzle hoặc giá trị hàm mục tiêu tối ưu). ↗
- 275 - **Hành động (Actions)**: Các thao tác để chuyển từ trạng thái hiện tại sang trạng thái lân cận (ví dụ: di chuyển ô trống trong 8-puzzle). ↗
- 276 - **Hàm mục tiêu (Objective Function)**: Đánh giá chất lượng của trạng thái, thường là hàm heuristic (như khoảng cách Manhattan) hoặc một hàm đánh giá khác. Trong tối ưu, có thể là tối thiểu hóa hoặc tối đa hóa giá trị hàm. ↗
- 277 - **Lân cận (Neighborhood)**: Tập hợp các trạng thái có thể đạt được từ trạng thái hiện tại bằng một hành động. ↗
- 278
- 279 ---
- 280
- 281 **### 2. Các thuật toán Local Search**
- 282
- 283 **#### a. Simple Hill Climbing**
- 284 - **Mô tả**: Chọn trạng thái lân cận đầu tiên có giá trị hàm mục tiêu tốt hơn trạng thái hiện tại (tối ưu hóa cục bộ). ↗
- 285 - **Cách hoạt động**:
- 286 1. Bắt đầu từ trạng thái ban đầu.
- 287 2. Đánh giá các trạng thái lân cận, chọn trạng thái đầu tiên có giá trị hàm mục tiêu tốt hơn (ví dụ: khoảng cách Manhattan nhỏ hơn). ↗
- 288 3. Chuyển sang trạng thái lân cận đó, lặp lại cho đến khi không tìm thấy trạng thái lân cận nào tốt hơn (đỉnh cục bộ). ↗
- 289 - **Đặc điểm**:
- 290 - **Hoàn chỉnh**: Không, dễ bị kẹt ở cực trị cục bộ.
- 291 - **Tối ưu**: Không, chỉ tìm giải pháp cục bộ.
- 292 - **Độ phức tạp**:
- 293 - Thời gian: Phụ thuộc vào số lân cận và số lần lặp, thường thấp ($O(k)$ mỗi bước, với k là số lân cận). ↗
- 294 - Không gian: $O(1)$, chỉ lưu trạng thái hiện tại và lân cận.
- 295 - **Ứng dụng**: Tìm giải pháp nhanh trong các bài toán như 8-puzzle, tối ưu hóa hàm đơn giản. ↗
- 296
- 297 **#### b. Steepest-Ascent Hill Climbing**
- 298 - **Mô tả**: Xem xét tất cả các trạng thái lân cận và chọn trạng thái có giá trị hàm mục tiêu tốt nhất (tối ưu hóa cục bộ). ↗
- 299 - **Cách hoạt động**:
- 300 1. Bắt đầu từ trạng thái ban đầu.
- 301 2. Đánh giá tất cả các trạng thái lân cận, chọn trạng thái có giá trị hàm mục tiêu tốt nhất (ví dụ: khoảng cách Manhattan nhỏ nhất). ↗
- 302 3. Chuyển sang trạng thái tốt nhất, lặp lại cho đến khi không có trạng thái lân cận nào tốt hơn. ↗
- 303 - **Đặc điểm**:
- 304 - **Hoàn chỉnh**: Không, có thể bị kẹt ở cực trị cục bộ.
- 305 - **Tối ưu**: Không, nhưng thường tốt hơn Simple Hill Climbing do chọn trạng thái lân cận tốt nhất. ↗

- 306 - ****Độ phức tạp****:
- 307 - Thời gian: $O(k)$ mỗi bước, với k là số lân cận, nhưng tốn thời gian [↗](#)
hơn Simple Hill Climbing do đánh giá tất cả lân cận.
- 308 - Không gian: $O(k)$, để lưu danh sách lân cận.
- 309 - ****Ứng dụng****: Phù hợp cho các bài toán như 8-puzzle khi cần cải thiện [↗](#)
chất lượng giải pháp so với Simple Hill Climbing.
- 310
- 311 ##### c. ****Stochastic Hill Climbing****
- 312 - ****Mô tả****: Chọn ngẫu nhiên một trạng thái lân cận có giá trị hàm mục [↗](#)
tiêu tốt hơn trạng thái hiện tại, thay vì chọn trạng thái tốt nhất.
- 313 - ****Cách hoạt động****:
- 314 1. Bắt đầu từ trạng thái ban đầu.
- 315 2. Tạo danh sách các trạng thái lân cận tốt hơn trạng thái hiện tại [↗](#)
(dựa trên hàm mục tiêu).
- 316 3. Chọn ngẫu nhiên một trạng thái từ danh sách đó, chuyển sang trạng [↗](#)
thái này.
- 317 4. Lặp lại cho đến khi không có trạng thái lân cận nào tốt hơn.
- 318 - ****Đặc điểm****:
- 319 - ****Hoàn chỉnh****: Không, vẫn có thể bị kẹt ở cực trị cục bộ.
- 320 - ****Tối ưu****: Không, nhưng tính ngẫu nhiên giúp tránh một số cực trị [↗](#)
cục bộ so với Simple/Steepest Hill Climbing.
- 321 - ****Độ phức tạp****:
- 322 - Thời gian: $O(k)$ mỗi bước, nhưng có thể nhanh hơn Steepest do không [↗](#)
cần đánh giá tất cả lân cận.
- 323 - Không gian: $O(k)$, để lưu danh sách lân cận tốt hơn.
- 324 - ****Ứng dụng****: Dùng khi muốn cân bằng giữa tốc độ và khả năng thoát khỏi [↗](#)
cực trị cục bộ, như trong 8-puzzle hoặc bài toán tối ưu hóa.
- 325
- 326 ##### d. ****Simulated Annealing****
- 327 - ****Mô tả****: Kết hợp tìm kiếm cục bộ với cơ chế ngẫu nhiên để thoát khỏi [↗](#)
cực trị cục bộ, sử dụng khái niệm "nhiệt độ" (temperature) để điều [↗](#)
khiển mức độ chấp nhận các trạng thái xấu hơn.
- 328 - ****Cách hoạt động****:
- 329 1. Bắt đầu từ trạng thái ban đầu, thiết lập nhiệt độ ban đầu cao và tốc [↗](#)
độ giảm nhiệt độ (cooling rate).
- 330 2. Chọn ngẫu nhiên một trạng thái lân cận.
- 331 3. Chấp nhận trạng thái lân cận nếu:
- 332 - Nó tốt hơn trạng thái hiện tại (theo hàm mục tiêu).
- 333 - Hoặc, nếu xấu hơn, chấp nhận với xác suất ' $\exp(-\Delta E/T)$ ', với ' ΔE ' [↗](#)
là độ chênh lệch hàm mục tiêu và ' T ' là nhiệt độ.
- 334 4. Giảm nhiệt độ dần theo lịch trình (thường là ' $T = T * \text{cooling_rate}$ '). [↗](#)
- 335 5. Lặp lại cho đến khi nhiệt độ đạt ngưỡng tối thiểu hoặc tìm được giải [↗](#)
pháp đủ tốt.
- 336 - ****Đặc điểm****:
- 337 - ****Hoàn chỉnh****: Không, nhưng có thể tìm giải pháp tốt nếu điều chỉnh [↗](#)
lịch trình nhiệt độ phù hợp.
- 338 - ****Tối ưu****: Không, nhưng có khả năng thoát khỏi cực trị cục bộ, tiến [↗](#)
gần giải pháp toàn cục.

- 339 - ****Độ phức tạp****:
- 340 - Thời gian: Phụ thuộc vào số lần lặp và lịch trình nhiệt độ, thường [↗](#)
cao hơn Hill Climbing.
- 341 - Không gian: $O(1)$, chỉ lưu trạng thái hiện tại và lân cận.
- 342 - ****Ứng dụng****: Phù hợp cho các bài toán tối ưu phức tạp như 8-puzzle, [↗](#)
lập lịch, hoặc tối ưu hóa hàm với nhiều cực trị cục bộ.
- 343
- 344 ##### e. ****Local Beam Search****
- 345 - ****Mô tả****: Duy trì một tập hợp 'k' trạng thái tốt nhất (beam) và mở [↗](#)
rộng chúng, thay vì chỉ làm việc với một trạng thái như Hill Climbing.
- 346 - ****Cách hoạt động****:
- 347 1. Bắt đầu với 'k' trạng thái ban đầu (thường chọn ngẫu nhiên).
- 348 2. Tạo tất cả các trạng thái lân cận từ 'k' trạng thái hiện tại.
- 349 3. Chọn 'k' trạng thái lân cận tốt nhất (dựa trên hàm mục tiêu).
- 350 4. Lặp lại cho đến khi đạt trạng thái mục tiêu hoặc không cải thiện [↗](#)
được thêm.
- 351 - ****Đặc điểm****:
- 352 - ****Hoàn chỉnh****: Không, có thể bỏ sót giải pháp nếu beam không chứa [↗](#)
trạng thái dẫn đến mục tiêu.
- 353 - ****Tối ưu****: Không, nhưng thường tìm được giải pháp tốt hơn Hill [↗](#)
Climbing do khám phá nhiều trạng thái cùng lúc.
- 354 - ****Độ phức tạp****:
- 355 - Thời gian: $O(kb)$ mỗi bước, với b là số nhánh trung bình và k là [↗](#)
kích thước beam.
- 356 - Không gian: $O(k)$, để lưu 'k' trạng thái.
- 357 - ****Ứng dụng****: Dùng trong các bài toán như 8-puzzle, khi cần cân bằng [↗](#)
giữa khám phá nhiều trạng thái và tiết kiệm tài nguyên.
- 358
- 359 ##### f. ****Genetic Algorithm****
- 360 - ****Mô tả****: Dựa trên cơ chế tiến hóa, duy trì một tập hợp các giải pháp [↗](#)
(population) và cải thiện chúng qua các thế hệ bằng cách sử dụng [↗](#)
****crossover****, ****mutation****, và ****selection****.
- 361 - ****Cách hoạt động****:
- 362 1. Khởi tạo một tập hợp các giải pháp ngẫu nhiên (population).
- 363 2. Đánh giá chất lượng mỗi giải pháp bằng hàm mục tiêu (fitness [↗](#)
function).
- 364 3. Chọn các giải pháp tốt (selection) để tạo thế hệ mới thông qua:
- 365 - ****Crossover****: Kết hợp hai giải pháp để tạo giải pháp mới.
- 366 - ****Mutation****: Thay đổi ngẫu nhiên một phần của giải pháp để tăng [↗](#)
tính đa dạng.
- 367 4. Lặp lại qua nhiều thế hệ cho đến khi tìm được giải pháp đủ tốt hoặc [↗](#)
đạt số thế hệ tối đa.
- 368 - ****Đặc điểm****:
- 369 - ****Hoàn chỉnh****: Không, nhưng có thể tìm giải pháp tốt nếu điều chỉnh [↗](#)
tham số hợp lý.
- 370 - ****Tối ưu****: Không, nhưng có khả năng tiến gần giải pháp toàn cục nhờ [↗](#)
tính đa dạng của population.
- 371 - ****Độ phức tạp****:
- 372 - Thời gian: Phụ thuộc vào kích thước population, số thế hệ, và chi [↗](#)

phí đánh giá hàm mục tiêu.

373 - Không gian: $O(p)$, với p là kích thước population.

374 - ****Ứng dụng****: Phù hợp cho các bài toán tối ưu hóa phức tạp như 8-puzzle, thiết kế, hoặc lập lịch, khi không gian trạng thái rất lớn.

375

376 ---

377

378 ### 3. ****So sánh tổng quát****

379	Thuật toán gian	Hoàn chỉnh Tối ưu Độ phức tạp thời gian Độ phức tạp không gian Ứng dụng chính
-----	--------------------	--

380 |-----|-----|-----|----- P

381	**Simple Hill Climbing**	Không	Không	$O(k)$ mỗi bước	↗
-----	---------------------------------	-------	-------	-----------------	-------------------

$O(1)$	Tìm giải pháp nhanh, đơn giản
--------	-------------------------------

382	**Steepest-Ascent Hill Climbing**	Không	Không	$O(k)$ mỗi bước ➡
-----	--	-------	-------	-------------------

| $O(k)$ | Cải thiện giải pháp cục bộ |

383	**Stochastic Hill Climbing**	Không	Không	$O(k)$ mỗi bước	↗
-----	-------------------------------------	-------	-------	-----------------	-------------------

$O(k)$	Tránh cực trị cục bộ nhé
--------	--------------------------

384	**Simulated Annealing**	Không	Không	Phụ thuộc lịch
-----	--------------------------------	-------	-------	----------------

trình	$O(1)$	Thoát cục tri cục bộ, tối ưu hóa
-------	--------	----------------------------------

385	**Local Beam Search**	Không	Không	$O(kb)$ mỗi bước	↗
-----	------------------------------	-------	-------	------------------	-------------------

$O(k)$	Khám phá nhiều trạng thái
--------	---------------------------

386	**Genetic Algorithm**	Không	Không	Phụ thuộc	?
-----	------------------------------	-------	-------	-----------	-------------------

population	$O(p)$	Tối ưu hóa không gian lớn
------------	--------	---------------------------

387

388 ---

389

```
390 ### 4. **Giải pháp tổng quát của Local Search**
```

391 - ****Quy trình chung****:

392 1. Chọn một trạng thái ban đầu (ngẫu nhiên hoặc cố định).

393 2. Xác định hàm mục tiêu (ví dụ: khoảng cách Manhattan trong 8-puzzle) để đánh giá chất lượng trạng thái.

394 3. Tạo và đánh giá các trạng thái lân cận, chọn hoặc chấp nhận trạng
thái tiếp theo dựa trên chiến lược:

395 - ****Simple Hill Climbing****: Chọn trạng thái lân cận đầu tiên tốt hơn.

```
396 - **Steepest-Ascent Hill Climbing**: Chọn trạng thái lân cận tốt nhất.
```

- ****Stochastic Hill Climbing****: Chọn ngẫu nhiên trạng thái lân cận tốt hơn.

```
398 - **Simulated Annealing**: Chấp nhận trạng thái lân cận dựa trên xác suất liên quan đến nhiệt độ.
```

```
399 - **Local Beam Search**: Duy trì và mở rộng 'k' trạng thái tốt nhất.
```

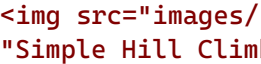
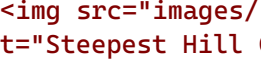
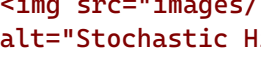
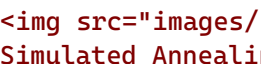
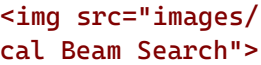
```
400 - **Genetic Algorithm**: Tiến hóa một tập hợp giải pháp qua
      selection, crossover, mutation.
```

4. Lặp lại cho đến khi đạt trạng thái mục tiêu, cực trị cục bộ, hoặc giới hạn tài nguyên (thời gian, số bước).

402 - ****Ưu điểm**:**


403 - Tiết kiệm bộ nhớ, vì chỉ làm việc với trạng thái hiện tại hoặc một

```

    tập nhỏ trạng thái.
404 - Nhanh, đặc biệt khi không cần giải pháp tối ưu toàn cục.
405 - Phù hợp cho không gian trạng thái lớn, như 8-puzzle, khi khám phá
    toàn bộ không khả thi.
406 - Nhược điểm:
407 - Không đảm bảo hoàn chỉnh hoặc tối ưu, dễ bị kẹt ở cực trị cục bộ (trừ
    Simulated Annealing và Genetic Algorithm, có khả năng thoát cục bộ).
408 - Hiệu quả phụ thuộc vào hàm mục tiêu và cách định nghĩa lân cận.
409 - Yêu cầu:
410 - Hàm mục tiêu hiệu quả, phản ánh đúng chất lượng giải pháp.
411 - Cơ chế thoát khỏi cực trị cục bộ (như ngẫu nhiên hóa hoặc lịch trình
    nhiệt độ).
412 - Điều chỉnh tham số (nhiệt độ, kích thước beam, population, v.v.) để
    cân bằng giữa chất lượng và hiệu suất.
413 ---
414 ### 🖼️ Hình ảnh các thuật toán được áp dụng trong trò chơi
415
416 | Thuật Toán | Minh Họa GIF
417 |-----|-----
418 | Simple Hill Climbing | 
419 | Steepest-Ascent Hill Climbing | 
420 | Stochastic Hill Climbing | 
421 | Simulated Annealing | 
422 | Local Beam Search | 
423 | Genetic Algorithm | 
424
425
426 ### 🔍 So sánh các thuật toán tìm kiếm cục bộ (Local Search Algorithms)
427
428 | Thuật Toán | Hoàn chỉnh | Tối ưu | Độ
    phức tạp thời gian | Độ phức tạp không gian | Hiệu suất
    trong 8-puzzle | Ưu điểm | Nhược điểm
429 |-----|-----|-----|
    -----|-----|
    -----|-----|
430 | Simple Hill Climbing | Không | Không | 'O(k)

```

```

    mỗi bước | '0(1)' | Nhanh, nhưng dễ
    kẹt ở cực trị cục bộ, kém hiệu quả khi cách xa mục tiêu.
    | ☒ Nhanh, tiết kiệm bộ nhớ | ☒ Dễ kẹt, không đảm
    bảo tìm được lời giải tốt
431 | **Steepest-Ascent Hill Climbing** | Không | Không | '0(k)
    ' mỗi bước | '0(k)' | Tốt hơn Simple,
    nhưng vẫn dễ kẹt ở cực trị cục bộ.
    | ☒ Chọn lân cận tốt nhất, cải thiện chất lượng | ☒ Tốn thời gian hơn
    Simple, vẫn không đảm bảo tối ưu
432 | **Stochastic Hill Climbing** | Không | Không | '0(k)
    ' mỗi bước | '0(k)' | Nhanh hơn
    Steepest, tránh được một số cực trị cục bộ.
    | ☒ Ngẫu nhiên, nhanh | ☒ Vẫn dễ kẹt,
    không tối ưu
433 | **Simulated Annealing** | Không | Không | Phụ
    thuộc lịch trình | '0(1)' | Có thể thoát
    cực trị cục bộ, hiệu quả với trạng thái xa mục tiêu nếu tham số phù
    hợp. | ☒ Thoát cực trị cục bộ, tiết kiệm bộ nhớ | ☒ Phụ thuộc
    tham số, tốc độ không ổn định
434 | **Local Beam Search** | Không | Không | '0
    (kb)' mỗi bước | '0(k)' | Tốt hơn Hill
    Climbing, nhưng phụ thuộc nhiều vào 'beam_width'.
    | ☒ Khám phá đồng thời nhiều trạng thái | ☒ Dễ bỏ sót lời
    giải nếu beam nhỏ
435 | **Genetic Algorithm** | Không | Không | Phụ
    thuộc population & thể hệ | '0(p)' | Hiệu quả nếu
    điều chỉnh tham số tốt, nhưng không đảm bảo tìm đúng lời giải.
    | ☒ Khám phá không gian lớn, đa dạng lời giải | ☒ Chậm, tốn tài
    nguyên, phụ thuộc nhiều vào tham số
436
437 ### **Chú thích:**
438
439 * 'k': Số trạng thái lân cận (≈ 2-4 trong 8-puzzle, tùy vị trí ô trống).
440 * 'b': Số nhánh trung bình trong không gian trạng thái.
441 * 'p': Kích thước quần thể (*population size*) trong Genetic Algorithm.
442 * **Hàm mục tiêu**: Khoảng cách Manhattan được dùng như một heuristic phổ
    biến, tuy nhiên **không đảm bảo tính hoàn chỉnh/tối ưu trong local
    search**.
443
444 Dựa trên mã nguồn trong file 'solve.py', tôi sẽ phân tích và đưa ra nhận
    xét về hiệu suất của các thuật toán **Local Search** (**Simple Hill
    Climbing**, **Steepest-Ascent Hill Climbing**, **Stochastic Hill
    Climbing**, **Simulated Annealing**, **Local Beam Search**, và
**Genetic Algorithm**) khi áp dụng vào bài toán **Sliding Puzzle 8 ô**
    (8-puzzle). Sau đó, tôi sẽ trình bày bảng so sánh chi tiết để minh họa
    các đặc điểm về hiệu suất, hoàn chỉnh, tối ưu, và độ phức tạp của các
    thuật toán này.
445
446 ###  **Nhân xét chung:**

```


- 447 - ****Simple Hill Climbing****:
- 448 - Nhanh nhất trong nhóm, nhưng dễ bị kẹt ở cực trị cục bộ, đặc biệt trong 8-puzzle do không gian trạng thái phức tạp. ➤
- 449 - Phù hợp khi cần kết quả nhanh với trạng thái ban đầu gần mục tiêu.
- 450 - ****Steepest-Ascent Hill Climbing****:
- 451 - Cải thiện so với Simple Hill Climbing bằng cách chọn lân cận tốt nhất, nhưng vẫn dễ bị kẹt. ➤
- 452 - Trong 8-puzzle, hiệu quả hơn Simple nhưng không phù hợp cho các cấu hình phức tạp. ➤
- 453 - ****Stochastic Hill Climbing****:
- 454 - Tính ngẫu nhiên giúp tránh một số cực trị cục bộ, nhưng vẫn không đảm bảo tìm được mục tiêu trong 8-puzzle. ➤
- 455 - Nhanh hơn Steepest, nhưng hiệu quả phụ thuộc vào sự may mắn trong lựa chọn lân cận. ➤
- 456 - ****Simulated Annealing****:
- 457 - Hiệu quả hơn Hill Climbing trong 8-puzzle nhờ khả năng thoát cực trị cục bộ, đặc biệt khi trạng thái ban đầu xa mục tiêu. ➤
- 458 - Hiệu suất phụ thuộc vào lịch trình nhiệt độ; trong mã, tham số mặc định (cooling_rate=0.99) khá hợp lý nhưng cần thử nghiệm thêm. ➤
- 459 - ****Local Beam Search****:
- 460 - Cải thiện so với Hill Climbing bằng cách duy trì nhiều trạng thái, nhưng hiệu quả phụ thuộc vào 'beam_width'. ➤
- 461 - Trong 8-puzzle, beam_width=3 có thể không đủ lớn để đảm bảo tìm mục tiêu trong không gian trạng thái lớn. ➤
- 462 - ****Genetic Algorithm****:
- 463 - Phù hợp cho không gian trạng thái lớn, nhưng trong 8-puzzle, hiệu suất thấp hơn do chi phí tính toán cao và khó điều chỉnh tham số. ➤
- 464 - Cách biểu diễn chuỗi di chuyển trong mã sáng tạo, nhưng không đảm bảo tìm mục tiêu chính xác. ➤
- 465 ---
- 466 **## Search with Nondeterministic Actions**
- 467
- 468 ---
- 469
- 470 **### 1. **Khái niệm chung về Search with Nondeterministic Actions****
- 471 - ****Search with Nondeterministic Actions**** giải quyết các bài toán trong môi trường mà kết quả của một hành động không xác định (một hành động có thể dẫn đến nhiều trạng thái khác nhau). ➤
- 472 - Thay vì tìm một chuỗi hành động cố định, thuật toán tìm một ****kế hoạch**** (plan) có thể xử lý mọi kết quả có thể xảy ra, thường được biểu diễn dưới dạng cây hoặc đồ thị. ➤
- 473 - ****AND-OR Search Trees**** là một phương pháp chính để giải quyết bài toán này, mô phỏng hai loại nút: ➤
- 474 - ****OR nodes****: Đại diện cho các lựa chọn của tác nhân (agent), nơi tác nhân chọn hành động tốt nhất. ➤
- 475 - ****AND nodes****: Đại diện cho các kết quả không xác định của môi trường, nơi tất cả các kết quả phải được xử lý để đảm bảo kế hoạch thành công. ➤
- 476 - ****Mục tiêu****: Xây dựng một kế hoạch có điều kiện (contingency plan) đảm ➤

bảo đạt được trạng thái mục tiêu bất kể kết quả không xác định nào xảy ra.

477

478 ****Các thành phần chính của AND-OR Search Trees****479 - ****Không gian trạng thái (State Space)****: Tập hợp tất cả các trạng thái có thể có của bài toán (ví dụ: các hoán vị của ô trong 8-puzzle). ➤480 - ****Trạng thái ban đầu (Initial State)****: Điểm xuất phát của bài toán.481 - ****Trạng thái mục tiêu (Goal State)****: Trạng thái cần đạt được.482 - ****Hành động (Actions)****: Các thao tác mà tác nhân có thể thực hiện (ví dụ: di chuyển ô trống lên, xuống, trái, phải trong 8-puzzle). ➤483 - ****Kết quả không xác định (Nondeterministic Outcomes)****: Mỗi hành động có thể dẫn đến nhiều trạng thái khác nhau do môi trường không xác định (ví dụ: một hành động có thể bị ảnh hưởng bởi nhiễu hoặc tác nhân đối thủ). ➤484 - ****Kế hoạch (Plan)****: Một cấu trúc dạng cây hoặc đồ thị, bao gồm:485 - ****OR nodes****: Tác nhân chọn một hành động từ tập hành động khả thi.486 - ****AND nodes****: Môi trường trả về một tập hợp các trạng thái có thể xảy ra, và kế hoạch phải giải quyết tất cả các trạng thái này. ➤487 - ****Hàm đánh giá (Evaluation Function)****: Có thể sử dụng heuristic (như khoảng cách Manhattan trong 8-puzzle) để ưu tiên các nhánh OR có khả năng dẫn đến mục tiêu nhanh hơn. ➤488 - ****Điều kiện dừng****: Đạt trạng thái mục tiêu hoặc xác định không có giải pháp. ➤

489

490 ---

491

492 **### 2. **Giải pháp tổng quát của AND-OR Search Trees****493 - ****Mô tả****:494 - AND-OR Search Trees xây dựng một cây tìm kiếm xen kẽ giữa ****OR nodes**** (lựa chọn hành động của tác nhân) và ****AND nodes**** (các kết quả không xác định của môi trường). ➤495 - Mục tiêu là tìm một ****subtree**** (cây con) mà:

496 - Bắt đầu từ trạng thái ban đầu.

497 - Đảm bảo đạt được trạng thái mục tiêu bất kể kết quả không xác định nào xảy ra. ➤

498 - Kế hoạch kết quả là một ****cây có điều kiện****, trong đó mỗi nhánh AND đại diện cho một kịch bản có thể xảy ra, và mỗi nhánh OR đại diện cho một quyết định của tác nhân. ➤499 - ****Cách hoạt động****:500 1. ****Khởi tạo****: Bắt đầu từ trạng thái ban đầu, tạo một OR node đại diện cho tác nhân. ➤501 2. ****Mở rộng OR node****:

502 - Liệt kê tất cả các hành động khả thi từ trạng thái hiện tại.

503 - Với mỗi hành động, tạo một AND node đại diện cho các kết quả không xác định của hành động đó. ➤

504 3. ****Mở rộng AND node****:

505 - Với mỗi kết quả không xác định, tạo một OR node mới cho trạng thái tương ứng. ➤

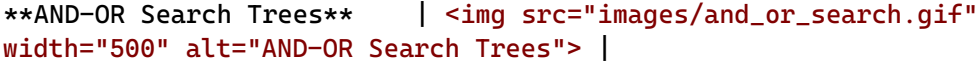
506 - Tiếp tục xen kẽ OR và AND nodes.

```

507 4. **Đánh giá**:
508     - Một OR node thành công nếu ít nhất một nhánh con của nó (qua một hành động) dẫn đến giải pháp.
509     - Một AND node thành công nếu tất cả các nhánh con của nó (tất cả kết quả không xác định) dẫn đến giải pháp.
510 5. **Điều kiện dừng**:
511     - Nếu đạt trạng thái mục tiêu, trả về kế hoạch.
512     - Nếu một OR node không có nhánh nào thành công hoặc một AND node có nhánh thất bại, quay lui (backtrack).
513     - Nếu không tìm được giải pháp, kết luận không có kế hoạch khả thi.
514 - **Đặc điểm**:
515     - **Hoàn chỉnh**: Có, nếu không gian trạng thái hữu hạn và có giải pháp, AND-OR Search sẽ tìm được kế hoạch.
516     - **Tối ưu**: Có thể tối ưu nếu sử dụng heuristic để ưu tiên các hành động tại OR nodes (ví dụ: chọn hành động giảm khoảng cách Manhattan).
517     - **Độ phức tạp**:
518         - **Thời gian**:  $O(b^m)$ , với  $b$  là số nhánh trung bình (số hành động hoặc kết quả không xác định) và  $m$  là độ sâu tối đa của cây. Trong môi trường phức tạp, chi phí có thể rất cao.
519         - **Không gian**:  $O(bm)$  nếu sử dụng tìm kiếm đệ quy, nhưng có thể giảm bằng cách lưu trữ trạng thái đã thăm.
520     - **Ứng dụng**:
521         - Bài toán trong môi trường không xác định, như lập kế hoạch trong robotics, trò chơi với đối thủ (adversarial games), hoặc bài toán như 8-puzzle với nhiễu (ví dụ: ô trống di chuyển ngẫu nhiên).
522         - Xử lý các tình huống cần kế hoạch có điều kiện, đảm bảo thành công bất kể kết quả nào xảy ra.
523 ---
524 ### 3. **So sánh tổng quát**
525 | Thuật toán | Hoàn chỉnh | Tối ưu | Độ phức tạp thời gian | Độ phức tạp không gian | Ứng dụng chính |
526 |-----|-----|-----|-----|-----|-----|
527 | **AND-OR Search Trees** | Có (nếu hữu hạn) | Có (nếu dùng heuristic) |  $O(b^m)$  |  $O(bm)$  | Lập kế hoạch trong môi trường không xác định (robotics, trò chơi, 8-puzzle với nhiễu) |
528 ---
529 ### 4. **Cấu trúc của AND-OR Search Tree**
530 - **OR nodes**:
531     - Đại diện cho trạng thái mà tác nhân phải chọn hành động.
532     - Thành công nếu ít nhất một hành động dẫn đến giải pháp.
533     - Ví dụ: Trong 8-puzzle, tác nhân chọn di chuyển ô trống lên, xuống, trái, hoặc phải.
534 - **AND nodes**:
535     - Đại diện cho các kết quả không xác định của một hành động.
536     - Thành công nếu tất cả các kết quả đều dẫn đến giải pháp.
537     - Ví dụ: Nếu môi trường có nhiễu, di chuyển "lên" có thể dẫn đến nhiều trạng thái khác nhau.
538 - **Kế hoạch kết quả**:

```

```

539 - Một cây với các nhánh OR (lựa chọn hành động) và AND (xử lý tất cả
    kết quả).
540 - Ví dụ trong 8-puzzle: "Nếu trạng thái là S1, di chuyển lên; nếu kết
    quả là S2, di chuyển phải; nếu kết quả là S3, di chuyển xuống."
541
542 - **Ưu điểm**:
543 - Xử lý tốt các môi trường không xác định, đảm bảo kế hoạch khả thi cho
    mọi kịch bản.
544 - Linh hoạt, có thể kết hợp với heuristic để cải thiện hiệu suất.
545 - Hoàn chỉnh trong không gian trạng thái hữu hạn.
546 - **Nhược điểm**:
547 - Độ phức tạp cao trong môi trường có nhiều kết quả không xác định.
548 - Yêu cầu bộ nhớ lớn nếu không gian trạng thái phức tạp, trừ khi sử
    dụng kỹ thuật tối ưu như lưu trữ trạng thái đã thăm.
549 - Cần xác định rõ các kết quả không xác định của mỗi hành động, có thể
    khó trong một số bài toán thực tế.
550 ---
551 ### 📷 **Hình ảnh các thuật toán được áp dụng trong trò chơi**
552
553 | **Thuật Toán** | **Minh Họa GIF**
554 |-----|-----
555 | **AND-OR Search Trees** | 
556
557 ### 🔍 So sánh thuật toán tìm kiếm với hành động không xác định (Search
    with Nondeterministic Actions)
558
559 | **Thuật toán** | **Hoàn chỉnh** | **Tối ưu** | **Độ phức tạp
    thời gian** | **Độ phức tạp không gian** | **Hiệu suất trong 8-puzzle**
    | **Ưu điểm** | **Nhược điểm** |
560 |-----|-----|-----|-----
561 | **AND-OR Search Tree** | Có (nếu hữu hạn) | Có (nếu dùng heuristic) | 0
    (b^m) | O(bm) | Hiệu quả khi xử
    lý môi trường không xác định, nhưng chậm và tốn tài nguyên nếu số kết
    quả không xác định lớn. | Xử lý không xác định, hoàn chỉnh, có thể tối
    ưu. | Độ phức tạp cao, tốn bộ nhớ, phụ thuộc vào mô hình không xác
    định. |
562
563 **Ghi chú**:
564 - **b**: Số nhánh trung bình, phụ thuộc vào số hành động và số kết quả
    không xác định mỗi hành động (trong 8-puzzle, b có thể từ 2-4 cho hành
    động và tăng thêm do nhiễu).
565 - **m**: Độ sâu tối đa của cây tìm kiếm.
566 - **Heuristic**: Khoảng cách Manhattan được sử dụng trong mã, là
    admissible và giúp ưu tiên các nhánh OR hiệu quả.

```

- 567 - ****Môi trường không xác định****: Trong `'solve.py'`, giả định rằng mỗi hành động có thể dẫn đến một tập hợp trạng thái (AND nodes), ví dụ: do nhiều hoặc đối thủ.
- 568
- 569 ---
- 570 **### 📖 **Nhận xét chung****
- 571 - AND-OR Search Tree là lựa chọn phù hợp khi bài toán 8-puzzle được mở rộng để bao gồm yếu tố không xác định, như nhiều môi trường hoặc hành động của đối thủ làm thay đổi trạng thái.
- 572 - Trong mã, việc sử dụng khoảng cách Manhattan làm heuristic giúp thuật toán ưu tiên các hành động đưa trạng thái gần mục tiêu, cải thiện hiệu suất so với tìm kiếm không định hướng.
- 573 - Tuy nhiên, thuật toán này không hiệu quả bằng các thuật toán xác định như A* hoặc IDA* trong 8-puzzle thông thường, vì nó phải xử lý nhiều kết quả không xác định, làm tăng chi phí tính toán.
- 574 ---
- 575 **## Searching with no observation và Searching in partially observable environments**
- 576 ---
- 577 **### 1. **Searching with No Observation (Tìm kiếm không quan sát)****
- 578
- 579 **#### **Khái niệm chung****
- 580 - ****Searching with No Observation**** áp dụng cho các bài toán trong môi trường mà tác nhân (agent) không nhận được thông tin về trạng thái hiện tại sau khi thực hiện hành động (không có quan sát hoặc cảm biến).
- 581 - Tác nhân chỉ biết trạng thái ban đầu, tập hợp hành động, và mô hình chuyển đổi trạng thái (transition model), nhưng không thể quan sát trạng thái sau mỗi bước.
- 582 - Mục tiêu là xây dựng một ****kế hoạch hành động mở**** (open-loop plan), tức là một chuỗi hành động cố định để đạt mục tiêu bất kể trạng thái thực tế.
- 583 - Thường áp dụng trong môi trường xác định hoặc không xác định, nhưng không có thông tin phản hồi.
- 584
- 585 **#### **Các thành phần chính****
- 586 - ****Không gian trạng thái (State Space)****: Tập hợp tất cả các trạng thái có thể có (ví dụ: các hoán vị trong 8-puzzle).
- 587 - ****Trạng thái ban đầu (Initial State)****: Điểm xuất phát, giả định tác nhân biết trạng thái này.
- 588 - ****Trạng thái mục tiêu (Goal State)****: Trạng thái cần đạt được.
- 589 - ****Hành động (Actions)****: Các thao tác khả thi (ví dụ: di chuyển ô trống lên, xuống, trái, phải trong 8-puzzle).
- 590 - ****Mô hình chuyển đổi trạng thái (Transition Model)****: Quy tắc xác định trạng thái tiếp theo sau một hành động (có thể xác định hoặc không xác định).
- 591 - ****Tập niềm tin (Belief State)****: Vì không có quan sát, tác nhân duy trì một tập hợp các trạng thái có thể có (belief state) dựa trên trạng thái ban đầu và lịch sử hành động.
- 592 - ****Kế hoạch (Plan)****: Một chuỗi hành động cố định hoặc một chính sách

(policy) đảm bảo đạt mục tiêu từ trạng thái ban đầu.

593

594 ##### ****Giải pháp tổng quát****

595 - ****Mô tả****:

596 - Tác nhân xây dựng một kế hoạch dựa trên mô hình chuyển đổi trạng thái, giả định rằng không có thông tin mới thu thập được trong quá trình thực hiện. ➤

597 - Trong môi trường xác định, kế hoạch là một chuỗi hành động cố định.

598 - Trong môi trường không xác định, kế hoạch phải xem xét tất cả các trạng thái có thể có trong tập niềm tin (belief state). ➤

599 - ****Cách hoạt động****:

600 1. ****Khởi tạo****: Bắt đầu từ trạng thái ban đầu hoặc tập niềm tin ban đầu (chỉ chứa trạng thái ban đầu). ➤

601 2. ****Dự đoán trạng thái****: Dựa trên mô hình chuyển đổi, tính toán tập niềm tin mới sau mỗi hành động (bao gồm tất cả trạng thái có thể xảy ra). ➤

602 3. ****Lập kế hoạch****:

603 - Chọn chuỗi hành động dẫn tập niềm tin đến một trạng thái chứa mục tiêu. ➤

604 - Trong môi trường không xác định, sử dụng kỹ thuật như ****Belief-State Search**** (tìm kiếm trong không gian tập niềm tin) để đảm bảo tất cả trạng thái trong tập niềm tin đều đạt mục tiêu. ➤

605 4. ****Thực thi****: Thực hiện chuỗi hành động mà không cần quan sát, hy vọng đạt mục tiêu. ➤

606 - ****Đặc điểm****:

607 - ****Hoàn chỉnh****: Có, nếu không gian trạng thái hữu hạn và tồn tại kế hoạch khả thi. ➤

608 - ****Tối ưu****: Có thể tối ưu nếu sử dụng hàm chi phí (ví dụ: số bước tối thiểu), nhưng khó trong môi trường không xác định. ➤

609 - ****Độ phức tạp****:

610 - ****Thời gian****: $O(|B|^d)$, với $|B|$ là kích thước tập niềm tin và d là độ sâu kế hoạch. ➤

611 - ****Không gian****: $O(|B|)$, để lưu trữ tập niềm tin.

612 - ****Ứng dụng****:

613 - Robotics trong môi trường không cảm biến (ví dụ: robot di chuyển trong bóng tối). ➤

614 - 8-puzzle với giả định không quan sát trạng thái sau mỗi di chuyển (tác nhân chỉ biết trạng thái ban đầu và thực hiện chuỗi hành động cố định). ➤

615 - ****Ví dụ trong 8-puzzle****:

616 - Tác nhân biết trạng thái ban đầu (ví dụ: `[2, 6, 5, 0, 8, 7, 4, 3, 1]`). ➤

617 - Không quan sát trạng thái sau mỗi di chuyển, chỉ thực hiện chuỗi hành động cố định (ví dụ: "lên, trái, xuống"). ➤

618 - Kế hoạch phải đảm bảo trạng thái mục tiêu (`[1, 2, 3, 4, 5, 6, 7, 8, 0]`) nằm trong tập niềm tin cuối cùng. ➤

619


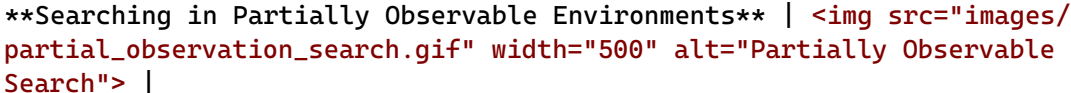
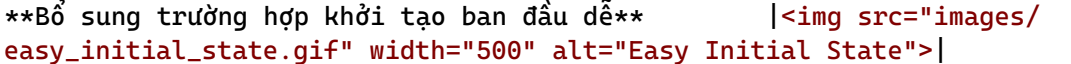
620 ##### ****Ưu và nhược điểm****

621 - ****Ưu điểm****:

- 622 - Đơn giản trong môi trường xác định, vì chỉ cần một chuỗi hành động cố định. ➤
- 623 - Có thể xử lý môi trường không xác định bằng cách duy trì tập niềm tin. ➤
- 624 - ****Nhược điểm****:
- 625 - Không hiệu quả nếu tập niềm tin lớn (trong 8-puzzle, tập niềm tin có thể lên đến $9!/2$ trạng thái). ➤
- 626 - Không tận dụng được thông tin mới, dẫn đến kế hoạch bảo thủ (overly cautious). ➤
- 627 - Khó tối ưu trong môi trường không xác định do phải xử lý tất cả trạng thái có thể. ➤
- 628
- 629 ---
- 630
- 631 **### 2. *Searching in Partially Observable Environments* (Tìm kiếm trong môi trường quan sát một phần)**** ➤
- 632
- 633 **#### ***Khái niệm chung*****
- 634 - ***Searching in Partially Observable Environments*** áp dụng cho các bài toán mà tác nhân nhận được một số thông tin quan sát (observation) sau mỗi hành động, nhưng không đủ để xác định trạng thái chính xác. ➤
- 635 - Tác nhân phải duy trì một ***tập niềm tin*** (belief state) dựa trên trạng thái ban đầu, lịch sử hành động, và các quan sát. ➤
- 636 - Mục tiêu là xây dựng một ***kế hoạch có điều kiện*** (contingency plan) hoặc chính sách (policy) để đạt mục tiêu, sử dụng thông tin quan sát để điều chỉnh hành động. ➤
- 637 - Thường được mô hình hóa dưới dạng ***Partially Observable Markov Decision Process (POMDP)***. ➤
- 638
- 639 **#### ***Các thành phần chính*****
- 640 - ***Không gian trạng thái (State Space)***: Tất cả các trạng thái có thể có. ➤
- 641 - ***Trạng thái ban đầu (Initial State)***: Một trạng thái hoặc tập niềm tin ban đầu. ➤
- 642 - ***Trạng thái mục tiêu (Goal State)***: Trạng thái cần đạt được.
- 643 - ***Hành động (Actions)***: Các thao tác khả thi.
- 644 - ***Mô hình chuyển đổi trạng thái (Transition Model)***: Xác suất hoặc quy tắc chuyển đổi giữa các trạng thái sau hành động. ➤
- 645 - ***Quan sát (Observations)***: Thông tin mà tác nhân nhận được sau mỗi hành động (ví dụ: vị trí của một số ô trong 8-puzzle). ➤
- 646 - ***Mô hình quan sát (Observation Model)***: Liên kết trạng thái với các quan sát có thể (ví dụ: xác suất nhận được quan sát 0 trong trạng thái S). ➤
- 647 - ***Tập niềm tin (Belief State)***: Một phân phối xác suất hoặc tập hợp các trạng thái có thể, cập nhật dựa trên hành động và quan sát. ➤
- 648 - ***Kế hoạch (Plan)***: Một chính sách hoặc cây có điều kiện, ánh xạ tập niềm tin đến hành động hoặc chuỗi hành động. ➤
- 649
- 650 **#### ***Giải pháp tổng quát*****

- 651 - ****Mô tả**:**
- 652 - Tác nhân duy trì một tập niềm tin và cập nhật nó sau mỗi hành động và quan sát, sử dụng ****lọc Bayes**** (Bayesian filtering) hoặc các phương pháp tương tự. ➤
- 653 - Kế hoạch là một chính sách (policy) ánh xạ từ tập niềm tin đến hành động, hoặc một cây có điều kiện dựa trên các quan sát nhận được. ➤
- 654 - Thuật toán thường sử dụng ****Belief-State Search**** hoặc các kỹ thuật như POMDP để tìm kế hoạch tối ưu. ➤
- 655 - ****Cách hoạt động**:**
- 656 1. ****Khởi tạo**:** Bắt đầu với tập niềm tin ban đầu (có thể là một trạng thái hoặc phân phối xác suất). ➤
- 657 2. ****Cập nhật tập niềm tin**:**
- 658 - Sau mỗi hành động, dự đoán tập niềm tin mới dựa trên mô hình chuyển đổi. ➤
- 659 - Sau mỗi quan sát, cập nhật tập niềm tin bằng cách loại bỏ các trạng thái không phù hợp (hoặc điều chỉnh xác suất trong POMDP). ➤
- 660 3. ****Lập kế hoạch**:**
- 661 - Tìm kiếm trong không gian tập niềm tin, sử dụng heuristic (như khoảng cách Manhattan trung bình trong tập niềm tin) để ưu tiên hành động. ➤
- 662 - Xây dựng cây có điều kiện: "Thực hiện hành động A; nếu nhận quan sát O1, làm X; nếu nhận O2, làm Y." ➤
- 663 4. ****Thực thi**:**
- 664 - Thực hiện hành động, nhận quan sát, cập nhật tập niềm tin, và lặp lại cho đến khi tập niềm tin chỉ chứa trạng thái mục tiêu. ➤
- 665 - ****Đặc điểm**:**
- 666 - ****Hoàn chỉnh**:** Có, nếu không gian trạng thái và quan sát hữu hạn, và tồn tại kế hoạch khả thi. ➤
- 667 - ****Tối ưu**:** Có thể tối ưu nếu sử dụng hàm chi phí và giải POMDP chính xác, nhưng thường phải xấp xỉ do độ phức tạp cao. ➤
- 668 - ****Độ phức tạp**:**
- 669 - ****Thời gian**:** $O(|B|^d)$, với $|B|$ là số tập niềm tin có thể (có thể rất lớn, thậm chí vô hạn nếu tập niềm tin là phân phối liên tục). ➤
- 670 - ****Không gian**:** $O(|B|)$, để lưu trữ tập niềm tin và cây kế hoạch. ➤
- 671 - ****Ứng dụng**:**
- 672 - Robotics với cảm biến hạn chế (ví dụ: robot định vị với GPS không chính xác). ➤
- 673 - Trò chơi với thông tin không đầy đủ. ➤
- 674 - 8-puzzle với quan sát một phần (ví dụ: chỉ thấy vị trí của một số ô sau mỗi di chuyển). ➤
- 675 - ****Ví dụ trong 8-puzzle**:**
- 676 - Tác nhân chỉ thấy vị trí của ô trống hoặc một số ô sau mỗi di chuyển. ➤
- 677 - Duy trì tập niềm tin về các trạng thái có thể, cập nhật dựa trên quan sát (ví dụ: "ô trống ở vị trí (2,2)"). ➤
- 678 - Xây dựng kế hoạch: "Di chuyển lên; nếu ô trống ở (1,2), di chuyển trái; nếu ở (2,1), di chuyển xuống." ➤
- 679
- 680 ##### ****Ưu và nhược điểm****
- 681 - ****Ưu điểm**:**


```

682 - Tận dụng thông tin quan sát để thu hẹp tập niềm tin, hiệu quả hơn tìm kiếm không quan sát.
683 - Linh hoạt, có thể xử lý môi trường xác định hoặc không xác định.
684 - **Nhược điểm**:
685 - Độ phức tạp cao, đặc biệt khi không gian tập niềm tin lớn hoặc quan sát phức tạp.
686 - Yêu cầu mô hình quan sát và chuyển đổi chính xác, khó triển khai trong thực tế.
687 - Giải POMDP chính xác thường không khả thi, cần xấp xỉ.
688
689 ---
690
691 ### 3. **So sánh tổng quát**
692 | Nhóm thuật toán | Hoàn chỉnh | Tối ưu | Độ phức tạp thời gian | Độ phức tạp không gian | Ứng dụng chính |
693 |-----|-----|-----|-----|-----|
694 | **Searching with No Observation** | Có (nếu hữu hạn) | Có (trong môi trường xác định) |  $O(|B|^d)$  |  $O(|B|)$  | Robotics không cảm biến, 8-puzzle không quan sát |
695 | **Searching in Partially Observable Environments** | Có (nếu hữu hạn) | Có (nếu giải POMDP) |  $O(|B|^d)$  |  $O(|B|)$  | Robotics với cảm biến hạn chế, trò chơi, 8-puzzle với quan sát một phần |
696
697 **Ghi chú**:
698 - **|B|**: Kích thước không gian tập niềm tin, có thể rất lớn trong môi trường phức tạp.
699 - **d**: Độ sâu kế hoạch hoặc số bước cần thiết để đạt mục tiêu.
700
701 ### 🖼️ **Hình ảnh các thuật toán được áp dụng trong trò chơi**
702
703 | **Thuật Toán / Phương pháp** | **Minh Họa GIF** |
704 |-----|-----|
705 | **Searching with No Observation** |  |
706 | **Searching in Partially Observable Environments** |  |
707 |-----|-----|
708 | **Bổ sung trường hợp khởi tạo ban đầu dễ** |  |
709
710 ### 🔍 So sánh các thuật toán tìm kiếm với môi trường không quan sát (Searching with No Observation) và tìm kiếm với môi trường không quan sát một phần (Searching in Partially Observable Environments)

```



```

711
712 | **Thuật toán** | **Hoàn chỉnh** | **Tối ưu** | ↗
    | **Độ phức tạp thời gian** | **Độ phức tạp không gian** | **Hiệu suất ↗
    | trong 8-puzzle** | **Ưu điểm** | **Nhược điểm** |
713 | -----|-----|-----|-----| ↗
    | -----|-----|-----|-----| ↗
    | -----|-----|-----|-----|
714 | **Searching with No Observation** | Có (nếu hữu hạn) | Không | ↗
    |  $O(|B|^d)$  |  $O(|B|)$  | Hiệu quả | ↗
    | thấp, phù hợp khi không có cảm biến, nhưng tốn tài nguyên nếu tập niềm | ↗
    | tin lớn. | Đơn giản, xử lý môi trường không quan sát. | Tập niềm tin | ↗
    | lớn, không tối ưu, không tận dụng thông tin. |
715 | **Searching in Partially Observable Environments** | Có (nếu hữu hạn) | ↗
    | Không |  $O(|B|^d)$  |  $O(|B|)$  | ↗
    | Hiệu quả hơn No Observation, phụ thuộc vào chất lượng quan sát. Tốt khi | ↗
    | quan sát mạnh. | Tận dụng quan sát, linh hoạt. | Độ phức tạp cao, phụ | ↗
    | thuộc mô hình quan sát, không tối ưu. |
716
717 **Ghi chú**:
718 - **|B|**: Kích thước không gian tập niềm tin, có thể lên đến  $9!/2$  ( $\approx$  | ↗
    | 181,440) trong 8-puzzle nếu không có hoặc ít quan sát.
719 - **d**: Độ sâu kế hoạch hoặc số bước cần thiết để đạt mục tiêu.
720 - **Heuristic**: Khoảng cách Manhattan được sử dụng trong mã, giúp ưu | ↗
    | tiên hành động nhưng không đảm bảo tối ưu trong môi trường không xác | ↗
    | định.
721 ---
722 ### 📝 **Nhận xét chung**
723 - **Searching with No Observation**:
724 - Phù hợp cho các kịch bản 8-puzzle không có cảm biến, nhưng hiệu suất | ↗
    | thấp do tập niềm tin có thể mở rộng nhanh chóng (đặc biệt trong môi | ↗
    | trường không xác định).
725 - Trong mã, việc sử dụng khoảng cách Manhattan làm heuristic giúp giảm | ↗
    | số hành động cần xem xét, nhưng vẫn không thể cạnh tranh với các | ↗
    | thuật toán như A* trong môi trường xác định.
726 - Chỉ thực sự hữu ích khi mô hình chuyển đổi đơn giản và số trạng thái | ↗
    | trong tập niềm tin được kiểm soát.
727 - **Searching in Partially Observable Environments**:
728 - Hiệu quả hơn Searching with No Observation nhờ tận dụng quan sát để | ↗
    | thu hẹp tập niềm tin.
729 - Trong 8-puzzle, hiệu suất phụ thuộc vào chất lượng quan sát. Nếu quan | ↗
    | sát mạnh (ví dụ: biết vị trí ô trống và một số ô), thuật toán có thể | ↗
    | gần với hiệu suất của A*. Nếu quan sát yếu, tập niềm tin vẫn lớn, | ↗
    | dẫn đến chi phí tính toán cao.
730 - Trong mã, việc cập nhật tập niềm tin và xây dựng kế hoạch có điều | ↗
    | kiện là phù hợp, nhưng yêu cầu mô hình quan sát chính xác.
731 - **Tình huống phù hợp**:
732 - **No Observation**: Hữu ích khi 8-puzzle được mô hình hóa không có | ↗
    | cảm biến (ví dụ: tác nhân chỉ biết trạng thái ban đầu và thực hiện | ↗
    | chuỗi di chuyển cố định).

```

733 - **Partially Observable**: Phù hợp khi có quan sát một phần (ví dụ: biết vị trí ô trống), đặc biệt trong các kịch bản thực tế như robotics hoặc trò chơi với thông tin hạn chế.

734 ---

735 **Constraint Satisfaction Problems**

736 ---

737 **1. Khái niệm chung về Constraint Satisfaction Problems (CSPs)**

738 - **Constraint Satisfaction Problems (CSPs)** là một cách biểu diễn bài toán tìm kiếm, trong đó mục tiêu là gán giá trị cho các biến sao cho thỏa mãn một tập hợp các ràng buộc (constraints).

739 - CSPs thường được sử dụng trong các bài toán có cấu trúc ràng buộc rõ ràng, như lập lịch, tô màu bản đồ, hoặc giải câu đố logic.

740 - Thay vì tìm kiếm trực tiếp trong không gian trạng thái, CSPs biểu diễn bài toán dưới dạng **biến**, **miền giá trị**, và **ràng buộc**, sau đó sử dụng các kỹ thuật như AC-3 và Backtracking để tìm giải pháp.

741

742 ---

743 **2. Các thành phần chính của CSPs**

744 - **Biến (Variables)**: Các đối tượng cần gán giá trị (ví dụ: trong 8-puzzle, mỗi ô có thể được xem là một biến đại diện cho giá trị tại vị trí đó).

745 - **Miền giá trị (Domains)**: Tập hợp các giá trị khả thi cho mỗi biến (ví dụ: trong 8-puzzle, miền giá trị là $\{0, 1, 2, \dots, 8\}$, với 0 là ô trống).

746 - **Ràng buộc (Constraints)**: Các điều kiện phải thỏa mãn giữa các biến, có thể là:

747 - **Ràng buộc đơn (Unary Constraints)**: Liên quan đến một biến (ví dụ: ô ở vị trí (1,1) không thể là 0).

748 - **Ràng buộc đôi (Binary Constraints)**: Liên quan đến hai biến (ví dụ: hai ô không thể có cùng giá trị).

749 - **Ràng buộc toàn cục (Global Constraints)**: Liên quan đến nhiều biến (ví dụ: tất cả các ô phải tạo thành một hoán vị hợp lệ).

750 - **Trạng thái mục tiêu (Solution)**: Một gán giá trị đầy đủ (assignment) cho tất cả các biến, thỏa mãn tất cả các ràng buộc.

751 - **Không gian trạng thái**: Tập hợp tất cả các gán giá trị có thể cho các biến, giới hạn bởi miền giá trị và ràng buộc.

752

753 ---

754

755 **3. Giải pháp tổng quát của CSPs**

756

757 **a. AC-3 (Arc Consistency Algorithm)**

758 - **Mô tả**:

759 - AC-3 là một thuật toán tiền xử lý dùng để giảm miền giá trị của các biến bằng cách đảm bảo **tính nhất quán cung** (arc consistency).

760 - Một cung (arc) giữa hai biến X_i và X_j là nhất quán nếu với mỗi giá trị trong miền của X_i , tồn tại ít nhất một giá trị trong miền của X_j thỏa mãn ràng buộc giữa chúng.

761 - AC-3 loại bỏ các giá trị không thỏa mãn ràng buộc, thu hẹp miền giá

- trị để giảm không gian tìm kiếm trước khi áp dụng thuật toán tìm kiếm chính (như Backtracking).
- 762 - **Cách hoạt động**:
 - 763 1. **Khởi tạo**: Tạo một hàng đợi chứa tất cả các cung tương ứng với các ràng buộc đôi trong CSP.
 - 764 2. **Xử lý cung**:
 - 765 - Lấy một cung $\backslash(X_i, X_j)\backslash$ từ hàng đợi.
 - 766 - Kiểm tra tính nhất quán của cung: Với mỗi giá trị trong miền của $\backslash(X_i)\backslash$, đảm bảo tồn tại giá trị trong miền của $\backslash(X_j)\backslash$ thỏa mãn ràng buộc.
 - 767 - Nếu một giá trị trong miền của $\backslash(X_i)\backslash$ không thỏa mãn, loại bỏ giá trị đó.
 - 768 3. **Cập nhật hàng đợi**:
 - 769 - Nếu miền của $\backslash(X_i)\backslash$ bị thay đổi, thêm tất cả các cung liên quan đến $\backslash(X_i)\backslash$ (như $\backslash(X_k, X_i)\backslash$) vào hàng đợi để kiểm tra lại.
 - 770 4. **Kết thúc**:
 - 771 - Tiếp tục cho đến khi hàng đợi rỗng (miền đã nhất quán) hoặc một miền trở nên rỗng (không có giải pháp).
 - 772 - **Đặc điểm**:
 - 773 - **Hoàn chỉnh**: Không, AC-3 chỉ là tiền xử lý, không đảm bảo tìm giải pháp mà chỉ giảm kích thước miền.
 - 774 - **Tối ưu**: Không liên quan, vì AC-3 không tìm giải pháp mà chỉ tối ưu không gian tìm kiếm.
 - 775 - **Độ phức tạp**:
 - 776 - **Thời gian**: $O(e * d^3)$ trong trường hợp xấu nhất, với e là số cung và d là kích thước miền lớn nhất.
 - 777 - **Không gian**: $O(e)$, để lưu hàng đợi các cung.
 - 778 - **Ứng dụng**:
 - 779 - Tiền xử lý cho các bài toán CSP như 8-puzzle, tô màu bản đồ, hoặc lập lịch.
 - 780 - Trong 8-puzzle, AC-3 có thể đảm bảo rằng các ô lân cận có giá trị phù hợp với các ràng buộc về hoán vị.
 - 781
 - 782 ##### **b. Backtracking Search**
 - 783 - **Mô tả**:
 - 784 - Backtracking Search là một thuật toán tìm kiếm đệ quy, gán giá trị cho các biến một cách tuần tự và quay lui (backtrack) khi gặp gán không thỏa mãn ràng buộc.
 - 785 - Thường được cải tiến với các kỹ thuật như chọn biến thông minh (most constrained variable), chọn giá trị tối ưu (least constraining value), và kiểm tra ràng buộc sớm (forward checking).
 - 786 - **Cách hoạt động**:
 - 787 1. **Khởi tạo**: Bắt đầu với một gán rỗng (không biến nào được gán giá trị).
 - 788 2. **Chọn biến**: Chọn một biến chưa được gán (có thể dùng tiêu chí như biến có miền nhỏ nhất).
 - 789 3. **Gán giá trị**: Thử từng giá trị trong miền của biến, kiểm tra xem gán này có thỏa mãn tất cả ràng buộc liên quan không.
 - 790 4. **Đệ quy**:

- 791 - Nếu gán hợp lệ, chuyển sang biến tiếp theo và lặp lại.
- 792 - Nếu gán không hợp lệ hoặc không dẫn đến giải pháp, quay lui để thử giá trị khác. ➤
- 793 5. ****Kết thúc****:
- 794 - Trả về gán đầy đủ thỏa mãn tất cả ràng buộc hoặc kết luận không có giải pháp. ➤
- 795 - ****Đặc điểm****:
- 796 - ****Hoàn chỉnh****: Có, nếu không gian trạng thái hữu hạn, Backtracking sẽ tìm được giải pháp hoặc xác định không có giải pháp. ➤
- 797 - ****Tối ưu****: Có thể tối ưu nếu sử dụng tiêu chí chọn giá trị dựa trên chi phí. ➤
- 798 - ****Độ phức tạp****:
- 799 - ****Thời gian****: $O(d^n)$ trong trường hợp xấu nhất, với n là số biến và d là kích thước miền lớn nhất. ➤
- 800 - ****Không gian****: $O(n)$, để lưu trạng thái gán hiện tại.
- 801 - ****Ứng dụng****:
- 802 - Giải các bài toán CSP như 8-puzzle, Sudoku, hoặc lập lịch.
- 803 - Trong 8-puzzle, Backtracking có thể gán giá trị cho các ô hoặc chuỗi di chuyển để đạt trạng thái mục tiêu. ➤
- 804
- 805 ##### ****c. Generate and Test****
- 806 - ****Mô tả****:
- 807 - Generate and Test là một phương pháp đơn giản để giải bài toán CSP bằng cách tạo ra tất cả các gán giá trị có thể cho các biến (các tổ hợp giá trị trong miền) và kiểm tra từng gán để tìm giải pháp thỏa mãn tất cả ràng buộc. ➤
- 808 - Đây là một cách tiếp cận "thô" (brute-force), không hiệu quả cho các bài toán có không gian tìm kiếm lớn, nhưng dễ triển khai và đảm bảo tìm được giải pháp nếu tồn tại. ➤
- 809 - ****Cách hoạt động****:
- 810 1. ****Khởi tạo****:
- 811 - Xác định danh sách các biến và miền giá trị của chúng.
- 812 - Tạo một tập hợp tất cả các tổ hợp gán giá trị có thể (sản phẩm Descartes của các miền). ➤
- 813 2. ****Tạo gán****:
- 814 - Lần lượt tạo từng tổ hợp gán giá trị cho tất cả các biến.
- 815 - Ví dụ: Nếu có $\backslash(n\backslash)$ biến, mỗi biến có miền kích thước $\backslash(d\backslash)$, tạo ra $\backslash(d^n\backslash)$ gán. ➤
- 816 3. ****Kiểm tra ràng buộc****:
- 817 - Với mỗi gán, kiểm tra xem nó có thỏa mãn tất cả các ràng buộc của CSP không. ➤
- 818 - Nếu thỏa mãn, lưu gán này là một giải pháp.
- 819 4. ****Kết thúc****:
- 820 - Trả về giải pháp đầu tiên tìm thấy, tất cả các giải pháp, hoặc kết luận không có giải pháp nếu không gán nào thỏa mãn. ➤
- 821 - ****Đặc điểm****:
- 822 - ****Hoàn chỉnh****: Có, Generate and Test sẽ kiểm tra toàn bộ không gian tìm kiếm, đảm bảo tìm được giải pháp nếu tồn tại. ➤
- 823 - ****Tối ưu****: Không, vì nó không ưu tiên các gán có khả năng thỏa mãn ➤

cao hơn, chỉ kiểm tra lần lượt.

824 - **Độ phức tạp**:

825 - **Thời gian**: $O(d^n \cdot c)$, với $\backslash(n\backslash)$ là số biến, $\backslash(d\backslash)$ là kích thước miền lớn nhất, và $\backslash(c\backslash)$ là chi phí kiểm tra ràng buộc cho mỗi gán. [↗](#)

826 - **Không gian**: $O(n)$, để lưu gán hiện tại, hoặc $O(d^n)$ nếu lưu tất cả các gán. [↗](#)

827 - **Hiệu quả**: Rất thấp, đặc biệt khi không gian tìm kiếm lớn, vì nó không sử dụng bất kỳ kỹ thuật tối ưu nào. [↗](#)

828 - **Ứng dụng**:

829 - Phù hợp cho các bài toán CSP nhỏ, nơi không gian tìm kiếm không quá lớn (ví dụ: CSP với ít biến hoặc miền nhỏ). [↗](#)

830 - Trong 8-puzzle, Generate and Test có thể tạo tất cả các hoán vị của lưới 3x3 ($9! = 362,880$ trạng thái) và kiểm tra xem có trạng thái nào khớp với 'goal_state' không, nhưng không thực tế do độ phức tạp cao. [↗](#)

831 - Thường được cải tiến bằng cách kết hợp với các kỹ thuật như AC-3 hoặc Backtracking để giảm số gán cần kiểm tra. [↗](#)

832

833 ---

834

835 **### 4. Giải pháp tổng quát của CSPs**

836 - **Quy trình chung**:

837 1. **Biểu diễn bài toán**:

838 - Xác định các biến, miền giá trị, và ràng buộc.

839 - Ví dụ trong 8-puzzle: 9 biến (mỗi ô), miền giá trị $\{0, 1, \dots, 8\}$, ràng buộc là các ô phải tạo thành hoán vị hợp lệ, thỏa mãn cấu trúc lưới và khả năng di chuyển ô trống. [↗](#)

840 2. **Tiền xử lý với AC-3**:

841 - Áp dụng AC-3 để thu hẹp miền giá trị, loại bỏ các giá trị không thỏa mãn ràng buộc đôi. [↗](#)

842 - Giảm kích thước không gian tìm kiếm trước khi chạy các thuật toán tìm kiếm như Backtracking hoặc Generate and Test. [↗](#)

843 3. **Tìm kiếm giải pháp**:

844 - **Generate and Test**:

845 - Tạo tất cả các tổ hợp gán giá trị cho các biến và kiểm tra từng gán. [↗](#)

846 - Thích hợp cho các bài toán nhỏ, nhưng không hiệu quả cho 8-puzzle do không gian tìm kiếm lớn. [↗](#)

847 - **Backtracking Search**:

848 - Gán giá trị cho các biến một cách tuần tự, kiểm tra ràng buộc, và quay lui khi cần. [↗](#)

849 - Sử dụng các kỹ thuật tối ưu:

850 - **Most Constrained Variable**: Chọn biến có miền nhỏ nhất.

851 - **Least Constraining Value**: Chọn giá trị ít hạn chế các biến khác. [↗](#)

852 - **Forward Checking**: Kiểm tra ràng buộc ngay sau mỗi gán.

853 - Hiệu quả hơn Generate and Test khi kết hợp với AC-3.

854 4. **Kết quả**:

855 - Trả về gán đầy đủ thỏa mãn tất cả ràng buộc hoặc kết luận không có [↗](#)

giải pháp.

- 856 - ****Ưu điểm****:
- 857 - ****Generate and Test****: Đơn giản, dễ triển khai, đảm bảo tìm giải pháp nếu tồn tại. ➤
- 858 - ****AC-3****: Giảm đáng kể không gian tìm kiếm, cải thiện hiệu suất cho Backtracking và Generate and Test. ➤
- 859 - ****Backtracking****: Linh hoạt, có thể kết hợp với nhiều kỹ thuật tối ưu, hiệu quả hơn Generate and Test. ➤
- 860 - ****Nhược điểm****:
- 861 - ****Generate and Test****: Không hiệu quả cho các bài toán lớn (như 8-puzzle) do kiểm tra toàn bộ không gian tìm kiếm ($O(d^n)$). ➤
- 862 - ****AC-3****: Chỉ là tiền xử lý, không đảm bảo tìm giải pháp. ➤
- 863 - ****Backtracking****: Có thể chậm trong trường hợp xấu nhất ($O(d^n)$), đặc biệt nếu không sử dụng kỹ thuật tối ưu. ➤
- 864 - Trong 8-puzzle, biểu diễn CSP (gán giá trị cho ô) phức tạp hơn so với tìm kiếm trạng thái (state-space search) bằng A* hoặc IDA*. ➤

865

866 ---

867

868 ### Ứng dụng cụ thể trong 8-puzzle

- 869 - ****Generate and Test****:
- 870 - Tạo tất cả các hoán vị của lưới 3x3 ($9! = 362,880$ trạng thái, nhưng chỉ $9!/2 \approx 181,440$ trạng thái khả thi do tính chẵn lẻ của hoán vị). ➤
- 871 - Kiểm tra từng hoán vị để tìm trạng thái khớp với `'goal_state'` và có thể đạt được từ `'start_state'` thông qua các di chuyển hợp lệ. ➤
- 872 - Không thực tế do số lượng trạng thái lớn và không tận dụng cấu trúc của bài toán (như khả năng di chuyển ô trống). ➤
- 873 - ****AC-3****:
- 874 - Thu hẹp miền giá trị của các ô dựa trên ràng buộc:
- 875 - Tính duy nhất: Mỗi ô có giá trị khác nhau.
- 876 - Di chuyển ô trống: Ô trống (0) chỉ có thể xuất hiện ở các ô lân cận. ➤
- 877 - Khả năng đạt mục tiêu: Các giá trị phải dẫn đến `'goal_state'`.
- 878 - Ví dụ: Nếu ô 0 trong `'start_state'` cố định giá trị 5, AC-3 loại bỏ các giá trị khác khỏi miền của ô 0. ➤
- 879 - ****Backtracking****:
- 880 - Gán giá trị cho các ô (hoặc chuỗi di chuyển) để đạt `'goal_state'`.
- 881 - Kết hợp với AC-3 để giảm số giá trị cần thử, nhưng vẫn kém hiệu quả hơn A* trong 8-puzzle do không tận dụng heuristic như khoảng cách Manhattan. ➤

882

883 ---

884

885 ### So sánh các thuật toán trong CSPs

886

887	Thuật toán	Hoàn chỉnh	Tối ưu	Độ phức tạp thời gian	
	Độ phức tạp không gian	Ứng dụng chính			
888	-----	-----	-----	-----	➤
	--- -----	--- -----	--- -----	--- -----	

```

889 | **Generate and Test** | Có | Không |  $O(d^n * c)$  | ↗
      |  $O(n)$  hoặc  $O(d^n)$  | | Bài toán CSP nhỏ, kiểm tra toàn bộ |
890 | **AC-3** | Không | Không |  $O(e * d^3)$  | ↗
      |  $O(e)$  | | Tiền xử lý, giảm miền giá trị |
891 | **Backtracking** | Có | Có thể |  $O(d^n)$  | ↗
      |  $O(n)$  | | Giải CSP với không gian hữu hạn |
892
893 **Ghi chú**:
894 -  $\backslash(n\backslash)$ : Số biến.
895 -  $\backslash(d\backslash)$ : Kích thước miền lớn nhất.
896 -  $\backslash(e\backslash)$ : Số cung (ràng buộc đôi).
897 -  $\backslash(c\backslash)$ : Chi phí kiểm tra ràng buộc.
898
899 ---
900 ### 🖼️ **Hình ảnh các thuật toán được áp dụng trong trò chơi**
901
902 | **Thuật Toán** | **Minh Họa GIF** | ↗
      | |
903 | ----- | ----- | ↗
      | ----- |
904 | **AC-3 and backtracking** |  | ↗
      | width="500" alt="AC-3 and backtracking" > |
905 | **Backtracking** |  | ↗
      | width="500" alt="Backtracking" > |
906 | **Generate and Test** |  | ↗
      | width="500" alt="Backtracking" >|
907 ---
908 ## Reinforcement Learning
909 ---
910 ### 1. **Khái niệm chung về Reinforcement Learning và Q-Learning**
911 - **Reinforcement Learning (RL)** là một phương pháp học máy, trong đó ↗
      một tác nhân (agent) học cách đưa ra quyết định bằng cách thử và sai ↗
      trong một môi trường động, nhằm tối đa hóa phần thưởng tích lũy ↗
      (cumulative reward).
912 - **Q-Learning** là một thuật toán RL không dựa trên mô hình (model- ↗
      free), thuộc nhóm **Temporal Difference (TD) Learning**, học một chính ↗
      sách tối ưu thông qua việc ước lượng giá trị hành động (action-value ↗
      function) mà không cần biết mô hình chuyển đổi trạng thái của môi ↗
      trường.
913 - Mục tiêu của Q-Learning là tìm một chính sách (policy) ánh xạ từ trạng ↗
      thái đến hành động, sao cho tối đa hóa phần thưởng dài hạn trong môi ↗
      trường không xác định hoặc xác định.
914
915 ---
916
917 ### 2. **Các thành phần chính của Q-Learning**
918 - **Tác nhân (Agent)**: Thực thể đưa ra quyết định và học từ môi trường ↗
      (ví dụ: tác nhân di chuyển ô trống trong 8-puzzle).
919 - **Môi trường (Environment)**: Không gian mà tác nhân tương tác, bao gồm ↗

```


- tất cả trạng thái, hành động, và phần thưởng (ví dụ: lưới 3x3 của 8-puzzle với các trạng thái hoán vị).
- 920 - **Trạng thái (State, S)**:** Một mô tả của môi trường tại một thời điểm (ví dụ: một hoán vị cụ thể của các ô trong 8-puzzle, như `[2, 6, 5, 0, 8, 7, 4, 3, 1]`).
- 921 - **Hành động (Action, A)**:** Các lựa chọn mà tác nhân có thể thực hiện từ một trạng thái (ví dụ: di chuyển ô trống lên, xuống, trái, phải).
- 922 - **Phần thưởng (Reward, R)**:** Phản hồi số từ môi trường sau mỗi hành động, định lượng mức độ tốt của hành động (ví dụ: +1 khi đạt trạng thái mục tiêu, -1 cho mỗi bước di chuyển, hoặc 0 nếu không đạt mục tiêu).
- 923 - **Chính sách (Policy, π)**:** Chiến lược của tác nhân, ánh xạ từ trạng thái đến hành động (ví dụ: chọn hành động có giá trị Q cao nhất).
- 924 - **Hàm giá trị hành động (Q-Value, $Q(s, a)$)**:** Ước lượng phần thưởng tích lũy kỳ vọng khi thực hiện hành động 'a' từ trạng thái 's' và theo chính sách tối ưu sau đó.
- 925 - **Mô hình chuyển đổi (Transition Model)**:** Không cần thiết trong Q-Learning, vì thuật toán học trực tiếp từ kinh nghiệm (model-free).
- 926 - **Tỷ lệ học (Learning Rate, α)**:** Quy định mức độ cập nhật giá trị Q sau mỗi kinh nghiệm ($0 < \alpha \leq 1$).
- 927 - **Hệ số chiết khấu (Discount Factor, γ)**:** Quy định tầm quan trọng của phần thưởng tương lai so với phần thưởng hiện tại ($0 \leq \gamma \leq 1$).
- 928 - **Chiến lược khám phá (Exploration Strategy)**:** Thường sử dụng **ϵ -greedy**, cân bằng giữa khám phá (exploration) và khai thác (exploitation) để thử các hành động mới hoặc chọn hành động tốt nhất.
- 929
- 930 ---
- 931
- 932 **### 3. Giải pháp tổng quát của Q-Learning**
- 933
- 934 **#### Mô tả**
- 935 - Q-Learning học một hàm giá trị hành động $Q(s, a)$ bằng cách cập nhật giá trị Q dựa trên phần thưởng nhận được và giá trị Q của trạng thái tiếp theo, sử dụng phương pháp **Temporal Difference (TD)**.
- 936 - Thuật toán không cần biết mô hình môi trường (chuyển đổi trạng thái hoặc phân phối phần thưởng), mà học trực tiếp từ các mẫu kinh nghiệm (state, action, reward, next state).
- 937 - Mục tiêu là tìm chính sách tối ưu $\pi^*(s) = \arg\max_a Q^*(s, a)$, chọn hành động có giá trị Q cao nhất từ mỗi trạng thái.
- 938
- 939 **#### Cách hoạt động**
- 940 1. **Khởi tạo**:
- 941 - Khởi tạo bảng Q (Q-table) với các giá trị ban đầu (thường là 0) cho tất cả cặp trạng thái-hành động (s, a) .
- 942 - Đặt các tham số: tỷ lệ học α , hệ số chiết khấu γ , và tham số khám phá ϵ (cho chiến lược ϵ -greedy).
- 943
- 944 2. **Lặp qua các episode**:
- 945 - Một episode là một chuỗi hành động từ trạng thái ban đầu đến trạng thái kết thúc (ví dụ: đạt trạng thái mục tiêu hoặc vượt quá số bước

tối đa).

946 - Trong mỗi episode:

947 a. ****Chọn hành động****:

948 - Với xác suất ϵ , chọn hành động ngẫu nhiên (khám phá).

949 - Với xác suất $1 - \epsilon$, chọn hành động có giá trị Q cao nhất: $a = \arg\max_a Q(s, a)$ (khai thác).

950 b. ****Thực hiện hành động****:

951 - Thực hiện hành động a , nhận phần thưởng r và chuyển sang trạng thái tiếp theo s' .

952 c. ****Cập nhật giá trị Q****:

953 - Sử dụng công thức cập nhật Q-Learning:

954
$$Q(s, a) \leftarrow Q(s, a) + \alpha [r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$$

955

956

957 - r : Phần thưởng nhận được.

958 - $\max_{a'} Q(s', a')$: Giá trị Q tối đa từ trạng thái tiếp theo.

959 - α : Tỷ lệ học, điều chỉnh mức độ cập nhật.

960 - γ : Hệ số chiết khấu, cân nhắc phần thưởng tương lai.

961 d. ****Chuyển sang trạng thái tiếp theo****: Đặt $s = s'$ và lặp lại cho đến khi episode kết thúc.

962

963 3. ****Giảm khám phá****:

964 - Giảm dần ϵ (ϵ -decay) để chuyển từ khám phá sang khai thác, giúp tác nhân tập trung vào chính sách tối ưu khi học đủ lâu.

965

966 4. ****Kết thúc****:

967 - Sau nhiều episode, bảng Q hội tụ đến Q^* , biểu diễn giá trị hành động tối ưu.

968 - Chính sách tối ưu được suy ra: $\pi^*(s) = \arg\max_a Q^*(s, a)$.

969

970 **#### Đặc điểm**

971 - ****Hoàn chỉnh****: Không đảm bảo hoàn chỉnh trong không gian trạng thái vô hạn hoặc nếu không khám phá đủ. Trong không gian hữu hạn (như 8-puzzle), Q-Learning hội tụ đến chính sách tối ưu nếu tất cả cặp trạng thái-hành động được thăm đủ nhiều lần.

972 - ****Tối ưu****: Có, Q-Learning tìm chính sách tối ưu nếu hội tụ (với α và ϵ được điều chỉnh phù hợp).

973 - ****Độ phức tạp****:

974 - ****Thời gian****: Phụ thuộc vào số episode, số trạng thái $|S|$, và số hành động $|A|$. Trong trường hợp xấu, cần $O(|S| * |A|)$ cập nhật cho mỗi episode.

975 - ****Không gian****: $O(|S| * |A|)$ để lưu bảng Q.

976 - ****Ứng dụng****:

977 - Các bài toán điều khiển (robotics, trò chơi).

```

978 - 8-puzzle với mục tiêu học cách di chuyển ô trống để đạt trạng thái  ➤
    mục tiêu.
979 - Các môi trường có phần thưởng thừa thớt hoặc không xác định.
980
981 ##### **Ví dụ trong 8-puzzle**
982 - **Trạng thái**: Một hoán vị của lưới 3x3 (ví dụ: `[1, 2, 3, 4, 0, 5, 6,  ➤
    7, 8]`).
983 - **Hành động**: Di chuyển ô trống (lên, xuống, trái, phải).
984 - **Phần thưởng**:
985   - +100 khi đạt trạng thái mục tiêu (`[1, 2, 3, 4, 5, 6, 7, 8, 0]`).
986   - -1 cho mỗi bước di chuyển (khuyến khích đường đi ngắn).
987   - 0 cho các trạng thái không phải mục tiêu.
988 - **Q-Learning**:
989   - Tác nhân học bảng Q ánh xạ mỗi trạng thái-hành động đến giá trị kỳ  ➤
    vọng.
990   - Sau khi học, chọn hành động `( \arg\max_a Q(s, a) )` từ mỗi trạng  ➤
    thái để đạt mục tiêu.
991
992 ---
993
994 ### 4. **Ưu điểm và nhược điểm**
995
996 ##### **Ưu điểm**
997 - **Model-free**: Không cần biết mô hình chuyển đổi trạng thái, phù hợp  ➤
    với môi trường không xác định.
998 - **Học trực tiếp từ kinh nghiệm**: Dễ triển khai trong các môi trường  ➤
    phức tạp.
999 - **Chính sách tối ưu**: Hội tụ đến chính sách tối ưu nếu khám phá đủ.
1000 - **Linh hoạt**: Áp dụng được cho nhiều bài toán, từ trò chơi đến điều  ➤
    khiển robot.
1001
1002 ##### **Nhược điểm**
1003 - **Hiệu suất chậm**: Yêu cầu nhiều episode để hội tụ, đặc biệt trong  ➤
    không gian trạng thái lớn (8-puzzle có  $9!/2 \approx 181,440$  trạng thái).
1004 - **Khám phá không hiệu quả**: Chiến lược  $\epsilon$ -greedy có thể bỏ lỡ các trạng  ➤
    thái quan trọng trong không gian lớn.
1005 - **Phần thưởng thừa thớt**: Trong 8-puzzle, nếu phần thưởng chỉ có khi  ➤
    đạt mục tiêu, việc học sẽ chậm.
1006 - **Bảng Q lớn**: Trong các bài toán phức tạp, lưu trữ bảng Q tốn bộ nhớ,  ➤
    đặc biệt nếu `( |S| )` và `( |A| )` lớn.
1007
1008 ---
1009
1010 ### 5. **So sánh tổng quát**
1011 | Thuật toán | Hoàn chỉnh | Tối ưu | Độ phức tạp thời gian | Độ phức  ➤
    tạp không gian | Ứng dụng chính |
1012 |-----|-----|-----|-----|-----|  ➤
    -----|
1013 | **Q-Learning** | Có (nếu khám phá đủ) | Có (khi hội tụ) |  $O(|S| * |A|$ 

```

```

\\ * episodes) | O\\S\\ * \\A\\) | Trò chơi, robotics, 8-puzzle, điều khiển |
1014
1015 **Ghi chú**:
1016 - **|S|**: Số trạng thái.
1017 - **|A|**: Số hành động.
1018 - **episodes**: Số vòng lặp học.
1019
1020 ### 📷 **Hình ảnh thuật toán được áp dụng trong trò chơi**
1021
1022 | **Quá trình** | **Minh Họa GIF** |
1023 |-----|-----|
1024 | **Học** | <img src="images/QLearning.gif" width="500" alt="AC-3 and A*" |
1025 | **Giải** | <img src="images/QLearning_solve.gif" width="500" alt="Backtracking" |
1026
1027 ---
1028 ## 📄 Tác giả
1029
1030 **Nguyễn Trí Lâm**
1031 Trường: Sư phạm kỹ thuật TP.HCM
1032 MSSV: `23110250`
1033 Môn: `Trí Tuệ Nhân Tạo`
1034 Giáo viên hướng dẫn: `Phan Thị Huyền Trang`
1035
1036 ---
1037

```