**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 03 (Sáng thứ 3)**

Tên đề tài: **PHÁT TRIỂN TRÒ CHƠI CHROME DINOSAUR DỰA VÀO GIẢI THUẬT LEO NÚI**

Họ tên sinh viên: **Hoàng Hữu Đức**

**Mã số sinh viên: 19110349 [15]**

**Lớp: 191102A**

**Ngày nộp: 19/06/2021**

**Ký tên:**

**TP.HCM, ngày 10 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 03 (Sáng thứ 3)**

Tên đề tài: **PHÁT TRIỂN TRÒ CHƠI CHROME DINOSAUR DỰA VÀO**

**GIẢI THUẬT LEO NÚI**

Họ tên sinh viên: **Hoàng Hữu Đức**

**Mã số sinh viên: 19110349 [15]**

**Lớp: 191102A**

**Tên sản phẩm đề tài: N3S3.15.HoangHuuDuc.19110349.DoaAnHP.AI.rar**

**Công cụ sử dụng (phiên bản): Visual Studio 2019**

**Ngôn ngữ lập trình sử dụng: Python 3.7.8**

**Nhận xét của giảng viên:**

**………………………………………………………..**

**………………………………………………………..**

**Điểm đánh giá:………….(…………………)**

**Ngày……./……../2021**

**Giảng viên Ký tên**

**TP.HCM, ngày 10 tháng 06 năm 2021**

# LỜI CẢM ƠN

Qua đề tài này thì em xin chân thành cảm ơn thầy Võ Xuân Thể với sự tận tâm, tâm huyết của thầy đã truyền đạt cho em, cũng như lớp em những kiến thức quý báu. Cảm ơn thầy đã giúp đỡ và hướng dẫn em trong suốt học kỳ cũng như giúp em đã làm đề tài này. Nhờ đó, em đã hoàn thành đề tài môn Trí tuệ nhân tạo một cách tốt nhất.

Đề tài này được em làm trong suốt quá trình học kỳ này. Trong quá trình làm đề tài thì sản phẩm khó tránh khỏi những sai sót nhỏ. Em mong nhận được sự góp ý từ thầy và các bạn giúp em phát triển đề tài này tốt hơn. Một lần nữa em xin chân thành cảm ơn thầy Võ Xuân Thể đã giúp đỡ em và lớp em trong quá trình học tập môn Trí tuệ nhân tạo ở học kỳ này.

Em xin chân thành cảm ơn thầy!!!!

# DANH MỤC CÁC TỪ VIẾT TẮT

|  |  |
| --- | --- |
|  | |
| CSDL hoặc DB | Cơ sở dữ liệu: DataBase |
|  | |
| AI | Trí tuệ (Trí thông minh) nhân tạo: Artificial Intelligence |
| ES | Hệ thống chuyên gia: Expert Systems |
| ML | Máy học = Học máy: Machine Learning |
| NLP | Xử lý ngôn ngữ tự nhiện: 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

[LỜI CẢM ƠN 3](#_Toc74290050)

[DANH MỤC CÁC TỪ VIẾT TẮT 4](#_Toc74290051)

[DANH MỤC CÁC THUẬT NGỮ 5](#_Toc74290052)

[MỤC LỤC 6](#_Toc74290053)

[PHIẾU GIAO NHIỆM VỤ ĐỀ TÀI ĐỒ ÁN HỌC PHẦN 9](#_Toc74290054)

[Chương 1: GIỚI THIỆU VỀ ĐỀ TÀI 10](#_Toc74290055)

[**1.1.** Tổng quan về đề tài 10](#_Toc74290056)

[**1.2.** Nội dung chuyên môn chính của đề tài 10](#_Toc74290057)

[**1.3.** Ngôn ngữ lập trình và công cụ sử dụng 10](#_Toc74290058)

[**1.4.** Sản phẩm của đề tài 11](#_Toc74290059)

[**1.5.** Bố cục của báo cáo 11](#_Toc74290060)

[Chương 2: CƠ SỞ LÝ THUYẾT VỀ TRÍ TUỆ NHÂN TẠO 12](#_Toc74290061)

[**2.1.** Tổng quan về Trí tuệ nhân tạo (AI) 12](#_Toc74290062)

[2.1.1. Các khái niệm 12](#_Toc74290063)

[2.1.2. Vai trò AI 13](#_Toc74290064)

[2.1.3. Nền tảng kỹ thuật của AI 14](#_Toc74290065)

[2.1.4. Các lĩnh vực nghiên cứu và ứng dụng cơ bản của AI 14](#_Toc74290066)

[**2.2.** Giới thiệu về các bài toán (vấn đề) AI 14](#_Toc74290067)

[2.2.1. Một số khái niệm 14](#_Toc74290068)

[2.2.2. Tìm kiếm lời giải (searching) cho AI Problem 16](#_Toc74290069)

[**2.3.** Biểu diễn một số bài toán AI 17](#_Toc74290070)

[2.3.1. Các bài toán Trò chơi 17](#_Toc74290071)

[2.3.2. Cây không gian trạng thái của Bài toán AI 17](#_Toc74290072)

[**2.4.** Các phương pháp biểu diễn tri thức 17](#_Toc74290073)

[2.4.1. Phân loại tri thức theo phương pháp biểu diễn 17](#_Toc74290074)

[2.4.2. Phân loại tri thức theo nền tảng hình thành Cơ sở tri thức 18](#_Toc74290075)

[2.4.3. Logic mệnh đề và Logic vị từ 19](#_Toc74290076)

[2.4.4. Biểu diễn tri thức bằng Frame (Cấu trúc = "Khung") 22](#_Toc74290077)

[2.4.5. Suy diễn tri thức bằng luật dẫn xuất 22](#_Toc74290078)

[2.4.6. Biểu diễn suy luận bằng đồ thị AND/OR 22](#_Toc74290079)

[2.4.7. Các biểu diễn tương đương trong Logic mệnh đề và Logic vị từ 23](#_Toc74290080)

[2.4.8. Xử lý ngôn ngữ tự nhiên (Natural Language Processing = NLP): Voice Assistant 23](#_Toc74290081)

[**2.5.** Các phương pháp tìm kiếm lời giải của bài toán AI 23](#_Toc74290082)

[2.5.1. Giới thiệu 23](#_Toc74290083)

[2.5.2. Các phương pháp tìm kiếm mù (Blind Search) = tìm kiếm không có thêm thông tin (uninformed search) 23](#_Toc74290084)

[2.5.3. Các phương pháp tìm kiếm theo kinh nghiệm (Heuristic Search) = tìm kiếm với thông tin bổ sung (Informed Search) = Cải tiến BFS 27](#_Toc74290085)

[Chương 3: GIỚI THIỆU VỀ MÔ HÌNH BÀI TOÁN AI: PHÁT TRIỂN TRÒ CHƠI CHROME DINO GIẢI THUẬT LEO NÚI 28](#_Toc74290086)

[3.1. Mô tả bài toán AI 28](#_Toc74290087)

[3.2. Các yếu tố xác định bài toán AI 28](#_Toc74290088)

[Chương 4: BIỂU DIỄN BÀI TOÁN AI VÀ CƠ SỞ TRI THỨC: PHÁT TRIỂN TRÒ CHƠI CHROME DINO GIẢI THUẬT LEO NÚI 29](#_Toc74290089)

[4.1. Giới thiệu 29](#_Toc74290090)

[4.2. Biểu diễn bài toán AI 29](#_Toc74290091)

[4.3. Xây dựng cơ sở tri thức (KBS) cho bài toán AI 29](#_Toc74290092)

[4.4. Lập trình cơ sở tri thức (KBS) cho bài toán AI 29](#_Toc74290093)

[Chương 5: PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI CHO BÀI TOÁN AI: PHÁT TRIỂN TRÒ CHƠI CHROME DINO GIẢI THUẬT LEO NÚI 30](#_Toc74290094)

[5.1. Giới thiệu 30](#_Toc74290095)

[5.2. Mô tả sơ bộ về mô hình cây không gian trạng thái của bài toán AI 30](#_Toc74290096)

[5.3. Cài đặt (lập trình) phương pháp tìm kiếm lới giải cho bài toán AI: Phát triển trò chơi chrome dino bằng giải thuật leo núi. 30](#_Toc74290097)

[Chương 6: GIỚI THIỆU SẢN PHẨM CỦA ĐỀ TÀI AI: PHÁT TRIỂN TRÒ CHƠI CHROME DINOSAUR BẰNG GIẢI THUẬT LEO NÚI 36](#_Toc74290098)

[6.1. Giới thiệu sản phẩm bài toán AI: Chrome dinosaur 36](#_Toc74290099)

[6.2. Kết quả thực nghiệm sản phẩm bài toán AI: Phát triển trò chơi chrome dino dựa vào giải thuật trí tuệ nhân tạo (Giải thuật cải tiến) 36](#_Toc74290100)

[6.3. Nhận xét và đánh giá về sản phẩm bài toán AI: 37](#_Toc74290101)

[Chương 7: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI 38](#_Toc74290102)

[7.1. Kết luận 38](#_Toc74290103)

[7.1.1. Những kết quả đạt được 38](#_Toc74290104)

[7.1.2. Hạn chế 38](#_Toc74290105)

[7.2. Hướng phát triển 38](#_Toc74290106)

[7.2.1. Hướng khắc phục các hạn chế 38](#_Toc74290107)

[7.2.2. Hướng mở rộng của đề tài 38](#_Toc74290108)

[DANH MỤC TÀI LIỆU THAM KHẢO 39](#_Toc74290109)

**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 – N3S3** | |
| Tên đề tài: **Phát triển trò chơi chrome dinosaur dựa vào giải thuật leo núi.** | |
| Sinh viên thực hiện: **Hoàng Hữu Đức, 19110349** | |
| Thời gian thực hiện: **23/02/2021**  đến **10/06/2021** | |
| **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 trò chơi chrome dino dựa vào các giải thuật trí tuệ nhân tạo (AI) (giải thuật cải tiến):  + Thư viện chính được dùng là thư viện pygame, sys, os,math, random  + Giải thuật chính là dùng truy vấn dữ liệu từ 1 dictionary (cơ sở trí thức) trong đó chứa lời giải để thực thi bài toán  + Các chức năng chính của sản phẩm: là AI tự chơi trò chơi khủng long nhảy qua chướng ngại vật. | |
| **GIẢNG VIÊN** | **Ngày 23 tháng 02 năm 2021**  **SV Thực hiện** |

# GIỚI THIỆU VỀ ĐỀ TÀI

## Tổng quan về đề tài

Trò chơi chrome dinosaur thường xuất hiện trên trình duyệt chrome khi chúng ta mở mà không có internet. Chúng ta chơi trò chơi này thường nhấn phím spacebar hoặc cảm ứng để điều khiển con khủng long nhảy qua các chướng ngại vật đến khi kết thúc khi con khủng long va chạm vật thể nào đó. Thì đề tài này sẽ phát triển trò chơi này với giải thuật AI để trò chơi này tự chơi, tự học và tự đưa ra quyết định, cách chơi để con khủng long được đi xa nhất có thể.

## Nội dung chuyên môn chính của đề tài

Trong đề tài này thì em sử dụng các thư viện như:

+ Thư viện PyGame: dùng để viết giao diện game và xử lý các sự kiện xảy ra trong game (Di chuyển, sự va chạm trong các chướng ngại vật,…)

+ Thư viện OS: dùng để tải hình ảnh, các tệp được lưu trữ ở ngoài đưa vào game để chạy (tải hình ảnh khủng long, sương rồng,…)

+ Thư viện random: để tạo các giá trị số ngẫu nhiên.

+ Thư viện math: dùng để viết ra hàm tính toán khoảng cách giữa hai vật trong game.

+ Thư viện sys.

## Ngôn ngữ lập trình và công cụ sử dụng

Trong đề tài, em dùng ngôn ngữ Python vì ngôn ngữ python là ngôn ngữ dễ học, dễ sử dụng, tiện lợi, có nhiều thư viện có sẵn đặc biệt là có nhiều thư viện hỗ trợ trong lập trình trí tuệ nhân tạo. Và ngôn ngữ trong đề tài này là ngôn ngữ Python 3.7.8.

Công cụ lập trình em sử dụng là công cụ visual studio 2019. Vì công cụ này có sẵn ngôn ngữ python. Công cụ này nhìn trực quan và dễ sử dụng.

## Sản phẩm của đề tài

Sản phẩm đề tài của em là một trò chơi Chrome Dinosaur có thể tự chơi, tự vượt qua chướng ngại vật ghi điểm cao và đi xa nhất có thể bằng giải thuật leo núi.

## Bố cục của báo cáo

Chương 1: Giới thiệu về đề tài

Chương 2: Cơ sở lý thuyết về trí tuệ nhân tạo

Chương 3: Giới thiệu về mô hình bài toán ai: phát triển trò chơi Chrome Dino giải thuật leo núi.

Chương 4: Biểu diễn bài toán AI và cơ sở tri thức: Phát triển trò chơi Chrome Dino giải thuật leo núi.

Chương 5: Phương pháp tìm kiếm lời giải cho bài toán AI: Phát triển trò chơi Chrome Dino giải thuật leo núi.

Chương 6: Giới thiệu sản phẩm đề tài AI: Phát triển trò chơi Chrome Dino giải thuật leo núi.

Chương 7: Kết luận và hướng phát triển của đề tài: Phát triển trò chơi Chrome Dino giải thuật leo núi.

# 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 công nghệ thông tin, có khả năng thực hiện các hành vi công việc “khôn” như con người.

+ Có 2 dạng biểu hiện “khôn” của các hệ thống AI: “hành vi” & “nhận thức”/“suy luận”

+ Các đặc trưng chính của hệ thống AI

Không nhất thiết chính xác 100%, tỷ lệ sai tùy thuộc vào hệ thống AI

Không nhất thiết phải giải thích hoặc chứng minh kết quả. Lấy hiệu quả làm mục tiêu của các hệ thống AI.

Có khả năng “học” và “tự học”

“Khôn” theo thời gian

Gồm “chuyên gia chuyên môn” và người lập trình xây dựng phần mềm AI.

Ngôn ngữ và công cụ riêng.

* + - 1. **Phân loại hệ thống Trí tuệ nhân tạo**

+ Dựa vào “nền tảng trí khôn”: heuristic và ES

+ Dựa vào biểu hiện “khôn”: “hành vi” và “nhận thức”/”suy luận”

+ Dựa vào nền tảng kỹ thuật: ML, NLP, ANN, Mờ, …

* + - 1. **Turing Test kiểm tra khả năng khôn của các hệ thống AI**
      2. **Tác tử (Agent: Intelligent Agent) trong hệ thống AI**

Nhân tố bé cấu thành thực hành các chức năng của trí tuệ nhân tạo gọi là tác tử.

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

Tri thức là hệ thống sự hiểu biết rằng lý thuyết hay thực tế về một chủ đề hay lĩnh vực.

Tri thức tự nhiên hoặc tri thức hệ thống (sản phẩm AI là tri thức hệ thống)

Phân biệt: phần mềm hệ thống và phần mềm trí tuệ nhân tạo:

+ Chương trình truyền thống: chạy bằng cấu trúc dữ liệu và thuật toán (thuật giải)

+ Chương trình trí tuệ nhân tạo: chạy bằng cơ sở tri thức (knowledge base) và qui tắc suy diễn (inference engine).

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

Là tập hợp các tri thức liên quan tới một vấn đề nào đó được hệ thống và lưu trữ trên các thiết bị số (máy tính) dưới dạng các luật (rule) hoặc mẫu (pattern).

Các luật nêu trên được hình thành thông qua các kho dữ liệu được tích lũy sau một thời gian lâu dài và khai phá dữ liệu (data mining)

Tập hợp tất cả rule hoặc pattern trong 1 hệ thống của 1 lĩnh vực thì gọi là một cơ sở tri thức (KB)

Chú ý: một số rule được hình thành quá lâu và ít khả năng sao trùng sẽ bị loại ra khỏi hệ thống

Cơ sở trí thức chứa các kiến thức được sử dụng để giải quyết các vấn đề cơ bản trong trí tuệ nhân tạo.

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

Hệ cơ sở trí thức có tối thiểu hai chức năng cơ bản:

+ Khối trí thức: Hình thành và duy trì cơ sở tri thức;

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

Với các hệ thống phức tạp, khối điều kiện cũng có thể là một hệ cơ sở tri thức.

Thực chất là một sản phẩm phần mềm mà trong đó có cung cấp hai nhóm công cụ và phương tiện

+ Các công cụ phương tiện để hình thành duy trì và phát triển các cơ sở trí thức cụ thể là các rule, pattern nhờ vào khai phá dữ liệu.

+ Các công cụ phương tiện hỗ trợ việc khai thác sử dụng cơ sở tri thức vào thực tiễn thông qua các quy luật suy diễn.

Cơ sở tri thức khác với hệ quản trị cơ sở dữ liệu:

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

VD: IPLV, auto LISP, PROLOG

### Vai trò AI

Là nhân tố trung tâm và hâu như không thể thiếu trong các hệ thống hiện nay và tương lai.

AI là yếu tố nền tảng mang tính bắt buộc của các hệ thống trong thời đại CMCN 4.0 (cách mạng công nghiệp lần thứ tư: IR 4.0 = The fourth Industrial Revolution)

### Nền tảng kỹ thuật của AI

Cơ sở tri thức (KB)

Cơ chế suy diễn || suy luận (Inference) và lập luận (Reasoning)

Xử lý ngôn ngữ tự nhiên

Các lý thuyết cơ sở

Lý thuyết biểu diễn tri thức và chuyên gia (logic mệnh đề , …)

Lý thuyết nhận dạng và xử lý tiếng nói (Computer vision)

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

Người máy

Tâm lý xử lý thông minh

Các lĩnh vực nghiên cứu:

Game, tối ưu, dự báo, dự đoán.

## 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 tình huống thực tế mà con người cần giải quyết thuộc một lĩnh vực nào đó, thông thường dựa vào “trí khôn” con người.

VD: Game cơ trưởng, Bài toán tìm đường đi tốt nhất, tư vấn pháp luật, dự đoán hành vi khách hàng,…

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

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 bài cờ (nếu có)

+ Trạng thái bắt đầu: (1 hoặc 1 số)

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

VD:

+ BT cờ tướng => mục tiêu: “chiếu tướng bí”

+ Rubik => mục tiêu: 6 mặt cùng màu

+ Tìm đường đi => mục tiêu: Tìm được 1 (hoặc 1 số) lộ trình đường đi mà người đó 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 1 quân cờ 2 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= quy tắc || quy luật || quy định || ràng buộc,…

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

Mục tiêu của bài toán AI là cơ chế xác định trạng thái mục tiêu nếu 1 trạng thái trong 1 không gian trạng thái thỏa mãn các mục tiêu thì được xem là giải pháp của bài toán.

Mục tiêu của bài toán AI rất đa dạng tùy thuộc vào từng bài toán cụ thể

Các mục tiêu có hai trường hợp: mục tiêu “định tính” và mục tiêu “định lượng”

Mục tiêu “định tính” là mục tiêu được vào tính chất nào đó của trạng thái

Mục tiêu “định lượng” là mục tiêu được xác định bằng các giá trị số cụ thể đo lường được tính toán được.

Các mục tiêu bài toán AI dạng định lượng được xác định bằng hàm mục tiêu

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

+ Hàm mục tiêu có thể xác định dựa vào các tham số (parameters) sau đây:

+ Tham số x là khoảng cách (đơn vị km)

+ Tham số y là thời gian (đo mức độ nhanh chậm của lộ trình đó) (đơn vị: giờ) (thời gian ngắn nhất)

+ Tham số z là chi phí (xác định chi phí cao hay thấp) (mục tiêu chi phí thấp nhất) (đơn vị nghìn đồng)

+ Trên thực tế bài toán trên ngoài tham số nêu trên thì còn nhiều tham số khác trong hàm mục tiêu nhưng tùy vào tính hiệu quả mà có thể bỏ qua không xét.

+ Khi cài đặt hàm mục tiêu này cho bài toán nêu trên thì cần phải xác định phương pháp và cách thức, tính toán giá trị của các tham số. Ví dụ trong bài toán trên các tham số được xác định tại thời điểm vị trí ở A và vì vậy giá trị khoảng cách được tính toán bằng cách tính toán cộng độ dài trong lộ trình dựa vào các tham số có sẵn trên bản đồ, không thể áp dụng phương pháp tính đồng hồ km của phương tiện vì lúc đó đã đến B.

+ Các tham số khác như thời gian y, chi phí z thường được xác định thông qua quá trình tự học của hệ thống phần mềm AI này.

Với mỗi một đối số nêu trên tùy vào từng trường hợp cụ thể người ta có thể xác định mức độ quan trọng khác nhau và mức độ quan trọng này thường được xác định bởi một giá trị số và được gọi là trọng số. Do đó, hàm mục tiêu của bài toán thông thường là dạng tuyến tính, hàm mục tiêu của bài toán nêu trên có thể được biểu diễn như sau:

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

x, y, z đã nêu trên

a1, a2, a3

Ngoài trường hợp dạng hàm mục tiêu tuyến tính, trên thực tế vẫn còn hàm mục tiêu “phi tính” đặc biệt là hàm mục tiêu dạng bậc hai parabol có dạng hệ số a<0. Trong trường hợp này, các vị trí của đỉnh parabol thường là mục tiêu.

Hàm mục tiêu là cơ sở quan trọng để xác định yếu tố thứ 3 của bài toán AI trong 4 yếu tố nêu trên.

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

Tìm kiếm lời giải cho bài toán AI là quá trình khám phá các chuỗi các hành động mà có thể đạt được những mục tiêu của bài toán

Do đó bài toán AI chỉ tìm được lời giải chấp nhận được

Kết quả bài toán AI bé hơn tất cả mục tiêu

Thủ tục tìm kiếm lời giải là thủ tục duyệt các trạng thái dựa vào Agent. Chạy qua các trạng thái trong cây không gian trạng thái của trí tuệ nhân tạo:

+ Vòng lặp chạy các Agent: KGTT, phát hiện trạng thái mục tiêu.

+Cây KGTT của bài toán AI thông thường chưa có sẵn, khi “duyệt” Agent đến đâu thì cây KGTT “mở” nút ra đến đó đến khi gặp “lá” hoặc “bí đường” thì quay lại.(Back tracking).

* Cây KGTT hình thành tùy thuộc phương pháp duyệt của Agent VD: DFS,BFS,…
* Cây KGTT của bài toán AI là vô cùng tận => KQ tìm lời giải của bài toán AI chỉ là các kết quả tìm thấy trong KGTT “đã mở” khó có thể tìm được hết tất cả lời giải.
  + - 1. **Một số dạng bài toán (vấn đề) AI**

Một số dạng bài toán: có 4 dạng bài toán AI quen thuộc

+ Bài toán trạng thái đơn giản

+ Bài toán đa trạng thái:

+ Bài toán ngẫu nhiên

+ Bài toán thăm dò: “đúng – sai”

* + - 1. **Các cấp độ tìm kiếm lời giải cho bài toán (vấn đề) AI**

Các cấp độ tìm kiếm lời giải của bài toán:

Việc tìm kiếm lời giải của một bài toán được chia ra làm 3 cấp độ sau:

* Có tìm được lời giải nào không?
* Đó có phải là một lời giải tốt không (có hàm mục tiêu tốt nhất?)
* Chi phí tìm kiếm với thời gian tìm kiếm và bộ nhớ yêu cầu để tìm một giải pháp đó là bao nhiêu?

## Biểu diễn một số bài toán AI

### Các bài toán Trò chơi

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

* + - 1. **Định nghĩa**

Cây KGTT cuả bài toán AI thường là không có sẵn, khi duyệt agent đến đâu thì cây KGTT mở nút ra đến đó cho đến khi gặp lá hoặc bí đường thì quay lại (back tracking)

🡪Cây KGTT hình thành tùy thuộc vào phương pháp duyệt của agent mà hình thành

🡪Cây KGTT là không gian vô cùng tận, kết quả tìm được là các kết quả tìm thấy trong KGTT đã mở khó có thể tìm được tất cả trạng thái.

* + - 1. **Phương pháp giải quyết bài toán AI**
         1. **Phương pháp gia tăng**
         2. **Phương pháp trạng thái**
      2. **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.

VD: Khẳng định về hiện tượng: mặt trời lặn ở phương tây

Khái niệm về tam giác đều là tam giác có ba góc bằng nhau

* + - 1. **Tri thức mô tả**

Mô tả hiện tượng vận động tự nhiên tri thức mô tả đối tượng hiện tượng hoặc vấn đề trong tự nhiên.

* + - 1. **Tri thức thủ tục**

Là tri thức biểu diễn cách giải quyết một 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.

* + - 1. **Tri thức mêta**

Tri thức tham chiếu đến các tri thức khác (tri thức thứ cấp)

VD: SOP = vị từ cấp hai

VD hệ thống phần mềm hỗ trợ xây dựng phần mềm mới (nên sử dụng EX, Hueristic)

* + - 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 tốt Agent (token) = bao gồm những yếu tố cấu thành giống nhau nhưng giá trị khác nhau (tương tự CTDL).

Trong 5 loại tri thức nêu trên thì tri thức thủ tục là loại phổ biến nhất.

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

* + - 1. **Tri thức hệ Heuristic**

Là tri thức heuristic thông thường là một mẹo(tips), dựa trên kinh nghiệm không gian, dựa vào quy luật xác xuất thống kê liên quan đến quy luật số đông, số lớn, nhằm dẫn dắt tiến trình lập luận vì vậy có nhiều rủi ro đơn giản.

Có tỷ lệ sai do đó phải dùng ràng buộc để các suy luận bài toán AI có ý nghĩa, giá trị.

VD: sử dụng thư viện Python-constraint

Yêu cầu của quy luật phải đáp ứng hai điều cơ bản

+ Đủ lớn

+ Đủ phổ biến/dàn trải

* + - 1. **Tri thức hệ chuyên gia (ES = Expert System)**

Là thuật toán 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.

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

VD: bác sĩ có chuyên môn+kinh nghiệm;

Hầu hết hệ chuyên gia là các hệ dựa luật

Hiện nay một số các hệ chuyên gia thành công trong các lĩnh vực: y học, khai khoáng

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

Giống nhau: đều là cơ sở tri thức

Khác nhau:

Heuristic: số đông, nhìu đối tượng, phổ thông

Expert: kinh nghiệm, số ít, gắn chuyên môn

Ứng dụng: tùy ý, có thể kết hợp cả hai

Chú ý: phân biệt tương đối.

Hệ phần mềm chẩn đoán y khoa: ES + Heuristic

Nhận xét: Phần mềm liên quan chuyên môn (ES)

Liên quan đến dân gian, rút kinh nghiệm: Heuristic.

### Logic mệnh đề và Logic vị từ

* + - 1. **Logic Mệnh đề**

Suy luận logic dựa vào cơ sở tri thức biểu diễn bằng logic mệnh đề và logic vị từ.

Mệnh đề là 1 khẳng định có thể nhận giá trị đúng hoặc sai.

Vị từ là mệnh đề.

Đa số các mệnh đề: đúng/sai tùy thộc vào nhiều yếu tố ngoại cảnh.

(1) Theo thời điểm

(2) Chủ đề liên quan

Chỉ xem xét chân trị của mệnh đề một cách tương đối.

VD: đa số cho rằng đúng; …

***Các ký hiệu***

Các phép hội (và), tuyển (hoặc), phủ định, kéo theo

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

Tất cả các sự kiện trong cơ sở tri thức (CSTT) có thể biểu diễn bởi một số hữu hạn các mệnh đề và các phép kết nối trên.

* + - 1. **Logic vị từ** 
         1. **Vị từ và lượng từ**

Vị từ là cách mở rộng mệnh đề

Ký hiệu: (vị từ||Mệnh đề) 1 <=> (vị từ||mệnh đề) 2

Chú ý: Biểu thức vị từ (có chân trị) = phép nối và lượng từ [mọi, tồn tại, tồn tại duy nhất] & vị từ (hàm) + biến mệnh đề

Các tương đương được sử dụng thường xuyên trong quá trình biến đổi một biểu thức vị từ từ dạng này sang dạng khác.

* + - * 1. **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 cấp 2 (SOP: Second Order Predicate)

-Trong 1 vị từ có vị từ khác thì đó là vị từ cấp 2

VD: Q (P1(x), P2(y))

Chú ý:

-Với 1 tri thức dạng "luật" (rule) có thể biểu diễn toàn bộ dưới dạng FOP (vị từ câp1 1=Lượng từ [mọi, tồn tại, tồn tại duy nhất], Vị từ [MAN], ... , Phép nối[V,^,~])

Hoặc 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.

XOR(x, y) <> XOR(y, x) || OR (x,y) =OR(y,x)

##### Logic tình huống

##### Logic trạng thái (modal)

Liên quan Fuzzy logic (logic mờ)

VD: "anh Bảo ở phòng khách và phòng bếp" = Mờ

Với phát biểu trên:

Anh Bảo ở phòng khách (không liên quan đến bếp)

Anh Bảo ở phòng bếp (không liên quan phòng khách)

Anh Bảo ở giữa 2 phòng = logic trạng thái = "Mờ" nêu trên chấp nhận được = cài đăt đc vào hệ thống AI dưới dạng logic trạng thái

Cho phép biểu diễn các tri thức mờ với nhiều cấp độ tin cậy của mệnh đề hoặc vị từ

=> Suy luận (lập luận) logic cho bài toán AI

VD: 0.8-> p (độ tin cậy 0.8)

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

Liên quan Fuzzy logic (logic "mờ")

Ví dụ: tiếp VD trên 2.3.3.3.3

Anh Bảo ở giữa 2 phòng = 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

ABC sẽ có 30% cơ thể ở giữa cửa nối 2 phòng

=logic xác suất (ướm tương đối -> thực tế không đo lường, mặc dù có PM/SAMRT)

=logic diễn đạt 1 sự kiện nào đó có kèm theo tỉ lệ xác suất xảy ra là bao nhiêu %

VD: ứng dụng trong nghiệp vụ điều tra của cơ quan an ninh

-----------

Logic khả xuất: Là trường hợp cá biệt (đặc biệt) của logic xác suất = thay vì "xác suất" thông thường thì xét "xác suất xảy ra" = bao nhiêu % xảy ra sự kiện đó.

Thường biểu diễn

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

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

VD cụ thể: "Trời nhiều mây thì 80% là chiều nay mưa" = biểu diễn bằng logic vị từ (logic vị từ dạng logic khả xuất)

"Mờ" = không rõ nhiều mây như thế nào

=> Minh hoạ 1 cách giải quyết:"chụp ảnh bầu trời" => phân tích xử lý ảnh

=> Xác định ra tỉ lệ % mây trên bầu trời dựa theo ảnh (biểu diễn dạng 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")

Là phương pháp 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 diễn tri thức dựa trên cơ sở lập trình truyền thống (dùng ctdl định nghĩa Agent =token).

Frame tương tự kiểu dữ liệu đối tượng trong OOP

Frame thường được dùng để biểu diễn những

+ Tri thức chuẩn

+ Hoặc những tri thức dựa trên

### Suy diễn tri thức bằng luật dẫn xuất

#### Khái niệm

Suy diễn = lập luận = suy luận là dựa trên cs tri thức ban đầu (đã có || đã biết)

+ Luật (rule) dẫn xuất (nhiều luật)

==> Xác định (Suy diễn ra||lập luậnđể xác định ra) các sự kiện mới bổ sung vào cơ sở tri thức để cơ sở tri thức ngày càng nhìu hơn.

Sự kiện mới là dạng logic vị từ[chân trị]=tri thức dẫn xuất||"suy luận"||"thứ cấp" nền tảng luật (qui tắc) suy diễn:

+ Suy diễn: tiến=cho tiền đề-> kq có được từ tiền đề đó

+ Suy diễn: lùi/ngược=cho kq -> xđ các tiền đề để có kq đó

VD: xét cơ sở tri thức: pháp luật

Người dân đặt ra yêu cầu một hoạt động nào đó - dựa vào PL --> xác định điều kiện ngược.

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

Là quy tắc làm cơ sở để suy luận tri thức.

Phổ biến là dẫn xuất kéo theo dạng tam đoạn luận.

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

Dùng để suy luận ngược

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

* Nút = sự kiện.
* Cung/cạnh: Suy luận một chiều từ sự kiện này sang sự kiện khác.
* Cung/cạnh từ 1 nút: biểu diễn cho các quan hệ AND/OR 🡪 dẫn xuất sự kiện.
* Gốc: sự kiện mục tiêu.

### 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 lựơc 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**

Chia thành hai nhóm: Uninformed search và Informed search.

* Uninformed search: DFS,BFS,Uninformed search,DWS,Best BFS, …
* Informed search: Greedy search(heuristic search), A\* search,Graph search,G\_bestFS và A\* , …

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

Các phương pháp tìm kiếm mù còn gọi là tìm kiếm mờ (blind search) là tìm kiếm mà không có đủ các thông tin cần thiết:

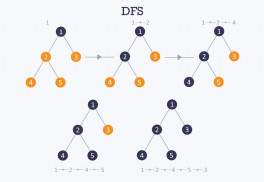
+ Không biết được sẽ qua bao nhiêu bước để có kết quả.

+ Không biết cần “chi phí” bao nhiêu để đi từ trạng thái hiện tại cho tới trạng thái đích

Việc tìm kiếm chủ yếu dựa vào nguyên tắc: Phân biệt một trạng thái đích với một trạng thái không phải là trạng thái đích.

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

Là giải thuật để duyệt cấu trúc cây không gian trạng thái để tìm kiếm lời giải cảu bài toán AI, trong đó nó lần lượt tìm theo chiều sâu của từng nhánh cây trước, cho đến khi không tìm được lời giải trên nhánh đó thì nó quay lui để tìm trên nhánh kế tiếp cho đến khi tìm ra đáp án hoặc duyệt hết toàn bộ không gian cây được xem xét mà vẫn không tìm được 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:**

[1] 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).

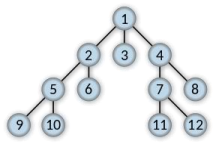
=> Đối với những vấn đề (bài toán) mà có rất nhiều giải pháp (đáp số), phép tìm kiếm sâu có thể nhanh hơn tìm kiếm rộng, bởi vì nó có một cơ hội tốt tìm ra một giải pháp chỉ sau khi khám phá một phần nhỏ của toàn bộ không gian.

=> 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”)

[2] 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 back track 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 không gian trạng thái để tìm kiếm lời giải của bài toán AI, trong đó nó lần lượt tìm theo chiều ngang của từng mức cây trước, cho đến khi không tìm được lời giải trên mức đó thì nó quay lui để tìm trên mức cây kế tiếp, cho đến khi tìm được lời giải hoặc duyệt hết toàn bộ không gian cây được xem xét mà vẫn không tìm được lời giải.



**\* 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 không 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.

. Tìm kiếm theo chiều rộng: trạng thái đích tim 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; không 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]**
      2. **Interative Deepening Search (IDS) = Tìm kiếm sâu dần = cải tiến DFS**
      3. **Depth-Limited Search (DLS) = Tìm kiếm giới hạn độ sâu = cải tiến DFS**

### 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 tối ưu.

Heuristic function là 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.

* + - 1. **Best first: BestBFS (Greedy\_BFS)**
      2. **A\* và SMA\* (A\* với bộ nhớ giới hạn): tìm kiếm trên đồi thị (Graph Search)**
      3. **Hill-Climbing Search (Tìm kiếm tối ưu cục bộ || leo đồi|| lep núi) = HCS**
      4. **Chiến lựơc tìm kiếm đối kháng (Board Game) với Giải thuật MiniMax||MinMax**

# GIỚI THIỆU VỀ MÔ HÌNH BÀI TOÁN AI: PHÁT TRIỂN TRÒ CHƠI CHROME DINO GIẢI THUẬT LEO NÚI

## Mô tả bài toán AI

Bài toán AI chơi game chrome dinosaur là bài toán AI vượt chướng ngại vật, có mục tiêu là vượt qua nhiều chướng ngại vật (Cây xương rồng) đi quãng đường xa nhất có thể. Gồm nhiều trạng thái trong bài toán này và được tập hợp thành cơ sở tri thức. Có sự chuyển đổi trạng thái trong bài toán qua quãng đường trong bài toán.

## Các yếu tố xác định bài toán AI

Bài toán này có nhiều trạng thái khác nhau. (Các trạng thái trong trò chơi là: tốc độ gameplay, chướng ngại vật, khoảng cách giữa các vật thể,…)

Bài toán này có trạng thái bắt đầu là con khủng long di chuyển với số điểm là 0.

Bài toán này có sự chuyển đổi trạng thái khi tốc độ gameplay càng tăng và quãng đường con khủng long di chuyển càng xa (dài).

Mục tiêu của bài toán này là đưa con khủng long di chuyển được xa nhất có thể.

# BIỂU DIỄN BÀI TOÁN AI VÀ CƠ SỞ TRI THỨC: PHÁT TRIỂN TRÒ CHƠI CHROME DINO GIẢI THUẬT LEO NÚI

## Giới thiệu

Bài toán AI này sử dụng cấu trúc dữ liệu dictionary trong ngôn ngữ Python để lúc các thông tin quan trọng trong bài toán này như tốc độ gameplay, khoảng cách các vật thể với nhau (Cụ thể là khoảng cách của con khủng long với cây xương rồng). Trong quá trình trò chơi này chạy thì các thông tin đó được biểu diễn trong trò chơi và được thuật toán lưu lại đưa vào dictionary làm cơ sở tri thức, lấy đó làm nền tảng với các trạng thái khác nhau trong trò chơi thì lấy được dữ liệu tương ứng với trạng thái đó để giải quyết bài toán (là vượt chướng ngại vật đưa con khủng long đi xa nhất có thể).

Phương pháp biểu diễn trên dễ thực hiện, không quá phức tạp, dễ hiểu ý tưởng. Chạy tốt và ổn định, …

## Biểu diễn bài toán AI

Bài toán AI được biểu diễn là dựa vào trạng thái trong chương trình game như tốc độ gameplay và khoảng cách các vật thể thì bài toán AI ghi nhận lại những thông tin đó và tìm kiếm trong cơ sở tri thức để lấy thông tin trả về để quyết định khi nào nhảy để vượt qua chướng ngại vật.

## Xây dựng cơ sở tri thức (KBS) cho bài toán AI

Khi chương trình bắt đầu chạy thì cơ sở tri thức là rỗng. Và cơ sở tri thức của đề tài này được chứa trong tập dữ liệu có cấu trúc là dictionary. Trong quá trình chạy chương trình thì tập dữ liệu này được cập nhật liên tục để có các tri thức phục vụ cho bài toán AI này.

Phân loại theo phương pháp biểu diễn thì cơ sở tri thức của đề tài sử dụng tri thức thủ tục.

Phân loại theo nền tảng hình thành thì cơ sở tri thức của đề tài sử dụng tri thức hệ Heuristic.

## Lập trình cơ sở tri thức (KBS) cho bài toán AI

Cơ sở tri thức trong bài toán AI này sử dụng cấu trúc dữ liệu là dictionary được khởi tạo bằng rỗng và nó được cập nhật thường xuyên trong quá trình học của nó bằng giải thuật leo núi.

# PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI CHO BÀI TOÁN AI: PHÁT TRIỂN TRÒ CHƠI CHROME DINO GIẢI THUẬT LEO NÚI

## Giới thiệu

Bài toán AI này dùng cấu trúc dữ liệu dictionary, kiểu dữ liệu này dễ sử dụng trong việc lưu lại và chỉnh sửa thông tin. Và chúng ta có thể truy vấn dữ liệu trực tiếp từ nó với các thông tin làm từ khóa để lấy dữ liệu. Và cách truy vấn này cũng là phương pháp tìm kiếm lời giải cho bài toán AI này (Giải thuật leo núi).

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

Cây không gian trạng thái của bài toán AI này tùy vào thời điểm mà nó có ít hoặc nhiều trạng thái khác nhau. Cụ thể, khi chương trình vừa bắt đầu chạy thì trong cơ sở tri thức này chỉ có duy nhất một dữ liệu về trạng thái của bài toán sau một thời gian nhất định thì cây không gian trạng thái này ngày càng nhiều tại vì trong quá trình chạy thì dữ liệu được cập nhật liên tục qua các quãng đường. Và cây không gian trạng thái này có thể kéo dài vô tận.

## Cài đặt (lập trình) phương pháp tìm kiếm lới giải cho bài toán AI: Phát triển trò chơi chrome dino bằng giải thuật leo núi.

+ Đoạn code sau là đoạn code khởi tạo các cơ sở tri thức lấy dữ liệu cho phục vụ tìm kiếm lời giải trong bài toán AI này với kb là cơ sở tri thức lưu vận tốc gameplay là từ khóa chứa thông tin về khoảng cách để nhảy, currentdistance (khoảng cách an toàn):

#Co so tri thuc

global kb,safedistance

safedistance=250

kb={20:250}

+ Đoạn code sau lấy dữ liệu từ cơ sở tri thức sau đó suy ra lời giải bài toán AI nếu trạng thái bài toán hiện tại không có trong cơ sở tri thức thì nó sẽ được cập nhật vào cơ sở tri thức nhưng nó chỉ mang ý nghĩa tương đối tại vì lúc này chưa biết lời giải này có giúp được con khủng long vượt qua chướng ngại vật hay không:

#tinh khoang cach tu khung long toi xuong rong

currentdistance=distance((dinosaur.rect.x, dinosaur.rect.y),

obstacle.rect.midtop)

#Quyet dinh nhay dua vao co so tri thuc la kb, co thi lay tu cstt ko la nhay dai

try:

if currentdistance<=kb.get(game\_speed) and dinosaur.rect.y == dinosaur.Y\_POS:

dinosaur.dino\_jump=True

dinosaur.dino\_run=False

except:

if currentdistance <=safedistance and dinosaur.rect.y == dinosaur.Y\_POS:

kb[game\_speed]=safedistance

dinosaur.dino\_jump=True

dinosaur.dino\_run=False

+ Đoạn code sau cập nhật lại dữ liệu vào cơ sở tri thức khi phát hiện con khủng long bị va chạm vào chướng ngại vật sau đó lấy dữ liệu được cập nhật này để phục vụ cho tìm kiếm lời giải cho bài toán AI khi trò chơi được chạy trong lượt mới.

if dinosaur.rect.colliderect(obstacle.rect):#Kiem tra co va cham

global safedistance

record=points

safedistance+=70

kb[game\_speed]=kb[game\_speed]+70;#cap nhat lai

remove(i) **Fulll source code:**

import pygame

import os

import random

import math

import sys

pygame.init()

#Bien toan cuc

#Man hinh

SCREEN\_HEIGHT=600

SCREEN\_WIDTH=1100

SCREEN=pygame.display.set\_mode((SCREEN\_WIDTH,SCREEN\_HEIGHT))

#khung long di chuyen

RUNNING = [pygame.image.load(os.path.join("Assets/Dino", "DinoRun1.png")),

pygame.image.load(os.path.join("Assets/Dino", "DinoRun2.png"))]

#khung long nhay

JUMPING = pygame.image.load(os.path.join("Assets/Dino", "DinoJump.png"))

SMALL\_CACTUS = [pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus1.png")),

pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus2.png")),

pygame.image.load(os.path.join("Assets/Cactus", "SmallCactus3.png"))]

LARGE\_CACTUS = [pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus1.png")),

pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus2.png")),

pygame.image.load(os.path.join("Assets/Cactus", "LargeCactus3.png"))]

BG = pygame.image.load(os.path.join("Assets/Other", "Track.png"))

FONT = pygame.font.Font('freesansbold.ttf', 20)

#Co so tri thuc

global kb,safedistance

safedistance=250

kb={20:250}

#Records

global record

record=0

#class khung long

class Dinosaur:

X\_POS = 80

Y\_POS = 310

JUMP\_VEL = 8.5

def \_\_init\_\_(self, img=RUNNING[0]):

self.image = img

self.dino\_run = True

self.dino\_jump = False

self.jump\_vel = self.JUMP\_VEL

self.rect = pygame.Rect(self.X\_POS, self.Y\_POS, img.get\_width(), img.get\_height())

self.color = (random.randint(0, 255), random.randint(0, 255), random.randint(0, 255))

self.step\_index = 0

def update(self):

if self.dino\_run:

self.run()

if self.dino\_jump:

self.jump()

if self.step\_index >= 10:

self.step\_index = 0

def jump(self):

self.image = JUMPING

if self.dino\_jump:

self.rect.y -= self.jump\_vel \* 4

self.jump\_vel -= 0.8

if self.jump\_vel <= -self.JUMP\_VEL:

self.dino\_jump = False

self.dino\_run = True

self.jump\_vel = self.JUMP\_VEL

def run(self):

self.image = RUNNING[self.step\_index // 5]

self.rect.x = self.X\_POS

self.rect.y = self.Y\_POS

self.step\_index += 1

def draw(self, SCREEN):

SCREEN.blit(self.image, (self.rect.x, self.rect.y))

#pygame.draw.rect(SCREEN, self.color, (self.rect.x, self.rect.y, self.rect.width, self.rect.height), 2)

#for obstacle in obstacles:

#pygame.draw.line(SCREEN, self.color, (self.rect.x + 54, self.rect.y + 12), obstacle.rect.center, 2)

#chuong ngai vat

class Obstacle:

def \_\_init\_\_(self, image, number\_of\_cacti):

self.image = image

self.type = number\_of\_cacti

self.rect = self.image[self.type].get\_rect()

self.rect.x = SCREEN\_WIDTH

def update(self):

self.rect.x -= game\_speed

if self.rect.x < -self.rect.width:

obstacles.pop()

def draw(self, SCREEN):

SCREEN.blit(self.image[self.type], self.rect)

class SmallCactus(Obstacle):

def \_\_init\_\_(self, image, number\_of\_cacti):

super().\_\_init\_\_(image, number\_of\_cacti)

self.rect.y = 325

class LargeCactus(Obstacle):

def \_\_init\_\_(self, image, number\_of\_cacti):

super().\_\_init\_\_(image, number\_of\_cacti)

self.rect.y = 300

#Xoa

def remove(index):

dinosaurs.pop(index)

#tinh khoang cach

def distance(pos\_a, pos\_b):

dx = pos\_a[0]-pos\_b[0]

dy = pos\_a[1]-pos\_b[1]

return math.sqrt(dx\*\*2+dy\*\*2)

def recordDisplay(point):

text = FONT.render(f'Lap: {str(point)}', True, (0, 0, 0))

SCREEN.blit(text, (50, 550))

def main():

global game\_speed, x\_pos\_bg, y\_pos\_bg,obstacles,dinosaurs, points,record

clock=pygame.time.Clock()

points=0

obstacles = []

dinosaurs = [Dinosaur()]

x\_pos\_bg=0

y\_pos\_bg=380

game\_speed=20

def score():

global points, game\_speed

points += 1

if points % 100 == 0:

game\_speed += 1

text = FONT.render(f'Points: {str(points)}', True, (0, 0, 0))

SCREEN.blit(text, (950, 50))

def background():

global x\_pos\_bg, y\_pos\_bg

image\_width = BG.get\_width()

SCREEN.blit(BG, (x\_pos\_bg, y\_pos\_bg))

SCREEN.blit(BG, (image\_width + x\_pos\_bg, y\_pos\_bg))

if x\_pos\_bg <= -image\_width:

x\_pos\_bg = 0

x\_pos\_bg -= game\_speed

run=True

while run:

for event in pygame.event.get():

if event.type==pygame.QUIT:

pygame.quit()

sys.exit()

SCREEN.fill((255,255,255))

for dinosaur in dinosaurs:

dinosaur.update()

dinosaur.draw(SCREEN)

if len(dinosaurs) == 0:

break

if len(obstacles) == 0:

rand\_int = random.randint(0, 1)

if rand\_int == 0:

obstacles.append(SmallCactus(SMALL\_CACTUS, random.randint(0, 2)))

elif rand\_int == 1:

obstacles.append(LargeCactus(LARGE\_CACTUS, random.randint(0, 2)))

recordDisplay(record)

for obstacle in obstacles:

obstacle.draw(SCREEN)

obstacle.update()

for i, dinosaur in enumerate(dinosaurs):

if dinosaur.rect.colliderect(obstacle.rect):#Kiem tra co va cham

global safedistance

record=points

safedistance+=70

kb[game\_speed]=kb[game\_speed]+70;#cap nhat lai

remove(i)

for i,dinosaur in enumerate(dinosaurs):

#tinh khoang cach tu khung long toi xuong rong

currentdistance=distance((dinosaur.rect.x, dinosaur.rect.y),

obstacle.rect.midtop)

#Quyet dinh nhay dua vao co so tri thuc la kb, co thi lay tu cstt ko la nhay dai

try:

if currentdistance<=kb.get(game\_speed) and dinosaur.rect.y == dinosaur.Y\_POS:

dinosaur.dino\_jump=True

dinosaur.dino\_run=False

except:

if currentdistance <=safedistance and dinosaur.rect.y == dinosaur.Y\_POS:

kb[game\_speed]=safedistance

dinosaur.dino\_jump=True

dinosaur.dino\_run=False

score()

background()

clock.tick(30)

pygame.display.update()

gameRun=True;

while gameRun:

main()

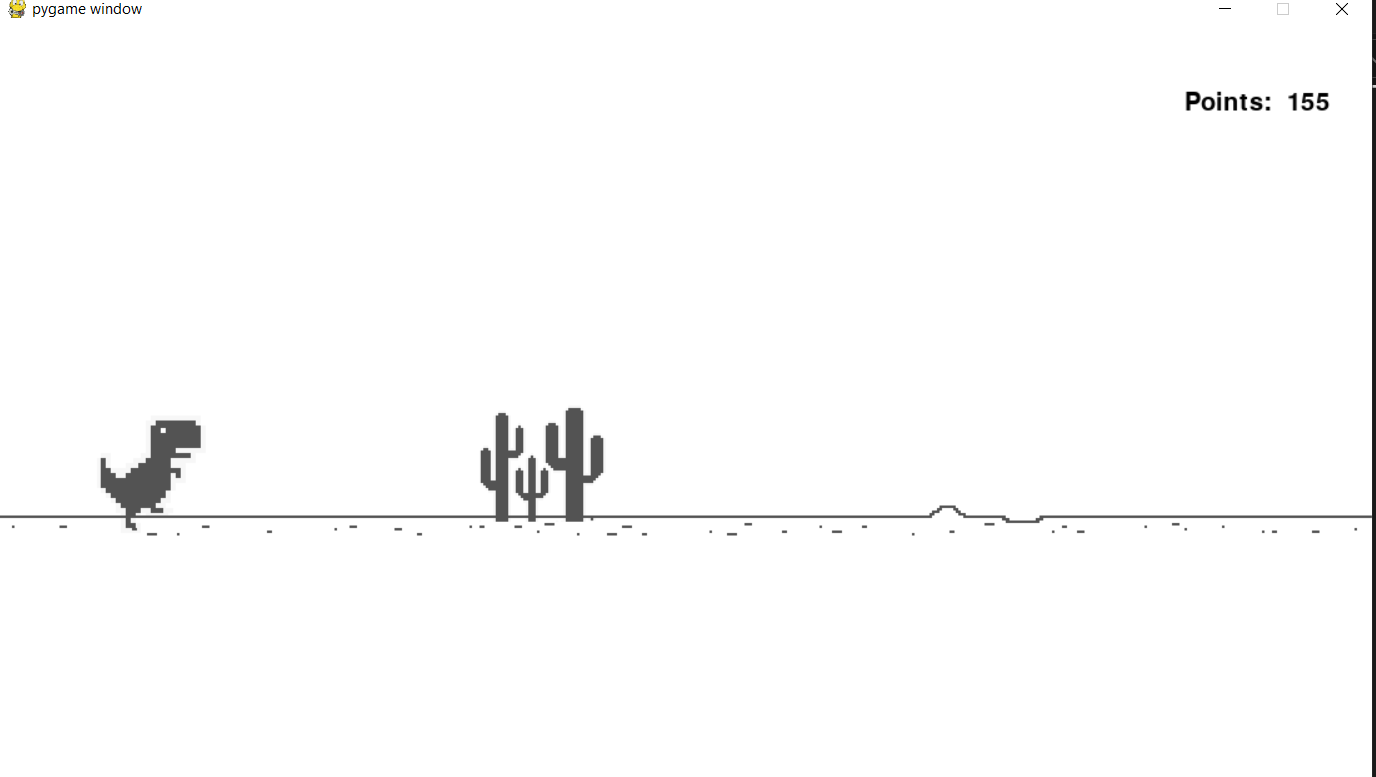
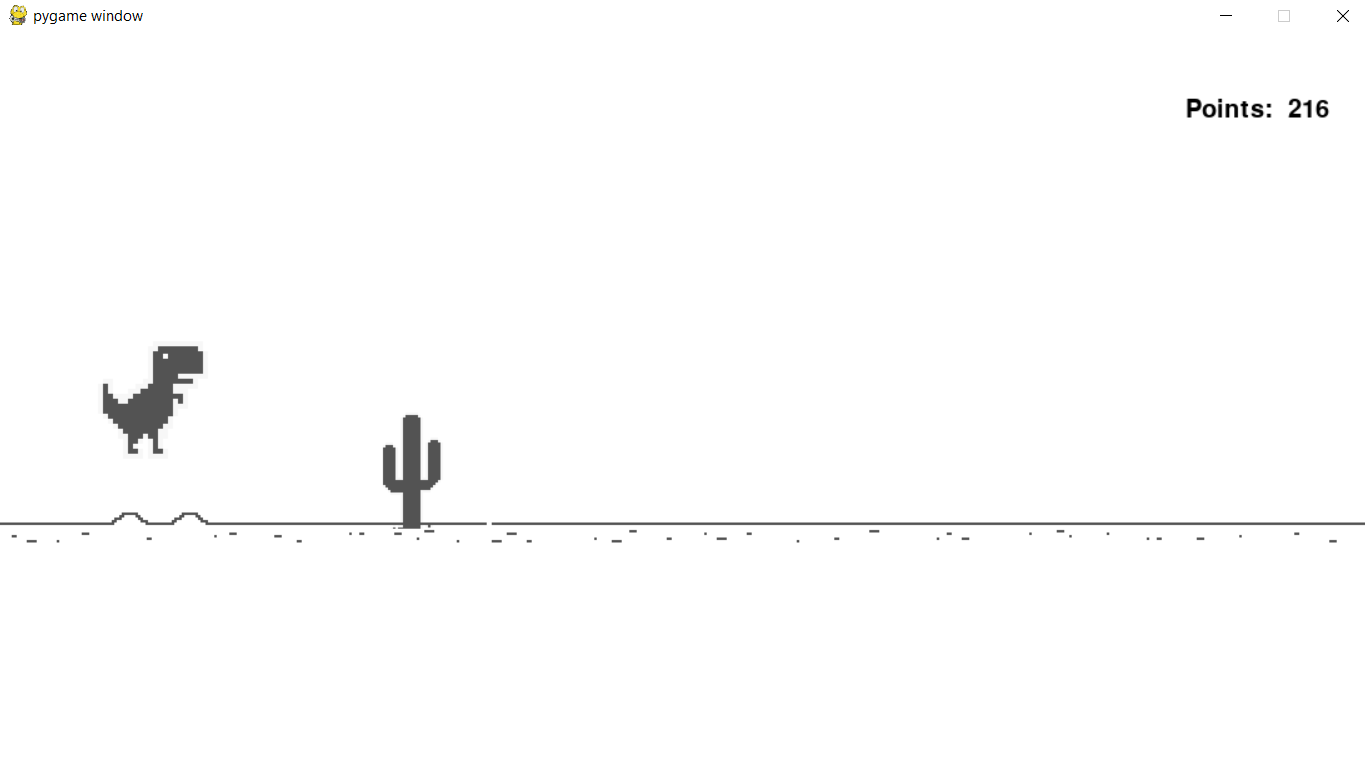
# GIỚI THIỆU SẢN PHẨM CỦA ĐỀ TÀI AI: PHÁT TRIỂN TRÒ CHƠI CHROME DINOSAUR BẰNG GIẢI THUẬT LEO NÚI

## Giới thiệu sản phẩm bài toán AI: Chrome dinosaur

Trò chơi chrome dinosaur thường xuất hiện trên trình duyệt chrome khi chúng ta mở mà không có internet. Chúng ta chơi trò chơi này thường nhấn phím spacebar hoặc cảm ứng để điều khiển con khủng long nhảy qua các chướng ngại vật đến khi kết thúc khi con khủng long va chạm vật thể nào đó. Thì đề tài này sẽ phát triển trò chơi này với giải thuật AI để trò chơi này tự chơi, tự học và tự đưa ra quyết định, cách chơi để con khủng long được đi xa nhất có thể.

Trò chơi này sử dụng ngôn ngữ Python viết bằng công cụ visual studio 2019.

## Kết quả thực nghiệm sản phẩm bài toán AI: Phát triển trò chơi chrome dino dựa vào giải thuật trí tuệ nhân tạo (Giải thuật cải tiến)



## Nhận xét và đánh giá về sản phẩm bài toán AI:

Sản phẩm đề tài này là sản phẩm dễ code, dễ làm, code ngắn gọn, có nhiều nguồn tham khảo, chạy ổn định và hiệu quả.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN CỦA ĐỀ TÀI

## Kết luận

### Những kết quả đạt được

Đề tài này đã làm ra được sản phẩm chrome dinosaur với giải thuật trí tuệ nhân tạo và nó có thể tự chơi, tự học. Với kết quả thu lại tốt, chạy ổn định và hiệu quả.

### Hạn chế

Đề tài này tuy chạy hiệu quả, ổn định nhưng khi cấp độ khó càng tăng cao thì nó học ngày càng khó qua nhiều lượt chơi thì đến cấp độ khó ấy thì đề tài này của em vẫn khó học để nó dễ dàng đạt thêm những kết quả tốt hơn.

## Hướng phát triển

### Hướng khắc phục các hạn chế

Thay đổi thuật toán cho đề tài này hoạt động hiệu quả hơn đặc biệt là khi lên cấp độ khó qua nhiều lượt chơi thì nó có khả năng tiến xa và đạt kết quả tốt hơn.(Backtracking)

### Hướng mở rộng của đề tài

Phát triển thêm thuật toán để chạy tối ưu hơn, đạt hiệu quả cao hơn.

Thêm chế độ người đấu với máy.

Phát triển thêm trợ lý ảo.

# 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. Đoạn code đề tài này được tham khảo từ kênh youtube “Code bucket”

Video này tham khảo cách làm sao để cài đặt hình ảnh khủng long và tạo hiệu ứng di chuyển và nhảy cho nó:

<https://www.youtube.com/watch?v=lcC-jiCuDnQ>

Video này tham khảo cách làm sao để tạo ra các chướng ngại vật trong game, khởi tạo quang cảnh trong game, cách tính điểm và xử lý va chạm của các vật thể trong game (Sự va chạm của khủng long và cây xương rồng):

<https://www.youtube.com/watch?v=NI2R1f1uTDk>

# 