**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP. HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

****

**ĐỒ ÁN HỌC PHẦN**

TÊN HỌC PHẦN: **TRÍ TUỆ NHÂN TẠO (Artificial Intelligence: AI)**

MÃ SỐ LỚP HP: **ARIN330585 - Nhóm 05 (Sáng thứ 7)**

Tên đề tài: **PHÁT TRIỂN HỆ THỐNG TÌM ĐƯỜNG ĐI NGẮN NHẤT TRÊN BẢN ĐỒ DỰA VÀO THUẬT TOÁN DIJKSTRA**

Họ tên sinh viên: **HỒNG TIẾN HÀO**

**Mã số sinh viên: 19133022 [19]**

**Lớp: 191330C**

**Ngày nộp: 19/06/2021**

**Ký tên:**

**TP.HCM, ngày 08 tháng 06 năm 2021**

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

**ĐỀ TÀI ĐỒ ÁN HỌC PHẦN**

**Giảng viên giảng dạy: VÕ XUÂN THỂ**

TÊN HỌC PHẦN: **TRÍ TUỆ NHÂN TẠO (Artificial Intelligence: AI)**

MÃ SỐ LỚP HP: **ARIN330585 - Nhóm 05 (Sáng thứ 7)**

Tên đề tài: **PHÁT TRIỂN HỆ THỐNG TÌM ĐƯỜNG ĐI NGẮN NHẤT TRÊN BẢN ĐỒ BẰNG THUẬT TOÁN DIJKSTRA**

Họ tên sinh viên: **HỒNG TIẾN HÀO**

**Mã số sinh viên:19133022 [19]**

**Lớp: 191330C**

**Tên sản phẩm đề tài: N5S7.19.HongTienHao.19133022.DoaAnHP.AI.rar**

**Công cụ sử dụng: Jupyter (Anaconda3)**

**Ngôn ngữ lập trình sử dụng: Python**

**Tập dữ liệu thực nghiệm: ảnh bản đồ trên internet**

**Nhận xét của giảng viên:**

**………………………………………………………..**

**………………………………………………………..**

**Điểm đánh giá:………….(…………………)**

**Ngày……./……../2021**

**Giảng viên Ký tên**

**TP.HCM, ngày 08 tháng 06 năm 2021**

**LỜI CẢM ƠN**

Lời nói đầu tiên, nhóm thực hiện xin được gửi đến thầy Võ Xuân Thể – giảng viên bộ môn lời cảm ơn chân thành và sâu sắc nhất.

Em thực hiện xin cảm ơn sự quan tâm và giúp đỡ tận tình của thầy trong suốt quá trình giảng dạy. Cảm ơn thầy đã luôn giải đáp những thắc mắc cũng như đưa ra những nhận xét, góp ý giúp nhóm thực hiện cải thiện chất lượng công việc của nhóm.

Vì khả năng còn hạn chế nên trong quá trình thực hiện báo cáo không tránh khỏi sai sót, kính mong nhận được những ý kiến đóng góp từ thầy để nhóm có thể cải thiện hơn sau này.

Em xin chân thành cảm ơn.

**DANH MỤC CÁC TỪ VIẾT TẮT**

|  |  |
| --- | --- |
|  | |
| CSDL hoặc DB | Cơ sở dữ liệu: DataBase |
|  | |
| AI | Trí tuệ (Trí thông minh) nhân tạo: Artificial Intelligence |
| ANN | Mạng nơ ron nhân tạo: Artificial Neural Network |
| ES | Hệ thống chuyên gia: Expert Systems |
| ML | Máy học = Học máy: Machine Learning |
| NLP | Xử lý ngôn ngữ tự nhiên: Natural Language Processing |

**DANH MỤC CÁC THUẬT NGỮ**

|  |  |
| --- | --- |
| Bài toán (vấn đề) AI:  AI Problem | Là các tình huống thực tế mà con người cần giải quyết thuộc một lĩnh vực nào đó, thông thường phải dựa vào “trí khôn” con người. |
| Giao diện người dùng  (User-Interface) | Là hệ thống các màn hình giao tiếp cho phép người sử dụng tương tác với các thành phần phần mềm, điều khiển phần mềm hoạt động theo yêu cầu của người dùng - tương ứng các chức năng hiện có của phần mềm. |
| Người dùng (User):  Tài khoản (Account) | Là một quyền làm việc trên hệ thống phần mềm được cấp phát cho một cá nhân thông qua tên tài khoản (username) và mật khẩu (password). |
|  |  |
| BigData | Dữ liệu lớn: là một tập hợp dữ liệu rất lớn và phức tạp, không thể xử lý dữ liệu bằng các phương pháp truyền thống. |
| Heuristics | Là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, nhằm đưa ra một giải pháp mà không được đảm bảo là tối ưu |

**MỤC LỤC**

[PHIẾU GIAO NHIỆM VỤ ĐỀ TÀI ĐỒ ÁN HỌC PHẦN 9](#_Toc74931914)

[Chương 1: GIỚI THIỆU VỀ ĐỀ TÀI 10](#_Toc74931915)

[**1.1.** Tổng quan về đề tài 10](#_Toc74931916)

[**1.2.** Nội dung chuyên môn chính của đề tài 10](#_Toc74931917)

[**1.3.** Ngôn ngữ lập trình và công cụ sử dụng 10](#_Toc74931918)

[**1.4.** Sản phẩm của đề tài 10](#_Toc74931919)

[**1.5.** Bố cục của báo cáo 10](#_Toc74931920)

[Chương 2: CƠ SỞ LÝ THUYẾT VỀ TRÍ TUỆ NHÂN TẠO 12](#_Toc74931921)

[**2.1.** Tổng quan về Trí tuệ nhân tạo (AI) 12](#_Toc74931922)

[2.1.1. Các khái niệm 12](#_Toc74931923)

[2.1.2. Vai trò AI 14](#_Toc74931924)

[2.1.3. Nền tảng kỹ thuật của AI 14](#_Toc74931925)

[2.1.4. Các lĩnh vực nghiên cứu và ứng dụng cơ bản của AI 15](#_Toc74931926)

[**2.2.** Giới thiệu về các bài toán 16](#_Toc74931927)

[**2.3.** Giới thiệu về các bài toán (vấn đề) AI 16](#_Toc74931928)

[2.1.5. Một số khái niệm 16](#_Toc74931929)

[2.1.6. Tìm kiếm lời giải (searching) cho AI Problem 19](#_Toc74931930)

[**2.4.** Biểu diễn một số bài toán AI 19](#_Toc74931931)

[2.1.7. Các bài toán Trò chơi 19](#_Toc74931932)

[2.1.8. Cây không gian trạng thái của Bài toán AI 20](#_Toc74931933)

[**2.5.** Các phương pháp biểu diễn tri thức 22](#_Toc74931934)

[2.1.9. Phân loại tri thức theo phương pháp biểu diễn 22](#_Toc74931935)

[2.1.10. Phân loại tri thức theo nền tảng hình thành Cơ sở tri thức 23](#_Toc74931936)

[2.1.11. Logic mệnh đề và Logic vị từ 24](#_Toc74931937)

[2.1.1.13.3. Mở rộng Logic vị từ 25](#_Toc74931938)

[2.1.1.13.4. Logic vị từ bậc||cấp cao 25](#_Toc74931939)

[2.1.1.13.5. Logic tình huống 26](#_Toc74931940)

[2.1.1.13.6. Logic trạng thái (modal) 26](#_Toc74931941)

[2.1.1.13.7. Logic xác suất và Logic khả xuất 27](#_Toc74931942)

[2.1.12. Biểu diễn tri thức bằng Frame (Cấu trúc = "Khung") 27](#_Toc74931943)

[2.1.13. Suy diễn tri thức bằng luật dẫn xuất 28](#_Toc74931944)

[2.1.1.14. Khái niệm 28](#_Toc74931945)

[2.1.1.15. Luật dẫn xuất trong Cơ sở tri thức: Suy diễn tiến 29](#_Toc74931946)

[2.1.2. Biểu diễn suy luận bằng đồ thị AND/OR 30](#_Toc74931947)

[2.1.3. Các biểu diễn tương đương trong Logic mệnh đề và Logic vị từ 31](#_Toc74931948)

[2.1.4. Xử lý ngôn ngữ tự nhiên (Natural Language Processing = NLP): Voice Assistant 31](#_Toc74931949)

[**2.6.** Các phương pháp tìm kiếm lời giải của bài toán AI 31](#_Toc74931950)

[2.1.14. Giới thiệu 31](#_Toc74931951)

[2.1.15. Các phương pháp tìm kiếm mù (Blind Search) = tìm kiếm không có thêm thông tin (uninformed search) 32](#_Toc74931952)

[2.1.16. Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) = tìm kiếm với thông tin bổ sung (Informed Search) = Cải tiến BFS 34](#_Toc74931953)

[2.1.17. Lập trình Logic (Logic Programming) phát triển cơ chế lập luận logic 35](#_Toc74931954)

[Chương 3: GIỚI THIỆU VỀ MÔ HÌNH BÀI TOÁN AI: TÌM ĐƯỜNG ĐI TRÊN BẢN ĐỒ BẰNG THUẬT TOÁN DIJKSTRA 36](#_Toc74931955)

[3.1. Mô tả bài toán AI 36](#_Toc74931956)

[3.2. Các yếu tố xác định bài toán AI 36](#_Toc74931957)

[3.3. Giới thiệu nguồn dữ liệu gốc dùng thực nghiệm bài toán AI 36](#_Toc74931958)

[Chương 4: BIỂU DIỄN BÀI TOÁN AI VÀ CƠ SỞ TRI THỨC: TÌM ĐƯỜNG ĐI TRÊN BẢN ĐỒ BẰNG THUẬT TOÁN DIJKSTRA 37](#_Toc74931959)

[4.1. Giới thiệu 37](#_Toc74931960)

[4.2. Biểu diễn bài toán AI 37](#_Toc74931961)

[Chương 5: PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI CHO BÀI TOÁN AI: TÌM ĐƯỜNG ĐI NGẮN NHẤT TRÊN BẢN ĐỒ BẰNG GIẢI THUẬT DIJKSTRA 38](#_Toc74931962)

[5.1. Giới thiệu 38](#_Toc74931963)

[5.2. Mô tả sơ bộ về mô hình cây không gian trạng thái của bài toán AI 39](#_Toc74931964)

[5.3. Xác định phương pháp tìm kiếm lời giải cho bài toán AI: Tìm đường đi ngắn nhất trên bản đồ 40](#_Toc74931965)

[Chương 6: GIỚI THIỆU SẢN PHẨM CỦA ĐỀ TÀI AI: TÌM ĐƯỜNG ĐI NGẮN NHẤT TRÊN BẢN ĐỒ BẰNG THUẬT TOÁN DIJKSTRA 40](#_Toc74931966)

[6.1. Giới thiệu sản phẩm bài toán AI: Tìm đường đi ngắn nhất trên bản đồ bằng giải thuật dijkstra 40](#_Toc74931967)

[6.2. Kết quả thực nghiệm sản phẩm bài toán AI: Tìm đường đi ngắn nhất trên bản đồ bằng giải thuật dijkstra 46](#_Toc74931968)

[6.3. Nhận xét và đánh giá về sản phẩm bài toán AI: Tìm đường đi ngắn nhất trên bản đồ bằng giải thuật dijkstra 46](#_Toc74931969)

[Chương 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI 47](#_Toc74931970)

[7.1. Kết luận 47](#_Toc74931971)

[7.1.1. Những kết quả đạt được 47](#_Toc74931972)

[7.1.2. Hạn chế 47](#_Toc74931973)

[7.2. Hướng phát triển 47](#_Toc74931974)

[7.2.1. Hướng khắc phục các hạn chế 47](#_Toc74931975)

[7.2.2. Hướng mở rộng đề tài 47](#_Toc74931976)

[DANH MỤC TÀI LIỆU THAM KHẢO 48](#_Toc74931977)

[CÁC PHỤ LỤC 49](#_Toc74931978)

[ĐÁNH GIÁ KẾT QUẢ HỌC TẬP HỌC PHẦN 50](#_Toc74931979)

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA CÔNG NGHỆ THÔNG TIN**

# PHIẾU GIAO NHIỆM VỤ ĐỀ TÀI ĐỒ ÁN HỌC PHẦN

|  |  |
| --- | --- |
| Giảng viên giảng dạy: **VÕ XUÂN THỂ** | |
| Tên học phần: **Trí tuệ nhân tạo (AI)**  Mã số lớp HP: **ARIN330585 - N5S7** | |
| Tên đề tài: **Phát triển hệ thống tìm đường đi ngắn nhất trên bản đồ dựa vào giải thuật dijkstra** | |
| Sinh viên thực hiện: **HỒNG TIẾN HÀO, 19133022** | |
| Thời gian thực hiện: **27/02/2021**  đến **08/06/2021** | |
| **Yêu cầu của đề tài**  **Lý thuyết:** Vận dụng AI vào thực tiễn và tiếp cận thành tựu tiên tiến trong lĩnh vực AI:  + Nền tảng và đặc trưng của các dạng bài toán AI và giải thuật tương ứng.  + Một số phương pháp biểu diễn không gian (trạng thái) giải quyết bài toán AI: biểu diễn tri thức và suy diễn logic.  + Một số phương pháp thông dụng trong tìm kiếm lời giải bài toán AI = giải quyết bài toán thông qua tìm kiếm lời giải (Solving Problems by Searching), đặc biệt là DFS và BFS,…  **Thực hành:** Phát triển hệ thống tìm đường đi ngắn nhất trên bản đồ dựa vào giải thuật dijkstra:  + Thư viện: opencv, matplotlib, numpy.  + Giải thuật chính của sản phẩm sử dijkstra  + Tập thực nghiệm là ảnh bản đồ trên internet  + Các chức năng chính của sản phẩm: Tìm đường đi trên bản đồ cho trước. | |
| **GIẢNG VIÊN** | **Ngày 27 tháng 02 năm 2021**  **SV Thực hiện** |

# GIỚI THIỆU VỀ ĐỀ TÀI

## Tổng quan về đề tài

Đề tài được lấy ý tưởng từ bài toán người đưa thư Trung Quốc, từ đó phát triển thành hệ thống tìm đường trên một dữ liệu hình ảnh bản đồ cho trước. Đề tài với giải thuật chính được sử dụng là dijkstra, cùng với các thư viện opencv, numpy, matplotlib để hỗ trợ xây dựng sản phẩm.

## Nội dung chuyên môn chính của đề tài

+ Thư viện:

Opencv: dùng để xử lý ảnh từ tập dữ liệu

Numpy: dùng xử lý mảng

Matplotlib: hỗ trợ trực quan hình ảnh từ tập dữ liệu

+ Giải thuật: sử dụng giải thuật dijkstra

+ Kết quả thực nghiệm: Thành công

## Ngôn ngữ lập trình và công cụ sử dụng

+ Ngôn ngữ lập trình: Python (phiên bản 3.8.3)

+ Công cụ sử dụng: Jupyter (anaconda3)

## Sản phẩm của đề tài

**N5S7.19.HongTienHao.19133022.DoaAnHP.AI.rar**

## Bố cục của báo cáo

Chương 1: Giới thiệu tổng quan về đề tài

Chương 2: Cơ sở lý thuyết của học phần: gồm những lý thuyết cơ bản về AI liên quan đến đề tài.

Chương 3: Giới thiệu về thư viện và giải thuật AI: Tìm đường đi ngắn trên bản đồ

Chương 4: Mô hình bài toán AI và cài đặt giải thuật tìm kiếm lời giải

Chương 5: Giới thiệu về sản phẩm hệ thống AI: Tìm đường đi ngắn nhất trên bản đồ

Chương 6: Kết luận về kết quả đạt được và những tồn tại, trên cơ sở đó đề xuất các giải pháp khắc phục tồn và hướng mở rộng đề tài.

# CƠ SỞ LÝ THUYẾT VỀ TRÍ TUỆ NHÂN TẠO

## Tổng quan về Trí tuệ nhân tạo (AI)

### Các khái niệm

* + - 1. **Trí tuệ nhân tạo = AI (Artificial Intelligence)**

+Là ngành khoa học nghiên cứu và ứng dụng các nền tảng kỹ thuật làm cơ sở cho việc thiết kế và sản xuất ra các hệ thống, đặc biệt là các hệ thống CNTT, có khả năng thực hiện các hành vi công việc “khôn” như con người.

+...Hành vi và nhận thức

+...Đặc trưng

* Không nhất thiết chính xác 100%: tỷ lệ sai tùy thuộc bài toán AI
* Không nhất thiết phải chứng minh hoặc giải thích, cần hiêu quả của giải pháp AI mang lại
* Học và tự học
* “Khôn” dần theo thời gian
* Sử dụng ngôn ngữ lập trình và công cụ riêng
  + - 1. **Phân loại hệ thống Trí tuệ nhân tạo**
* Nền tảng “trí khôn”: Heuristic và ES
* Biểu hiện: hành vi và nhận thức
* Nền tảng công nghệ: ML, NLP, ANN, ...
  + - 1. **Turing Test kiểm tra khả năng khôn của các hệ thống AI**

+ Tri thức là hệ thống sự hiểu biết bằng lý thuyết hay thức tế về một chủ đề hay lĩnh vực

+ Tri thức tự nhiên OR tri thức hệ thống Phân biệt (ss) : PM QUẢN LÝ (TRUYỀN THỐNG) – PM TTNT

+ Chương trình truyền thống = cấu trúc dữ liệu + thuật toán (thuật giải)

+ Chương trình trí tuệ nhân tạo =cơ sở tri thức (knowledge base) + quy tắc suy diễn (inference engine)

* + - 1. **Tác tử (Agent: Intelligent Agent) trong hệ thống AI**

+Tác tử(Agent) trong hệ thống AI là 1 nhân tố nhỏ nhất biểu hiện khả năng “Khôn” [nhận thức || hành động] cấu thành nên hệ thống AI

+Tác tử (Agent) có thể biểu diễn dạng ánh xạ:

f:P\* 🡺 A

[Lịch sử quá trình nhận thức] 🡺 [Hành động]

* + - 1. **Tri thức là gì (Knowledge)?**

- Tri thức là hệ thống sự hiểu biết bằng lý thuyết hay thực tế về một chủ đề hay lĩnh vật.

- Tri thức tự nhiên OR tri thức hệ thống

+Tri thức tự nhiên: bản thân trong tự nhiên mà có

+Tri thức hệ thống: do con người vun đắp

- Phân biệt (ss): PM QUẢN LÝ TRUYỀN THỐNG (MIS: PM HTTTQL) – PM TINT

+Chương trình truyền thống = cấu trúc dữ liệu + thuật toán (giải thuật)

+Chương trình trí tuệ nhân tạo = Cơ sở tri thức (knowledge base) + qui tắc suy diễn (inference engine)

* + - 1. **Cơ sở tri thức (Knowledge Base: KB)**

- Cơ sở tri thức là tập hợp các tri thức liên quan đến vấn đề quan tâm được hệ thống và lưu trữ trên các thiết bị số (máy tính) dưới một dạng nào đó.

- Cơ sở tri thức chứa các kiến thức được sử dụng để giải quyết các vấn đề (bài toán) trong trí tuệ nhân tạo

* + - 1. **Hệ quản trị cơ sở tri thức (KBMS)**

- Hệ cơ sở tri thức tối thiểu có hai chức năng c/b

+ Khối tri thức: hình thành và duy trì cơ sở tri thức

+ Khối điều khiển: cung cấp các công cụ cho phép xd và pt các cơ chế suy diễn

-Với các hệ thống phức tạp, khối đk cũng có thể là một hệ thống cơ sở tri thức khác

🡺 ....chứa .....các siêu tri thức (tri thức về các tri thức)

-Cấu trúc chương trình truyền thống (bên trái) và cấu trúc chương trình trí tuệ nhân tạo (bên phải)

\*Cơ chế suy diễn: là phương pháp vận dụng tri thức trong cơ sở tri thức để giải quyết vấn đề.

-Một số hệ quản trị cơ sở tri thức thông dụng: IPLV, autoLISP, PLANNER, PROLOG,...

### Vai trò AI

-Là nhân tố trung tâm và gần như không thể thiếu trong các HT hiện nay và tương lai.

-AI và yếu tố nền tảng mang tính bắt buộc của các HT trong thời đại CMCN 4.0 (cách mạng công nghiệp lần thứ 4: IR 4.0 = The Fourth Industrial Revolution)

### Nền tảng kỹ thuật của AI

[1] Lý thuyết giải bài toán và suy diễn thông minh

[2] Lý thuyết tìm kiếm may rủi (Ngẫu nhiên)

[3] Các ngôn ngữ công cụ về trí tuệ nhân tạo

[4] Lý thuyết biểu diễn tri thức và hệ chuyên gia

[5] Lý thuyết về nhận dạng và xử lý tiếng nói

[6] Người máy (ROBOT)

[7] Tâm lý học trong xử lý thông tin

### Các lĩnh vực nghiên cứu và ứng dụng cơ bản của AI

* Các bài toán tối ưu (giải pháp tối ưu, bài toán tối ưu)
* Dự đoán, dự báo
* games

Một số bài toán liên quan Trí tuệ nhân tạo

* Nhận dạng mẫu
  + Nhận dạng chữ cái quang học (Optical character recognition)
  + Nhận dạng chữ viết tay
  + Nhận dạng tiếng nói
  + Nhận dạng khuôn mặt (gương mặt)
* Xử lý ngôn ngữ tự nhiên, dịch tự động (dịch máy) và chatterbot
* Điều khiển phi tuyến và Robotics
* Computer vision, Thực tế ảo và Xử lý ảnh
* Lý thuyết trò chơi và Lập kế hoạch (Strategic planning)
* Trò chơi TTNT va Computer game bot

## Giới thiệu về các bài toán

## Giới thiệu về các bài toán (vấn đề) AI

### Một số khái niệm

* + - 1. **AI Problem (Bài toán AI = Vấn đề AI)**

Bài toán AI là các tính huống thực tế mà con người cần giải quyết thuộc một lĩnh vực nào đó và thông thường phải dựa vào “trí khôn” con người.

VD: game cờ tướng, Bài toán tìm đường đi tốt nhất, Tư vấn pháp luật, Dự đoán hành vi khách hang,…

\*Với một bài toán AI thông thường được xác định bởi: 4 yếu tố cơ bản:

+ Tập hợp các trạng thái của bài toán.

VD: trạng thái của bàn cờ (quân cờ)

+ Trạng thái bắt đầu (một hoặc một số)

+ Có mục tiêu đặt ra thông qua hàm mục tiêu mà kĩ thuật AI phải giải quyết cho bài toán đó;

VD: . Bài toán cờ tướng => mục tiêu: “chiếu tướng bí”.

. Rubik => mục tiêu: 6 mặt cùng màu

.Tìm đường đi => mục tiêu: tìm được 1 (một số) lộ trình đường đi mà người đi họ “hài lòng” nhất

+ Có cơ chế chuyển trạng thái: từ “trạng thái” này sang “trạng thái” khác

VD: .Cờ tướng = nước đi của một quân cờ trên bàn cờ tương ứng trạng thái đầu.

.Tháp Hà Nội = trường hợp chuyển “thớt đĩa” từ cột này sang “cột khác”.

Chú ý: Cơ chế chuyển trạng thái phải có quy tắc || qui luật || qui định || ràng buộc,…

\*(Hàm) Mục tiêu của Bài toán AI

-Là một trong những cơ chế xác định mục tiêu của bài toán AI theo định lượng (tính toán được bằng giá trị nào đó)

-Đa dạng tùy thuộc vào từng bài toán cụ thể

VD: Xét bài toán tìm đường đi tốt nhất từ A đến B

+Mục tiêu: có rất nhiều tiêu chí đặt ra để đánh giá độ tốt của lộ trình đường đi mà mình đã tìm được, mỗi tiêu chí là một tham số hay một đối số của hàm mục tiêu.

+Tiêu chí:

.Khoảng cách (x) ngắn nhất

.Thời gian (t) nhanh nhất

.Chi phí (z) rẻ nhất

Mỗi một tiêu chí nêu trên được đo lường thông qua một giá trị nào đó. Chẳng hạn: khoảng cách của một lô trình là tổng chiều dài đoạn đường đó đi qua hoặc có thể tính toán bằng cách khác dựa vào thực tiễn. Với mỗi tham số nêu trên sẽ có một giá trị “trọng số” (hệ số xác định mức độ quan trọng của đối số hay tham số đó), các giá trị trọng số này là khác nhau tùy vào trường hợp bài toán, mục tiêu người dung, tình huống cụ thể ứng dụng bài toán,… Chẳng hạn, khi cần đi gấp thì lúc đó đặt yếu tố thời gian có trọng số cao nhất.

-Hàm mục tiêu có rất nhiều dạng: thông thường là dạng tuyến tính.

->F = a1\*x + a2\*y + a3\*z

Trong đó: x, y, z là các đối số nêu trên.

a1, a2, a3 là trọng số của các đối số trên.

Thông thường các đối số của các giá trị x, y, z được nạp từ dữ liệu có sẵn, thông thường

tập dữ liệu rất lớn như kho dữ liệu, sử dụng thư viện Pandas và Numpy để nạp các kho dữ

liệu này lên truyền các hàm mục tiêu để xác định trạng thái nào tối ưu nhất trong không

gian trạng thái.

Các hàm mục tiêu có thể là hàm phi tuyến tùy thuộc vào bài toán cụ thể, tuy nhiên dạng này ít phố biến, tốn kém trong thực thi chương trình nên rất ít dụng (hàm bậc 2 hệ số a âm). Đối với hàm phi tuyến hàm mục tiêu để tìm được giải pháp tốt nhất là đỉnh parabol

* + - 1. **Các yếu tố cơ bản xác định bài toán AI**

Tập hợp các trạng thái của BT;

+ Mỗi trạng thái là một tình huống, nó có thể là 1 giải pháp của bài toán; không nhất thiết là giải pháp đúng và thậm chí là giải pháp đúng cũng không có nghĩa là đúng nhất

Vd: trạng thái của bàn cờ hoặc nước đi của cờ tùy thuộc vào trạng thái biểu diễn

+ Trạng thái bắt đầu (1 hoặc 1 số)

Cần phải xác định trạng bắt đầu hoặc một số trạng thái bắt đầu

+ Có mục tiêu đặt ra (thông qua hàm mục tiêu) mà kỹ thuật AI phải giải quyết cho bài toán đó

Vd: Cơ chế chuyển trạng thái: từ “trạng thái” này sang “trạng thái”khác là quy tắc, quy luật hay quy ước để chuyển từ trạng thái này sang trạng thái khác trong tập không gian trạng thái

Vd: nước đi của 1 trong 2 quân cờ trong bàn cờ tướng

Như vậy cài đặt một bài toán trí tuệ nhân tạo phải đủ 4 yếu tố nêu trên

* + - 1. **(Hàm) Mục tiêu của Bài toán AI**

Hàm mục tiêu là một cơ chế một công thức và một quy tắc để xác định một trạng thái nào đó trong không gian trạng thái của bài toán được xem là mục tiêu đạt được của bài toán đó; nếu một trạng thái nào đó trong không gian trạng thái mà thỏa mãn các điều kiện của mục tiêu thì được xem là một lời giản/giải pháp của bài toán Hàm mục tiêu rất đa dạng và tùy thuộc vào bài toán cụ thể, có 2 dạng mục tiêu là mục

tiêu định lượng và mục tiêu định tính và khi là mục tiêu định lượng thì có thể xác định

bằng hàm và lúc đó gọi là hàm mục tiêu

Vd: bài toán = A=> B

+ Mục tiêu để xác định đường đi tốt nhất phụ thuộc vào nhiều đối số hoặc tham số

. (x): k/c

. (y): t/g => min

. (z): tiền => min

+ Các đối số và hàm mục tiêu nêu trên rất đa dạng tùy thuộc vào bài toán và được đo lường tính toán thông qua lộ trình cụ thể

Vd: có thể tính tổng độ dài của các đoạn đường trong lộ trình đó dựa vào tham số trên bản đồ có sẵn; chú ý với bài toán này ta phải xác định khoảng cách trước khi đi ( tại điểm A )

Yếu tố thời gian và yếu tố chi phí được xác định thông qua lịch sử sử dụng AI này nhờ vào khả năng tự học của nó

Ngoài các tiêu chí nêu trên có thể có nhiều tiêu chí khác, con đường đi tốt nhất phong

cảnh đẹp có nhiều cây xanh..

Mỗi một đối số hay tham số nêu trên đều có 1 mức độ quan trọng khác nhau tùy thuộc

vào từng tình huống cụ thể của bài toán và mỗi mức độ quan trọng được đánh giá bởi một

hệ số kèm theo, thường gọi là “ trọng số ” (là một hệ số đánh giá mức độ quan trọng)

Hàm mục tiêu cũng rất đa dạng tùy vào từng trường hợp cụ thể và thông thường là dạng

hàm mục tiêu tuyến tính, phổ biến nhất là hàm mục tiêu tuyến tính và được biểu diễn như

sau

 F = a1\*x + a2\*y + a3\*z

Ngoài hàm mục tiêu tuyến tính nêu trên ta còn dạng hàm mục tiêu phi tuyến đặc biệt là

hàm mục tiêu bậc 2 dạng parabol với hệ số a âm khi đó trạng thái mục tiêu được xem là

đỉnh của parabol Các dạng hàm mục tiêu nêu trên là một trong những dạng tiêu chí cơ bản để xác định trạng thái mục tiêu của AI

### Tìm kiếm lời giải (searching) cho AI Problem

* là quá trình khám phá các chuỗi hành động mà có thể đạt được những mục tiêu của bt
* quá trình tìm kiếm lời giải thông thương là thủ tục duyệt nhờ vào các agent chạy qua các trạng thái trong không gian trạng thái của bt AI
  + vòng lặp chạy các agent/KGTT, phát hiện ra trạng thái mục tiêu
  + cây KGTT của AI bài toán AI thông thường là chưa có sẵn và khi duyệt agent đến đâu thì cây KGTT mở nút ra đến đó
    - 1. **Một số dạng bài toán (vấn đề) AI**
* bài toán một trạng thái đơn giải
* bài toán đa trạng thái
* bài toán ngẫu nhiên
* bài toán thăm dò: “thử” – “sai”
  + - 1. **Các cấp độ tìm kiếm lời giải cho bài toán (vấn đề) AI**
* tìm ra lời giải
* lời giải tốt nhất
* thời gian tối thiểu

## Biểu diễn một số bài toán AI

### Các bài toán Trò chơi

VD1: Trò chơi m quân cờ (cờ ta canh) = tic tac toe (có nhiều dạng)/n cột dòng

+Trò chơi có m = n2 – 1 quân cờ; VD: n = 3

Ban đầu Đích

1 4 3 1 2 3

7 6 8 4

1. 8 2 7 6 5

* Gồm một bảng có kích thước n = 3 (3x3) với n2 – 1 = 8 quân cờ được đánh số từ 1 đến 8 và một ô trống.
* Một quân cờ đứng cạnh ô trống có thể di chuyển vào ô trống đó
  + Trạng thái (không gian trạng thái): là một ô tả trong đó chỉ rõ vị trí của mỗi quân cờ trong 8 quân cờ ở trong 9 ô vuông.

=>Để biểu diễn, ta xét cả vị trí của ô trống =>9! trạng thái

* + Các toán tử (luật di chuyển): ô trống di chuyển sang trái, phải, lên trên, đi xuống.=L-RU-D
  + Kiểm tra mục tiêu: trạng thái khớp với hình dạng mục tiêu.
  + Trạng thái bắt đầu (như trên: hình bên trái)
  + Chi phí đường đi: mỗi bước đi chi phí là 1, vì vậy chi phí đường đi bằng độ dài đường đi bằng tổng chi phí bước đi.

=>Hàm chi phí: số quân cờ bị sai vị trí + tổng số bước đi chuyển vị trí.

Trò chơi quân cờ nêu trên có thể mở rộng thành trò chơi 15 quân cờ và nhiều hơn (n quân)

### Cây không gian trạng thái của Bài toán AI

* Định nghĩa: không gian trạng thái của bài toán (vấn đề) AI: là tập các trạng thái (=nút trên cây) có thể đạt được bằng chuỗi hành động bất kì xuất phát từ trạng thái ban đầu. Cây không gian trạng thái sẽ được mở rộng dần (các nút trạng thái) theo tiến trình tìm kiếm lời giải của bài toán AI.
* Một hành trình (lộ trình lời giải) trong không gian trạng thái = các tập hợp hành động xuất phát từ trạng thái này đến trạng thái khác.
  + VD2: Tháp Hà Nội (Hanoi Tower)
    - Cho 3 cọc: 1, 2, 3
    - Cọc 1: ban đầu có n đĩa sắp xếp theo thứ tự từ dưới lên: to -> bé (to dần xuống dưới = bé dần lên trên)
    - Yêu cầu: chuyển hết n đĩa cọc 1 sang cọc 3 chỉ sử dụng cọc 2 làm trung gian với điều kiện:
      * .Mỗi lần di chuyển 1 đĩa
      * .Trong mỗi cọc không cho phép đĩa to nằm trên đĩa bé
  + VD3: Game caro
  + VD4: bài toán 8 quân hậu
    - Mục tiêu của bài toán 8 quân hậu: đặt 8 con hậu trên một bàn cờ vua sao cho không con nào ăn con nào. (Một con hậu sẽ ăn bất kì con nào nằm trên cùng hàng, cùng cột hoặc cùng đường chéo với nó). Và bài toán tổng quát n con hậu:
    - Gợi ý: có 2 phương pháp giải quyết chính
      * Phương pháp gia tăng: đặt các con hậu từng con một.
      * Phương pháp trạng thái: bắt đầu với 8 con hậu và tiến trình di chuyển các con hậu.
    - Giải thích 2 phương pháp trên:
      * Phương pháp gia tăng
      * Phương pháp trạng thái
      1. **Định nghĩa**
      2. **Phương pháp giải quyết bài toán AI**
         1. **Phương pháp gia tăng**
         2. **Phương pháp trạng thái**
      3. **Sự bùng nổ không gian trạng thái**

## Các phương pháp biểu diễn tri thức

### Phân loại tri thức theo phương pháp biểu diễn

* + - 1. **Tri thức sự kiện**

Là một khẳng định về một sự kiện, hiện tượng hay một khái niệm nào đó trong một hoàn cảnh không gian hoặc thời gian nhất định.

Ví dụ:

Khẳng định về hiện tượng: “Mặt trời lặn ở phương Tây”.

Khái niệm về: “Tam giác đều là tam giác có 3 góc bằng nhau”.

* + - 1. **Tri thức mô tả**

Tri thức mô tả đối tượng, hiện tượng, vấn đề trong tự nhiên

* + - 1. **Tri thức thủ tục**
* Là tri thức biểu diễn cách giải quyết vấn đề, quy trình xử lý các công việc, lịch trình tiến hành các thao tác Các dạng của tri thức thủ tục thường dùng là các “luật” , chiến lược, lịch trình.
  + VD1: NẾU xe máy không khởi động được THÌ đầu tiên kiểm tra bugi…
  + VD2: Trang web không truy xuất được?
    - B1: Các dạng kết nối mạng, PC
    - B2: Cấu hình thông số kỹ thuật: ip, gateway, DNS,…
    - B3: Thiết lập bảo mật: firewall,…
    - B4: Router
    - B5: Host web
    - B6: Trang web
      1. **Tri thức mêta**

Tham chiếu đến các tri thức khác (dựa trên tri thức thứ cấp)

VD: SOP: vị từ cấp 2 ;FOP: vị từ cấp 1

VD: hệ thống (phần mềm) hỗ trợ xây dựng phần mềm mới [nên sử dụng ES, Heuristic]

* + - 1. **Tri thức có cấu trúc**

Tri thức về các quan hệ giữa các khái niệm, quan hệ giữa các đối tượng có quy tắc hình thành các yếu tố Agent (token) bao gồm các yếu tố cấu thành giống nhau, giá trị khác nhau (tương tự cấu trúc dữ liệu)…

### Phân loại tri thức theo nền tảng hình thành Cơ sở tri thức

* + - 1. **Tri thức hệ Heuristic**
* Là quy luật xác suất thống kê = quy luật số đông (lớn)
  + Có 2 yêu cầu cơ bản : đủ lớn và đủ phổ biến/dàn trải
* Tri thức heuristic thường được coi là một “mẹo” (tip) nhằm dẫn dắt tiến trình lập luận.

=> Vì vậy có nhiều rủi ro, đơn giản,..=> có tỷ lệ sai (phải dùng ràng buộc đảm bảo có suy

luận của hệ thống AI có giá trị ví dụ sử dụng thư viện Python-constraint).

* + - 1. **Tri thức hệ chuyên gia (ES = Expert System)**
* Chuyên gia (Expert) là hệ thống AI được xây dựng và hình thành dựa trên cơ sở chuyên gia
  + +Người có đầy đủ kỹ năng, kiến thức sâu (cả về luật và các sự kiện ) về một lĩnh vực nào đó;
  + Người có thể làm những việc mà người khác ít khả năng làm được
    - VD : bs=-có chuyên môn + kinh nghiệm
* Hệ chuyên gia là chương trình máy tính có thể thực hiện các công việc, vấn đề trong thuộc lĩnh vực hẹp ở mức tương tự như một người chuyên gia
  + hầu hết các hệ chuyên gia là các hệ dựa “luật”
  + Hiện nay một số các hệ chuyên gia thành công trong các lĩnh vực : bán hàng,kỹ nghệ y khoa và khai khoáng (tìm mỏ : GIS), các hệ điện lực

Chú ý : phân biệt {tương đối ví dụ đông y được học trong điều trị y khoa (ES,H)}

So sánh : tri thức heuristic ⬄ ES = expert system

• Giống : Cơ sở tri thức

• Khác

H : số đông E : kinh nghiệm

H : Nhiều đt [đủ lớn + đủ phổ biến] E : 1 số ít -

H : Phổ thông E : gắn chuyên môn

* + - 1. **So sánh Tri thức hệ Heuristic với tri thức hệ chuyên gia (ES)**

\*Nhận xét:

- Phần mềm (hệ thống) liên quan yếu tố chuyên môn: ES

- Phần mềm (hệ thống) nhận xét, đánh giá, rút kinh nghiệm: Heuristic

### Logic mệnh đề và Logic vị từ

* + - 1. **Logic Mệnh đề**

a/ Mệnh đề: là một khẳng định có thể nhận giá trị Đúng/Sai (chân trị)

* Đa số mệnh đề: đúng sai tùy thuộc vào nhiều yếu tố ngoại cảnh như: theo thời điểm, theo chủ đề liên quan
  + VD: (1) “Hà Nội là thủ đô của Việt Nam”   
     (2) “Mặt trời mọc ở phương Đông”   
     (3) “Hai nhân hai bằng bốn”   
     (4) “Chạy 11 mét trong 1 giây là dễ chịu”
* Các bài toán AI chỉ xem xét mệnh đề 1 cách tương đối (xét chân trị mệnh đề) bằng ví dụ dựa và số đông cho là đúng.
* Mệnh đề: a, b, p, q,…
* Các phép kết nối mệnh đề:
  + Hội (và): ˄
  + Tuyển (hoặc): ˅
  + Phủ định (đảo): ̶ hoặc  hoặc ~
  + Kéo theo (suy ra): → (a→b = a ˅ b)
* Luật suy diễn:
  + A: đúng và A → B: đúng => B đúng modus ponens
  + A → B: đúng và B: sai => A: sai modus tollens
  + (a → b = a ˅ b)

T T T T T T

F F T F F T

T F F T F F

F T [F] F T [T]

* Mỗi sự kiện trong cơ sở tri thức với logic mệnh đề được biểu diễn bởi một số hữu hạn các mệnh đề với các phép kết nối trên.
  + Các biến: x, y, z, u,…
  + Các hàm: F, G, H, P, Q,…

c/ Logic vị từ: mở rộng logic mệnh đề bằng cách dựa vào khái niệm vị từ và các lượng từ.

* Một số lượng từ:
  + Phổ dụng ()
  + Tồn tại () hoặc (!)
* Vị từ là hàm của biến || tham số là “biến mệnh đề”
  + VD: NAM (x) = người x có Nam giới hay không? NAM là vị từ.

LOVE (x,y) = x và y Yêu nhau? LOVE là vị từ {nếu x và y thực sự yêu nhau => Vị từ LOVE (x,y) có chân trị TRUE}

* + VD1: Vị từ P(x): số chẵn; với x = “Số Nguyên”

Tùy vào x là số nào mà P(x) có thể đúng/sai

* + VD2: Biểu diễn phát biểu: “Chẳng có vật nào lớn nhất và cũng chẳng có vật nào nhỏ nhất” Biểu diễn: quy ước kí hiệu như sau
    - P(y,x): “y lớn hơn x”
    - Q(z,x): “z bé hơn x”
    - Câu phát biểu trên: (x y: P(y,x)) ˄ (x z: Q(z,x))

=>Logic vị từ: biểu diễn gần như toàn bộ các khái niệm và nguyên lý của các khoa học cơ bản.

* + - 1. **Logic vị từ**
         1. **Vị từ và lượng từ**
         2. **Biểu diễn Logic vị từ và chân trị của biểu diễn tri thức vị từ**

##### Mở rộng Logic vị từ

##### Logic vị từ bậc||cấp cao

VD: P(x,y) -> Q (P1(x), P2(y): vị từ bậc/cấp 2

P1(x) và P2(y): vị từ bậc/cấp 1

\*Vị từ cấp 1 (FOP: First Order Predicate) = Vị từ sơ cấp

\*Vị từ cấp 2 (SOP: Second Order Predicate) = Vị từ thứ cấp

VD1:

P1 = kết hôn =>FOP x = nam ? y = nữ ? P1(x,y)

P2 = bà con (cùng huyết thống) => FOP x, y = người P2(x,y)

Q = phạm luật => SOP Q(P1(x,y), P2(x,y))

VD2:

P1 = kết hôn =>FOP x = nam ? y = nữ ? P1(x,y)

P2 = đã có/lập gia đình x = người ? P2(x)

Q = phạm luật => SOP Q(P1(x,y), P2(x))

Ghi chú:

+ Với các tri thức dạng “luật” (Rule) có thể biểu diễn toàn bộ Rule đó dưới dạng FOP (vị từ cấp 1 = lượng từ [, , !], vị từ [MAN(…), PASS(…),…], phép nối [˄, ˅, ~]) hoặc cũng có thể biểu diễn dạng SOP (vị từ cấp 2) hoặc cấp cao hơn.

+ Với các tri thức dạng “luật” (Rule) có thể biểu diễn FOP với nhiều kết quả biểu diễn khác nhau

##### Logic tình huống

Logic vị từ có bổ sung thêm các biến tình huống.

VD: Thông tin (Input)

“Tàu XYZ cập bến lúc 11g”

“Bốc dỡ hàng kết thúc lúc 18g”

“Anh A chỉ huy bốc dỡ hàng”

“Tàu XYZ rời cảng lúc 19g”

Hỏi (Goal): “17 giờ A đang ở đâu ?” => Trả lời: A ở “bến cảng”

PC (Soft truyền thông) => không biết ???  người: có thể biết => PC (Soft AI: Logic vị từ): biết

11h -> 18h -> 19h 11h -> 17h -> 18h

Lập Phương pháp: sao cho Input [người sử dụng nhập vào câu phát biểu] = Output [ Logic tình huống: cài phát biểu vào CSTT]

biến tình huống = biến thời gian = lượng hóa = biểu diễn định lượng

Kết luận: Cài đặt bằng Logic vị từ dạng “tình huống” với biến tình huống như trên

##### Logic trạng thái (modal)

Liên quan Fuzzy logic (Logic “mờ”)

VD: Wiki: “Anh ABC ở phòng khách và phòng bếp” = “Mờ” Với phát biểu trên có 3 trạng thái xảy ra:

-Anh ABC ở trong phòng khách (không liên quan phòng bếp)

-Anh ABC ở trong phòng bếp (không liên quan phòng khách)

-Anh ABC ở cửa nối 2 phòng bếp và khách = Logic trạng thái

##### Logic xác suất và Logic khả xuất

Liên quan Fuzzy logic (Logic “mờ”):

VD: tiếp VD trên 2.2.6.3

Anh ABC ở giữa nối 2 phòng bếp và phòng khách = Logic trạng thái

Trên thực tế: ABC sẽ có 30% cơ thể “lọt” (ở bên) phòng khách

ABC sẽ có 40% cơ thể “lọt” (ở bên) phòng bếp

ACB sẽ có phần còn lại (30%) cơ thể ở giữa cửa nối 2 phòng

=> Logic xác suất (ướm tương đối -> thực tế KHÔNG đo lường mặc dù có phần mềm SMART)

Logic xác suất là Logic diễn đạt một sự kiện, có kèm theo tỷ lệ xác suất xảy ra là bao nhiêu phần trăm.

VD: ứng dụng trong nghiệp vụ điều tra.

Logic khả xuất là trường hợp cá biệt (đặc biệt) của Logic xác suất có nghĩa là thay vì “xác

suất” thông thường thì xét “xác suất xảy ra” là bao nhiêu phần trăm xảy ra sự kiện đó.

Thường biểu diễn

VD: p -> 0.8q (độ tin cậy 0.8): 80% xảy ra

Tức là, khi có sự kiện p thì 80% xảy ra sự kiện q

VD cụ thể: “Trời nhiều mây thì 80% chiều nay mưa” là Logic vị từ (Logic vị từ dạng khả suất) “Mờ” là không rõ “nhiều mây như thế nào”

=>minh họa một cách giải quyết: chụp ảnh bầu trời => phân tích xử lý ảnh

=>xác định tỷ lệ % mây trên bầu trời thông qua hình ảnh (Biểu diễn Logic vị từ dạng Logic xác suất)

=>xác định “ngưỡng” (dưới = tối thiểu)[%mây] = “nhiều mây

### Biểu diễn tri thức bằng Frame (Cấu trúc = "Khung")

* Frame là một phương thức biểu diễn tri thức có cấu trúc dữ liệu, chứa tất cả tri thức liên quan đến một đối tượng cụ thể nào đó bằng hình thức biểu dinễ tri thức dựa trên cơ sở lập trình truyền thống (dung cấu trúc dữ liệu = định nghĩa Agent = Token)

### Suy diễn tri thức bằng luật dẫn xuất

#### Khái niệm

\*Suy diễn hay còn gọi là lập luận hoặc suy luận là dựa trên cơ sở tri thức ban đầu (đã có

hoặc đã biết) gồm có tri thức cơ sở và luật (Rules) dẫn xuất (tập luật) .

=>Xác định (suy diễn ra hoặc lập luận để xác định ra các “sự kiện mới” bộ sung vào cơ

sở tri thức

+ “Sự kiện mới” là dạng logic “vị từ” [chân trị] hay tri thức “dẫn xuất” hoặc “suy luận” hoặc “thứ cấp”.

\*Nền tảng luật (quy tắc) suy diễn:

+Suy diễn tiến là cho các tiền đề sau đó xác định và suy ra kết quả có được từ các tiền đề đó.

+Suy diễn lùi (ngược) là cho (đặt ra) kết quả cần đạt sau đó xác định và suy ra các tiền đề

cần thiết để có kết quả đó.

\*VD1: Xét cơ sở tri thức : Pháp luật

.Người dân đặt ra yêu cầu nào đó dựa vào pháp luật sau đó xác định điều kiện  suy diễn ngược .Nêu được các điều kiện để người dân được phép thực hiện hoạt động nào đó  suy diễn tiến.

.Để thành lập doanh nghiệp thí chủ doanh nghiệp phải đảm bảo những điều kiện và thủ tục gì?

.Giả sử chủ phải có 50 triệu vốn và cơ sở mặt bằng quy mô tương đối (500 m2 trở lên).

VD2: Cơ sở tri thức: Từ điển Anh – Việt

+Câu từ tiếng Anh => tiếng Việt

+Câu từ tiếng Việt => tiếng Anh

\*Các biểu diễn cơ sở tri thức: phải đảm bảo dùng được cho cả 2 dạng suy diễn (nêu trên).

#### Luật dẫn xuất trong Cơ sở tri thức: Suy diễn tiến

\*Là các qui tắc hoặc quy luật làm cơ sở để “suy diễn tri thức” thường là dạng biểu diễn hình thức hóa tri thức tự nhiên bằng các kí hiệu (toán) được lưu trữ trên máy tính phục vụ hệ thống phần mềm AI.

\*Phổ biến là luật dẫn xuất “kéo theo” dạng tam đoạn luận (logic học):

Cho một cơ sở tri thức biểu diễn (giả thiết hoặc tiền đề) => kết luận (conclusion)

\*Cơ sở tri thức dùng luật dẫn xuất để suy luận ra các kết luận bổ sung các sự kiện vào tập sự kiện biểu diễn (tri thức)

VD1:Xét cá sự kiện:

A: Sinh viên

B: Điểm trung bình >= 8

D: Sinh viên giỏi

C: Rèn luyện giỏi

E: Khen thưởng

Giả sử có các luật dẫn xuất sau:

A ˄ B -> D (A[Sinh viên] ˄ B[điểm trung bình] -> D[Sinh viên giỏi] = “Sinh viên có điểm trung bình từ 8 trở lên là sinh viên giỏi”)

D ˄ C -> E (D[Sinh viên giỏi] ˄ C[Rèn luyện giỏi] -> D[khen thưởng] = “Sinh viên giỏi và có rèn luyện giỏi thì được khen thưởng”)

Hỏi: A, B, C ? E => kiểm tra chân trị của vị từ suy diễn: A ˄ B ˄ C -> E là chân trị đúng

VD2: Cho

+Sự kiện: H, K

+Tập dẫn xuất:

(R1): A -> E

(R2): B -> D

(R3): H -> A

(R4): E ˄ G -> C

(R5): E ˄ K -> B

(R6): D ˄ E ˄ K -> C

(R1): G ˄ K ˄ F -> A

Yêu cầu suy luận: tìm tất cả các sự kiện có thể có được từ {H, K} là “bao đóng” của tập {H, K}?

Kí hiệu {H, K}+ hoặc nếu kí hiệu P = {H, K} thì P+ là bao đóng của tập P

(R3): H -> A {A, H, K}

(R1): A -> E {A, E, H, K}

(R5): E ˄ K -> B {A, B, E, H, K}

(R2): B -> D {A, B, D, E, H, K}

(R6): D ˄ E ˄ K -> C {A, B, C, D, E, H, K}

Kết quả = {A, B, C, D, E, H, K} = {H, K}+ là bao đóng của tập {H, K}

### Biểu diễn suy luận bằng đồ thị AND/OR

\*Dùng để suy luận lùi

\*Là dạng đồ thị có hướng AND OR, trong đó:

+Nút: sự kiện

+Cung/cạnh: suy luận 1 chiều sự kiện này dẫn đến sự kiện khác (tiến)

+Cung/cạnh từ 1 nút biểu diễn choc các quan hệ AND || OR (VÀ HOẶC) -> dẫn

xuất điều kiện

+Gốc là sự kiện mục tiêu

 Mục đích: tím tất cả sự kiện dẫn xuất, sự kiện mục tiêu Minh họa thực tế cho đồ thị AND/OR nêu trên:

Xét quy ước kí hiệu cho sự kiện

Q=’’Trúng tuyển vào trường Đại học”

P1=”điêm thi môn 1 của kết quả thi TN THPTQG là >=5”

P2=”điểm thi môn 2 của kết quả thi TN THPTQG là >=5”

P3=”điểm thi môn 3 của kết quả thi TN THPTQG là >=5”

S1=”Kết quả học tập của 3 năm THPT AVG(ALL)>=6”

S2=”hạnh kiểm đạt loại Tốt trở lên”

Có luật sau (luật dẫn xuất sau là ĐÚNG)

Q=(P1^P2^P3) hoặc (S1^S2)

“thí sinh trúng tuyển vào Đại học” nếu “điểm thi của mỗi môn trong 3 môn thi THPTQG

từ 5 điểm trở lên” hoặc “ kết quả bình quân các môn 3 năm học THPT từ 6 trở lên VÀ

hạnh kiểm đạt từ tốt trở lên”

Vẽ đồ thị AND/OR để xác định các điều kiện để có thể “trúng tuyển vào đại học” = Q

(suy diễn ngược)

### Các biểu diễn tương đương trong Logic mệnh đề và Logic vị từ

### Xử lý ngôn ngữ tự nhiên (Natural Language Processing = NLP): Voice Assistant

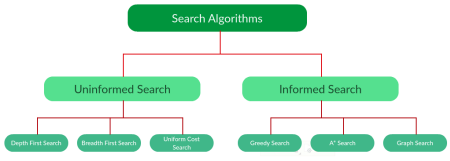
## Các phương pháp tìm kiếm lời giải của bài toán AI

### Giới thiệu

* + - 1. **Các tiêu chí đánh giá**
* Tính hoàn thành: chiến lược có bảo đảm tìm thấy giải pháp cho bài toán.
* Độ phức tạp thời gian: chiến lược mất bao lâu để tìm ra một giải pháp?
* Độ phức tạp không gian (dung lượng bộ nhớ): chiến lược đó cần bao nhiêu dung lượng bộ nhớ để thực hiện việc tìm kiếm.
* Tính tối ưu: chiến lược có tìm được giải pháp có chất lượng cao nhất khi có một số các  giải pháp khác nhau ?
  + - 1. **Phân loại các phương pháp (chiến lược) tìm kiếm lời giải**

Có nhiều cách phân loại các chiến lược tìm kiếm, phổ biến là chia làm 2 loại:

* Các phương pháp tìm kiếm mù (Blind Search) = tìm kiếm không có thêm thông tin  (uninformed search)
* Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) = tìm kiếm với thông tin  bổ sung (Informed Search) = tìm kiếm rõ

 [DFS, BFS, UCS, IDS, DLS,…] [G\_BestFS, A\* & SMA\*= Graph Search, HCS. .]  Đã Thực hành cài đặt: DFS BFS Greedy\_BestFS

A\* và SMA\* (Simplified Memory bounded A\*): tìm kiếm trong đồ thị (Graph Search) Chú ý: **Phương pháp** tìm kiếm … = còn gọi là **Chiến lược** tìm kiếm ….

### Các phương pháp tìm kiếm mù (Blind Search) = tìm kiếm không có thêm thông tin (uninformed search)

* + - 1. **Depth-First Search (DFS): Tìm kiếm theo chiều sâu**

Là giải thuật để duyệt cấu trúc cây [đồ thị] không gian trạng thái để tìm lời giải của bài toán AI, trong  đó nó lần lượt tìm theo chiều sâu của từng nhánh cây [đồ thị] trước, cho đến khi KHÔNG tìm được lời  giải [đáp án] trên nhánh đó [còn gọi là “bế tắc” || “bí” || hết đường || cùng đường] thì nó quay lui  [back\_tracking] để tìm trên nhánh kế tiếp: cho đến khi:

* Tìm được lời giải [đáp án] OR
* Duyệt hết toàn bộ (OR giới hạn phạm vi) không gian cây [đồ thị] được xem xét mà vẫn không tim  được lời giải [đáp án]

**Mô tả DFS:**

* Tìm kiếm theo chiều sâu luôn luôn mở rộng một trong các nút ở mức sâu nhất của cây.
* Chỉ khi phép tìm kiếm đi tới một "điểm cụt" : "nút lá" (một nút không phải đích mà không có nút con), việc  tìm kiếm sẽ quay lại và xét những nút nông/cạn hơn: BACK TRACKING
* Giải pháp này có thể được thực hiện bởi 1 phép tìm kiếm với sự hỗ trợ of một stack: các trạng thái mới  được sinh ra đc push vào đỉnh stack (thông thường: phải -> trái)

**Nhận xét DFS:**

Do nút được xét là sâu nhất, các nút kế tiếp của nó sẽ sâu hơn và khi đó sẽ trở thành sâu nhất.

=> Phép tìm kiếm theo chiều sâu yêu cầu dung lượng bộ nhớ rất thấp / nhanh / rủi ro

=> Tìm kiếm: nó chỉ cần phải lưu một đường duy nhất từ gốc tới nút lá, cùng với các nút anh em với các nút trên đường đi  chưa được mở rộng còn lại.

=> Đối với một không gian trạng thái với hệ số rẽ nhánh b và độ sâu tối đa m, phép tìm kiếm theo nhiều sâu chỉ yêu cầu lưu  trữ b\*m nút.

 Độ phức tạp thời gian của phép tìm kiếm sâu là O(b\*m).

=> Đối với những vấn đề (bài toán) mà có rất nhiều giải pháp (đáp số), phép tìm kiếm sâu có thể nhanh hơn tìm kiếm rộng, bởi vì nó có một  cơ hội tốt tìm ra một giải pháp chỉ sau khi khám phá một phần nhỏ của toàn bộ không gian.

=> Mặt hạn chế của phép tìm kiếm sâu là nó có thể bị tắc khi đi theo một đường sai: rủi ro lớn (bị “chìm”)

Rất nhiều bài toán có các cây tìm kiếm rất sâu, thậm chí vô hạn, vì vậy tìm kiếm sâu có thể sẽ không quay lại  được (ko backtrack được) NẾU sự lựa chọn sai (chọn nhầm nhánh cây quá “sâu”) = Phép tìm kiếm sẽ luôn luôn  tiếp tục đi xuống mà không quay trở lại, thậm chí trong khi có một giải pháp ở mức rất nông tồn tại ở các nhánh  khác. Như vậy đối với những bài toán này, phép tìm kiếm sâu sẽ bị sa lầy trong một vòng lặp vô hạn hoặc không  bao giờ đưa ra một giải pháp, hoặc là cuối cùng nó có thể đưa ra một đường đi giải pháp ko tốt so với phương án  tối ưu = RỦI RO DFS rất cao.

* + - 1. **Breadth-First Search (BFS): Tìm kiếm theo chiều rộng || ngang**

Là giải thuật để duyệt cấu trúc cây [đồ thị] không gian trạng thái để tìm lời giải của bài toán AI, trong  đó nó lần lượt tìm theo chiều ngang [rộng] của từng mức [cấp] cây [đồ thị] trước, cho đến khi KHÔNG  tìm được lời giải [đáp án] trên mức [cấp] đó [còn gọi là “bế tắc” || “bí” || hết đường || cùng đường] thì  nó quay lui [back\_tracking] để tìm trên mức [cấp] kế tiếp: cho đến khi:

* Tìm được lời giải [đáp án] OR
* Duyệt hết toàn bộ (OR giới hạn phạm vi) không gian cây [đồ thị] được xem xét mà vẫn không tìm  được lời giải [đáp án]

**MÔ TẢ BFS**

Trong chiến lược BFS này,

* nút gốc được xét trước tiên,
* sau đó đến lượt tất cả các nút con trực tiếp của nút gốc: được xét

- tiếp đến là những nút con kế tiếp của chúng và cứ lặp lại như vậy

=> cho đến khi tìm ra nút đích (hoặc hết phần không gian trạng thái mà ko tìm thấy lời giải) **\* Nhận xét BFS**

- Tất cả các nút ở độ sâu d trên cây tìm kiếm được xét trước các nút ở độ sâu d+1.

Tìm kiếm theo chiều rộng có thể được thực hiện bằng cách gọi giải thuật general-search (dưới) với một hàng đợi (queue =  FIFO = First In First Out = vào trước ra trước)

* Tìm kiếm theo chiều rộng là một chiến lược tìm kiếm lời giải có hệ thống bởi vì nó xem xét tất cả các đường đi có độ dài  bằng 1 trước, sau đó đến tất cả những đường đi có độ dài bằng 2, và lặp lại như vậy => nên :
  + tìm kiếm theo chiều rộng sẽ luôn tìm ra trạng thái đích cạn/nông nhất trước tiên.
  + ìm kiếm theo chiều rộng: trạng thái đích tìm đc là trạng thái có chi phí tối ưu.

 (nếu chi phí = một hàm tăng của độ sâu các nút).

kết luận: BFS phù hợp các bài toán:

* cây không gian trạng thái quá sâu; ko có nút "lá"
* số lượng nút con của mỗi nút trên cây: có giới hạn
  + - 1. **Uniform Cost Search (UCS): Tìm kiếm với chi phí ĐỀU (không thay đổi) = chi phí cực tiểu [Áp dụng thuật toán Dijkstra]**

Thuật toán UCS là một thuật toán duyệt, tìm kiếm trên một cấu trúc cây, hoặc đồ thị có trọng số (chi phí). Việc tìm kiếm bắt đầu tại nút gốc và tiếp tục bằng cách duyệt các nút tiếp theo với trọng số hay chi phí thấp nhất tính từ nút gốc.

* + - 1. **Iterative Deepening Search (IDS) = Tìm kiếm sâu dần = cải tiến DFS**

Trong khoa học máy tính, tìm kiếm theo chiều sâu lặp đi lặp lại hoặc cụ thể hơn là tìm kiếm theo chiều sâu lặp đi lặp lại (IDS hoặc IDDFS) là một chiến lược tìm kiếm không gian / đồ thị trạng thái trong đó phiên bản tìm kiếm theo chiều sâu giới hạn độ sâu được chạy lặp lại với độ sâu ngày càng tăng giới hạn cho đến khi mục tiêu được tìm thấy. IDDFS tối ưu giống như tìm kiếm theo chiều rộng, nhưng sử dụng ít bộ nhớ hơn nhiều; ở mỗi lần lặp lại, nó truy cập các nút trong cây tìm kiếm theo thứ tự giống như tìm kiếm theo chiều sâu, nhưng thứ tự tích lũy trong đó các nút được truy cập đầu tiên là theo chiều rộng thực sự

* + - 1. **Depth-Limited Search (DLS) = Tìm kiếm giới hạn độ sâu = cải tiến DFS**

Tìm kiếm giới hạn theo độ sâu là thuật toán tìm kiếm mới dành cho tìm kiếm không được thông tin. Vấn đề cây không giới hạn xảy ra xuất hiện trong thuật toán tìm kiếm theo độ sâu và nó có thể được khắc phục bằng cách áp đặt ranh giới hoặc giới hạn cho độ sâu của miền tìm kiếm. Làm cho chiến lược tìm kiếm DFS trở nên tinh tế hơn và được tổ chức thành một vòng lặp hữu hạn. Do đó, tìm kiếm giới hạn Độ sâu có thể được gọi là phiên bản mở rộng và tinh chỉnh của thuật toán DFS.

Tóm lại, chúng ta có thể nói rằng để tránh tình trạng vòng lặp vô hạn trong khi thực thi các mã, và thuật toán tìm kiếm giới hạn độ sâu đang được thực thi thành một tập hợp độ sâu hữu hạn gọi là độ sâu giới hạn.

### Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) = tìm kiếm với thông tin bổ sung (Informed Search) = Cải tiến BFS

* Heuristic: là các kỹ thuật dựa trên kinh nghiệm để giải quyết vấn đề, nhằm đưa ra một giải  pháp mà không được đảm bảo là tối ưu
* Heuristic function: Hàm đánh giá dựa trên kinh nghiệm, dựa vào đó để xếp hạng thứ tự tìm kiếm, cách chọn hàm đánh giá quyết định nhiều đến kết quả tìm kiếm.
* Một số chiến lược tìm kiếm phổ biến
  + thông thường là trọng số (hệ số đánh giá mức độ quan trọng), hàm chi phí (Cost). Chiến lược tìm kiếm Best first: Best BFS (Greedy BFS)
  + Chiến lược tìm kiếm A\* Search
  + Chiến lược tìm kiếm tối ưu cục bộ như leo đồi||núi (Hill-Climbing Search)   = Cải tiến BFS (tương tự như sâu dần của DFS)
  + Chiến lược tìm kiếm đối kháng = (Board Game) dùng giải thuật MiniMax || MinMax . . .
    - 1. **Best first: BestBFS (Greedy\_BFS)**

Trong trí tuệ nhân tạo, thuật toán greedy best first search là một chiến lược tìm kiếm với tri thức bổ sung từ việc sử dụng các tri thức cụ thể của bài toán. Thuật toán sẽ sử dụng 1 hàm đánh giá là hàm heuristic h(n). hàm heuristic h(n) này sẽ đánh giá chi phí để đi từ nút hiện tại n đến nút đích (mục tiêu).

* + - 1. **A\* và SMA\* (A\* với bộ nhớ giới hạn): tìm kiếm trên đồi thị (Graph Search)**

Trong khoa học máy tính, A\* (đọc là A sao) là thuật toán tìm kiếm trong đồ thị. Thuật toán này tìm một đường đi từ một nút khởi đầu tới một nút đích cho trước (hoặc tới một nút thỏa mãn một điều kiện đích). Thuật toán này sử dụng một "đánh giá heuristic" để xếp loại từng nút theo ước lượng về tuyến đường tốt nhất đi qua nút đó. Thuật toán này duyệt các nút theo thứ tự của đánh giá heuristic này. Do đó, thuật toán A\* là một ví dụ của tìm kiếm theo lựa chọn tốt nhất (best-first search).

* + - 1. **Hill-Climbing Search (Tìm kiếm tối ưu cục bộ || leo đồi|| leo núi) = HCS**

Trong khoa học máy tính, giải thuật Hill Climbing là một kỹ thuật tối ưu toán học thuộc họ tìm kiếm cục bộ. Nó thực hiện tìm một trạng thái tốt hơn trạng thái hiện tại để mở rộng. Để biết trạng thái tiếp theo nào là lớn hơn, nó dùng một hàm H để xác định trạng thái nào là tốt nhất.

Hill Climbing dễ dàng tìm thấy một giải pháp tốt cục bộ (local optimum) nhưng khó tìm thấy giải pháp tốt nhất (global optimum) trong tất cả các giải pháp được đưa ra (search space). Hill Climbing phù hợp để giải các bài toán “convex” (dịch tạm: lồi) như là tìm kiếm đơn giản (simplex programming) trong lập trình tuyến tính, tìm kiếm nhị phân.

Tính đơn đơn giản của giải thuật khiến nó trở thành lựa chọn đầu tiên trong số các giải thuật tối ưu. Nó được sử dụng rất nhiều trong trí tuệ nhân tạo, dùng cho mục đích đi đến trạng thái đích từ một node bắt đầu. Việc chọn node tiếp theo và node bắt đầu có thể thay đổi nhiều giải thuật khác nhau.

* + - 1. **Chiến lược tìm kiếm đối kháng (Board Game) với Giải thuật MiniMax||MinMax**

Minimax (còn gọi là mimax) là một phương pháp trong lý thuyết quyết định có mục đích là tối thiểu hóa (minimize) tổn thất vốn được dự tính có thể là "tối đa" (maximize). Có thể hiểu ngược lại là, nó nhằm tối đa hóa lợi ích vốn được dự tính là tối thiểu (maximin). Nó bắt nguồn từ trò chơi có tổng bằng không. Nó cũng được mở rộng cho nhiều trò chơi phức tạp hơn và giúp đưa ra các quyết định chung khi có sự hiện diện của sự không chắc chắn.

### Lập trình Logic (Logic Programming) phát triển cơ chế lập luận logic

# GIỚI THIỆU VỀ MÔ HÌNH BÀI TOÁN AI: TÌM ĐƯỜNG ĐI TRÊN BẢN ĐỒ BẰNG THUẬT TOÁN DIJKSTRA

## Mô tả bài toán AI

Bài toán toán tìm đường đi trong đề tài này chúng ta có thể xem bản đồ là một ma trận các pixel. Mỗi pixel có giá trị RGB là 0,0,0 (đen) hoặc 255,255,255 (trắng). Mục tiêu của bài toán là tìm ra một con đường ngắn nhất đi theo con đường màu trắng và không vi phạm các ranh giới màu đen.

## Các yếu tố xác định bài toán AI

Bài toán trí tuệ nhân tạo được xác định bởi 4 yếu tố sau đây :

* Tập hợp các trạng thái của bài toán : Trạng thái bản đồ gồm các đường đi , đường đó chưa đi lần nào, …
* Trạng thái bắt đầu : Là tọa độ tại một điểm nào đó trên bản đồ và điểm đó đó không nằm trong điểm bị giới hạn.
* Có mục tiêu đặt ra ( thông qua hàm mục tiêu ) mà kĩ thuật AI phải giải quyết cho bài toán đó .
  + Mục tiêu của bài toán là có Trạng thái kết thúc tức là trạng thái có điểm đầu và điểm cần đến thông với nhau (có đường đi đến) vói giá trị thấp nhất.
  + Còn nếu tất cả các đường đều được đi mà trạng thái chưa kết thúc thì không có đường đi
* Có cơ chế chuyển trạng thái: Sẽ tìm tất cả các đường đi có thể có của bài toán, sau cùng đường đi ngắn nhất sẽ được chọn

## Giới thiệu nguồn dữ liệu gốc dùng thực nghiệm bài toán AI

Là hình ảnh một bản đồ đơn giản ngẫu nhiên được lấy trên internet.

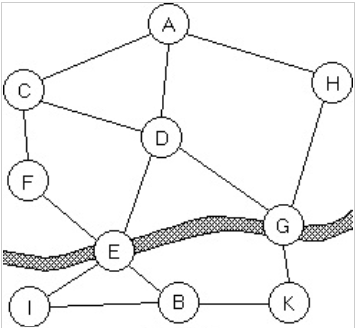
# BIỂU DIỄN BÀI TOÁN AI VÀ CƠ SỞ TRI THỨC: TÌM ĐƯỜNG ĐI TRÊN BẢN ĐỒ BẰNG THUẬT TOÁN DIJKSTRA

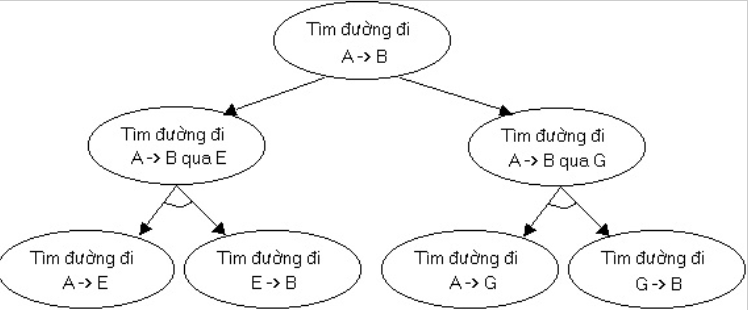
## Giới thiệu

Trong phần này sẽ trình bày về một trong những ứng dụng của lý thuyết đồ thị vào việc tìm đường đi trên bản đồ. Hình ảnh bản đồ sẽ được mô hình hóa thành một đồ thị có trọng số để áp dụng thuật toán vào việc tìm kiếm đường đi tối ưu.

Bài toán bắt đầu bằng ở vị trí nguồn và kết thúc sau khi tìm được thấy vị trí đích bằng đường đi tối ưu nhất. Sau khi đi qua từng nút, cuối cùng chúng ta sẽ có một biểu đồ hiển thị độ dài đường đi ngắn nhất từ ​​nguồn đến mọi nút.

## Biểu diễn bài toán AI





# PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI CHO BÀI TOÁN AI: TÌM ĐƯỜNG ĐI NGẮN NHẤT TRÊN BẢN ĐỒ BẰNG GIẢI THUẬT DIJKSTRA

## Giới thiệu

Trong đề tài này có rất nhiều phương để giải quyết bài toán toán tìm đường, nhưng cần phải xem xét phương pháp giải thuật phù hợp nhất.

**Các tiêu chí đánh giá các chiến lược tìm kiếm:**

* Tính hoàn thành: chiến lược phải bảo đảm tìm thấy giải pháp.
* Độ phức tạp thời gian: chiến lược mất thời gian chấp nhận được để tìm ra một giải pháp
* Độ phức tạp không gian (dung lượng bộ nhớ): Độ phức tạp vừa phải.
* Tính tối ưu: chiến lược tìm được giải pháp có chất lượng cao nhất khi có một số các  giải pháp khác nhau

**PHÂN LOẠI các chiến lược tìm kiếm:**

* Có nhiều cách phân loại các chiến lược tìm kiếm, phổ biến là chia làm 2 loại:
  + Các phương pháp tìm kiếm mù (Blind Search) hay tìm kiếm không có thêm thông tin  (uninformed search)
  + Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) hay tìm kiếm với thông tin  bổ sung (Informed Search)
  + A\* và SMA\* (Simplified Memory bounded A\*): tìm kiếm trong đồ thị (Graph Search)

**Các phương pháp tìm kiếm mù (Blind Search) – tìm kiếm không có thêm thông tin (uninformed search)**

* Còn được gọi là tìm kiếm mù/mờ (blind search), không có đủ thông tin cần thiết:
  + không biết được sẽ qua bao nhiêu bước để có kết quả
  + hay không biết cần "chi phí" bao nhiêu để đi từ trạng thái hiện tại cho tới đích, chủ yếu dựa vào nguyên tắc:  phân biệt một trạng thái đích với một trạng thái không phải là trạng thái đích.
* Một số chiến lược tìm kiếm phổ biến
  + DFS (Depth First Search): Tìm kiếm theo chiều sâu
  + BFS (Breadth First Search): Tìm kiếm theo chiều rộng
  + UCS (Uniform Cost Search): Tìm kiếm với chi phí tối thiểu (Áp dụng thuật toán Dijkstra)
  + IDS (Iterative Deepening Search) hay DWS [DepthWise Search]: Tìm kiếm sâu dần là một cải tiến của DFS
  + DLS (Depth-Limited Search): Tìm kiếm giới hạn độ sâu là cải tiến DFS nhưng đặt một giới tìm kiếm
* Với Blind Search, cây không gian trạng thái ko biết trước và sẽ mở rộng dần theo chiến lược tìm kiếm
* DFS và BFS là 2 phương pháp tìm kiếm cơ bản, các phương pháp tìm kiếm khác đều được phát triển (tốt hơn) dựa  trên cơ sở 2 phương pháp này.

**Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) – tìm kiếm với  thông tin bổ sung (Informed Search)**

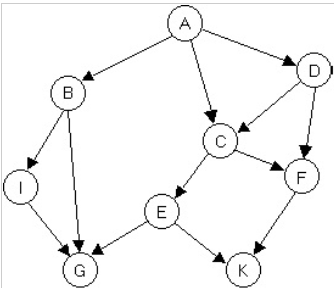
**Xét 2 giải thuật có sử dụng hàm đánh giá (chi phí)**:

* G-BFS hiệu quả nhưng chưa chắc thật sự tối ưu, vì chỉ chọn những đường đi tốt nhìn thấy được trước mắt
* Heuristic A\* Search= UCS + Greedy Best-First-Search
  + Kết hợp tính hiệu quả (nhanh) của G-BFS và tính tối ưu (luôn tìm ra lời giải đúng) của UCS (Uniform Cost Search)
  + Hàm đánh giá: f(n)= g(n) + h(n)
    - g(n) – Chi phí từ node hiện tại tới node n (tính hàm g)
    - h(n) – ước lượng khoảng cách min từ node n – đích
  + UCS chỉ dùng g(n) nên chậm, G-BFS chỉ dựa h(n) nên không tối ưu  ⬄ f(n) = tối ưu + hiệu quả
  + h(n), g(n) dùng độ đo Euclidean distance (Đường thẳng):

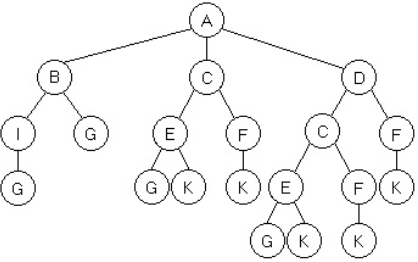
⬄ d(A[xa, ya], B[xb, yb]) = **Sqrt[(xa-xb)2 + (ya-yb)2]**

## Mô tả sơ bộ về mô hình cây không gian trạng thái của bài toán AI

*Hình ảnh chỉ có thể mô tả được phần nào của bài toán, vì bản đồ thực tế rất lớn rất lớn có thể có rất nhiều nút không thể mô tả hết được*



ĐỒ THỊ MÔ TẢ KHÔNG GIAN TRẠNG THÁI



CÂY TÌM KIẾM TƯƠNG ỨNG

## Xác định phương pháp tìm kiếm lời giải cho bài toán AI: Tìm đường đi ngắn nhất trên bản đồ

Sau khi So sánh các chiến lược tìm kiếm sao cho phù hợp với bài toán đề tài, em rút ra được một số nhận xét như sau:

* Đối với phương pháp tìm kiếm mù phương pháp tìm kiếm phù hợp nhất có lẽ là UCS (áp dụng thuật toán dijkstra) giúp tối ưu chi phí nhất đối với đề tài này là đường đi ngắn nhất.
* Tương ứng với đối với các phương pháp tìm kiếm với thông tin bổ sung, để tối ưu bài toán ta có thể kết hợp thêm thuật toán Greedy tăng tốc độ giải thuật (phần phụ) thành thuật toán A\* cho kết quả nhanh hơn.
* Nhưng với tập dữ liệu nhỏ (hình ảnh có kích thước nhỏ), nên trong đề tài này sẽ chỉ sử dụng thuật toán UCS (dijkstra) để đơn giản bài toán nhưng vẫn cho kết quả tốt nhất có thể.

# GIỚI THIỆU SẢN PHẨM CỦA ĐỀ TÀI AI: TÌM ĐƯỜNG ĐI NGẮN NHẤT TRÊN BẢN ĐỒ BẰNG THUẬT TOÁN DIJKSTRA

## Giới thiệu sản phẩm bài toán AI: Tìm đường đi ngắn nhất trên bản đồ bằng giải thuật dijkstra

Thông tin sản phẩm

* Tên gọi : Tìm đường đi ngắn trên bản đồ
* Được tạo nên từ công cụ jupyter (anaconda3) bằng ngôn ngữ python với các thư viện như matplotlib, numpy và opencv để thực hiện giải thuật dijkstra cũng như biểu diễn trực quan bài toán
* Nền tảng công nghệ: nhận diện hình ảnh

Full codes:

import cv2

import matplotlib.pyplot as plt

import numpy as np

#Hàm và lớp trợ giúp

class Vertex:

def \_\_init\_\_(self,x\_coord,y\_coord):

self.x=x\_coord

self.y=y\_coord

self.d=float('inf') #khoảng cách hiện tại từ node nguồn

self.parent\_x=None

self.parent\_y=None

self.processed=False

self.index\_in\_queue=None

#Trả giá trị neighbor về trên, dưới, trái, phải

def get\_neighbors(mat,r,c):

shape=mat.shape

neighbors=[]

#Chắc chắn rằng neighbors nằm trong ranh giới bản đồ

if r > 0 and not mat[r-1][c].processed:

neighbors.aphương phápend(mat[r-1][c])

if r < shape[0] - 1 and not mat[r+1][c].processed:

neighbors.aphương phápend(mat[r+1][c])

if c > 0 and not mat[r][c-1].processed:

neighbors.aphương phápend(mat[r][c-1])

if c < shape[1] - 1 and not mat[r][c+1].processed:

neighbors.aphương phápend(mat[r][c+1])

return neighbors

def bubble\_up(queue, index):

if index <= 0:

return queue

p\_index=(index-1)//2

if queue[index].d < queue[p\_index].d:

queue[index], queue[p\_index]=queue[p\_index], queue[index]

queue[index].index\_in\_queue=index

queue[p\_index].index\_in\_queue=p\_index

quque = bubble\_up(queue, p\_index)

return queue

def bubble\_down(queue, index):

length=len(queue)

lc\_index=2\*index+1

rc\_index=lc\_index+1

if lc\_index >= length:

return queue

if lc\_index < length and rc\_index >= length: #just left child

if queue[index].d > queue[lc\_index].d:

queue[index], queue[lc\_index]=queue[lc\_index], queue[index]

queue[index].index\_in\_queue=index

queue[lc\_index].index\_in\_queue=lc\_index

queue = bubble\_down(queue, lc\_index)

else:

small = lc\_index

if queue[lc\_index].d > queue[rc\_index].d:

small = rc\_index

if queue[small].d < queue[index].d:

queue[index],queue[small]=queue[small],queue[index]

queue[index].index\_in\_queue=index

queue[small].index\_in\_queue=small

queue = bubble\_down(queue, small)

return queue

#Lấy khoảng cách từ ảnh của tập dữ liệu thực nghiệm bản đồ

def get\_distance(img,u,v):

'''Thực hiện công thức khoảng cách bình phương euclide

=> làm cho các vị trí không phải đường đi (lề, nhà, sông, không thuộc bản đồ...)

có khoảng cách vô hạn không đi được '''

return 0.1 + (float(img[v][0])-float(img[u][0]))\*\*2+(float(img[v][1])-float(img[u][1]))\*\*2+(float(img[v][2])-float(img[u][2]))\*\*2

#Vẽ đường đi từ điểm bắt đầu

def drawPath(img,path, thickness=2):

#đường đi là danh sách các bộ giá trị (x, y)

x0,y0=path[0]

for vertex in path[1:]:

x1,y1=vertex

cv2.line(img,(x0,y0),(x1,y1),(255,0,0),thickness)

x0,y0=vertex

#Tìm đường đi ngắn nhất

def find\_shortest\_path(img,src,dst):

pq=[]

source\_x=src[0]

source\_y=src[1]

dest\_x=dst[0]

dest\_y=dst[1]

imagerows,imagecols=img.shape[0],img.shape[1]

matrix = np.full((imagerows, imagecols), None) #ma trận hóa hình ảnh thành ma trận[dòng][cột]

#Điền ma trận vào các đỉnh

for r in range(imagerows):

for c in range(imagecols):

matrix[r][c]=Vertex(c,r)

matrix[r][c].index\_in\_queue=len(pq)

pq.aphương phápend(matrix[r][c])

#đặt giá trị khoảng cách source = 0

matrix[source\_y][source\_x].d=0

pq=bubble\_up(pq, matrix[source\_y][source\_x].index\_in\_queue)

while len(pq) > 0:

u=pq[0]

u.processed=True #node chưa xử lý có giá trị nhỏ nhất

#xóa node khỏi hàng đợi

pq[0]=pq[-1]

pq[0].index\_in\_queue=0

pq.pop()

pq=bubble\_down(pq,0)

neighbors = get\_neighbors(matrix,u.y,u.x)

for v in neighbors:

dist=get\_distance(img,(u.y,u.x),(v.y,v.x))

if u.d + dist < v.d:

v.d = u.d+dist

v.parent\_x=u.x #Theo vết đường đi

v.parent\_y=u.y

idx=v.index\_in\_queue

pq=bubble\_down(pq,idx)

pq=bubble\_up(pq,idx)

path=[]

iter\_v=matrix[dest\_y][dest\_x]

path.aphương phápend((dest\_x,dest\_y))

while(iter\_v.y!=source\_y or iter\_v.x!=source\_x):

path.aphương phápend((iter\_v.x,iter\_v.y))

iter\_v=matrix[iter\_v.parent\_y][iter\_v.parent\_x]

path.aphương phápend((source\_x,source\_y))

return path

img = cv2.imread('map.jpg') # Đọc ảnh từ tập dữ liệu bằng opencv

cv2.circle(img,(378,40), 3, (255,0,0), -1) # Thêm tọa độ cho điểm bắt đầu

cv2.circle(img, (1810,500), 3, (0,0,255), -1) # Thêm tọa độ cho điểm đến

plt.figure(figsize=(100,100))

plt.imshow(img)

plt.show()

img = cv2.imread('map.jpg') # Đọc ảnh từ tập dữ liệu bằng opencv

cv2.imwrite('map-initial.png', img)

p = find\_shortest\_path(img, (378,40), (1810,500))# tìm đường đi ngắn nhất từ 2 tọa độ cho trước

drawPath(img,p)# vẽ đường đi

cv2.imwrite('map-solution.png', img)

plt.figure(figsize=(10,10))

plt.imshow(img)

plt.show()

## Kết quả thực nghiệm sản phẩm bài toán AI: Tìm đường đi ngắn nhất trên bản đồ bằng giải thuật dijkstra







## Nhận xét và đánh giá về sản phẩm bài toán AI: Tìm đường đi ngắn nhất trên bản đồ bằng giải thuật dijkstra

Sản phẩm đã có thể đáp ứng được phần nào việc tìm kiếm đường đi thông qua giải thuật dijkstra. Nhưng vẫn còn nhiều điểm cần khắc phục nhất là về mặt thời gian chạy sẽ chậm nếu tập dữ liệu thực nghiệm có kích thước lớn.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

## Kết luận

### Những kết quả đạt được

* Ứng dụng được trí tuệ nhân tạo bằng thuật toán dijkstra vào đề tài
* Có thể tìm đường đi ngắn nhất trong nhiều trường hợp
* Hình ảnh Bản đồ có thể mã hóa thành ma trận để có thể áp dụng thuật toán

### Hạn chế

* Kết quả cho ra cứng nhắc, vì một số kết quả vẫn có thể chấp nhận được trong tình huống thực tế
* Thuật toán chạy chậm nếu bản đồ phức tạp hơn, nhiều trở ngai hơn
* Chỉ nhận diện được bản đồ 2D để áp dụng thuật toán

## Hướng phát triển

### Hướng khắc phục các hạn chế

* Đưa ra nhiều kết quả tìm được cho người dùng lựa chọn phương pháp phù hợp
* Cải tiến thuật toán bằng thuật toán A\* hoặc sử dụng thiết bị phần cứng mạnh hơn để chạy chương trình nhanh hơn
* Sử dụng thư viện hỗ trợ để nhận diện được bản đồ phức tạp hơn.

### Hướng mở rộng đề tài

* AI Có thể định vị được vị trí ở thế giới thực với bản đồ được mở rộng
* AI có thể nhận diện tốt đối với các hình ảnh nhiều chiều.

# DANH MỤC TÀI LIỆU THAM KHẢO

1. Võ Xuân Thể (2021), *Bài giảng học phần Trí tuệ nhân tạo*, TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT TP.HCM (lưu hành nội bộ).
2. <https://labs.flinters.vn/algorithm/algorithm-cac-thuat-toan-tim-kiem-trong-ai/>
3. <https://ichi.pro/vi/giai-quyet-me-cung-bang-python-272284064565567>
4. <https://en.wikipedia.org/wiki/Color_difference#Euclidean>

# CÁC PHỤ LỤC

(nếu có)

# ĐÁNH GIÁ KẾT QUẢ HỌC TẬP HỌC PHẦN

ĐỒ ÁN HP THAY CHO BÀI THI KẾT THÚC HỌC PHẦN

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Hình thức KT** | **Nội dung** | **Chuẩn đầu ra đánh giá** | **Trình độ năng lực** | **Phương pháp đánh giá** | **Công cụ đánh giá** | **Tỉ lệ (%)** |
| **BÀI TẬP LẬP TRÌNH** | | | | | |  |
| BL#1 | Bài tập lập trình theo từng Buổi thực hành = tính bình quân  Kiểm tra viết | G1.1 | 3 | Bài tập  Kiểm tra | Bài tập  Kiểm tra | 50 |
| G2.1 | 3 |
| G3.1  G3.2 | 4  3 |
| G4.1 | 5 |
| **ĐỒ ÁN HỌC PHẦN (Project)** | | | | | |  |
| ĐA#1 | Phát triển một hệ thống AI đơn giản:  + Đề tài cá nhân  + Xây dựng và thực nghiệm giải thuật AI theo bài toán cá nhân  => báo cáo vào ngày thi. | G1.1 | 3 | Báo cáo, Demo và thuyết trình | Rubric | 50 |
| G2.1 | 3 |
| G3.1 | 4 |
| G3.2  G4.1 | 3  5 |
| G2.1 | 4 |
| G3.1 | 4 |
| G3.2  G4.1 | 3  5 |