# **TỔNG QUAN VỀ TRÍ TUỆ NHÂN TẠO AI**

1. **Khái niệm về AI:**

**-** Hiện nay, trên thế giới có nhiều định nghĩa khác nhau về trí tuệ nhân tạo. Tuy nhiên, vẫn chưa thống nhất một dạng định nghĩa.

**-** Mặc dù vậy, có 2 trường phái khác về TTNT:

* Strong AI: Có thể tạo ra thiết bị thông minh và các chuong trình máy tính thông minh hơn người.
* Weak AI: Chương trình máy tính có thể mô phỏng các hành vi thông tin của con người.

- Có 4 quan điểm về AI:

|  |  |
| --- | --- |
| Suy nghĩ như người | Suy nghĩ có lý trí |
| Hành động như người | Hành động có lý trí |

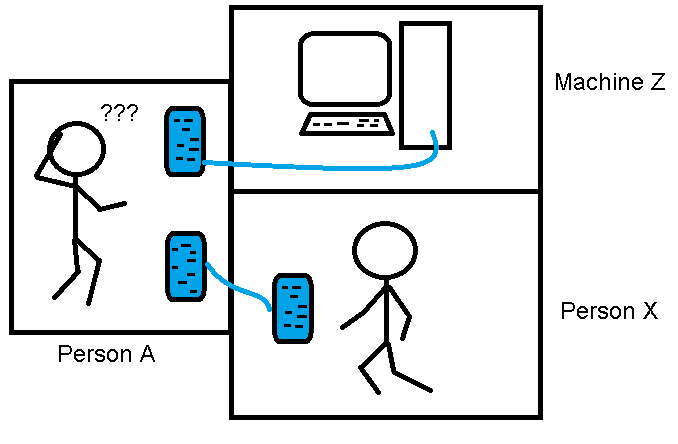
- Trong khuôn khổ chương trình, chúng ta cần tập trung vào quan điểm là *“Hành động có lý trí”.*

* 1. **Hành động như người: Turing Test**

- Turing (1950) “Computing machinery and Intelligence”.

- “Máy có thể suy nghĩ ?” → “Máy tính có thể hành động thông minh ?”

- Turing test: Trò chơi bắt chước người.



- **Ưu điểm:**

* Khái niệm khách quan về trí tuệ.
* Tránh đi những thảo luận về quá trình bên trong và ý thức.
* Loại trừ định kiến thiên vị của người thẩm vấn.

- Các ý kiến phản đối Turing Test:

* Thiên vị các nhiệm vụ giải quyết vấn đề bằng kí hiệu.
* Trói buộc sự thông minh máy tính theo cách hiểu con người, trong khi đó con người có: Bộ nhớ ngắn hạn, có khuynh hướng nhầm lẫn.

- Tuy nhiên, trắc nghiệm Turing đã cung cấp một cơ sở cho nhiều sơ đồ đánh giá dùng thực sự cho các chương trình AI hiện đại.

* 1. **Suy nghĩ như người:**

**-** Chủ yếu quan tâm đến việc nghiên cứu xem trí tuệ của con người là gì? Các chức năng thể hiện trí tuệ như: *Xử lý ngôn ngữ, nghĩ, học, lập luận* được thực hiện như thế nào ?

**-** Hai cách tiếp cận:

* Tiếp cận của khoa học nhận thức → Tìm hiểu *cách suy nghĩ* của con người.
* Tiếp cận theo *diễn biến bên trong não người* → Mô hình mạng nerual và mô phỏng hoạt động của bộ não của con người.
  1. **Suy nghĩ có lý trí:**

**-** Sử dụng các *ký hiệu và luật suy luận* để quản lý các thao tác suy nghĩ.

**-** Sử dụng các *chuẩn hóa (quy tắc hóa)* hơn là các mô tả.

Tiền đề → Luật suy luận → Kết luận.

- Các vấn đề:

* Không phải tất cả các hành động thông minh đều xuất phát từ các suy nghĩ logic.
* Mục đích của sự suy nghĩ là gì? Những suy nghĩ nào mà tôi nên thực hiện trong số các suy nghĩ mà tôi có thể có?
  1. **Hành động có lý trí:**

- Thực hiện đúng việc cần làm (\*).

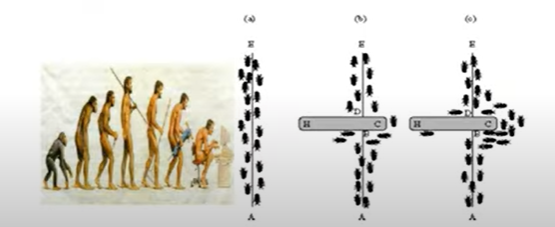
- (\*) nghĩa là việc (hành động) giúp *cực đại hóa* việc đạt được các mục tiêu với điều kiện thông tin phương tiện cho phép môi trường mà nó đang tồn tại.

- Như vây, lợi ích của định nghĩa này là:

* Thông minh, không nhất thiết phải là con người hay giống người.
* Hành vi thông minh không nhất thiết phải thực hiện thông qua suy nghĩ, lí luận.

- Tuy nhiên, cần quan tâm đến tài nguyên tính toán và thời gian thực hiện.

- Ví dụ về TTNT: Sự tiến hóa (Evolutionary Intelligence), Tính bầy đàn (Swarm Intelligence).



- AI là khoa học nghiên cứu các hành vi thông minh nhằm giải quyết các vấn đề được đặt ra đối với các chương trình máy tính.

1. **Các lĩnh vực nền tảng của AI:**

**-** Triết học:

* Phương pháp suy diễn.
* Sự hợp lý.

- Toán học:

* Logic.
* Tính toán.
* Xác suất.

- Kinh tế học:

* Lý thuyết ra quyết định.

- Khoa học thần kinh:

* Nền tảng hoạt động tự nhiên của hoạt động trí óc.

- Tâm lý học:

* Làm sao để con người và động vật có thể suy nghĩ và hành động?

- Công nghệ máy tính:

* Xây dựng máy tính có tốc độ tính toán nhanh.

- Lý thuyết điều khiển:

* Thiết kế hệ thống nhằm cực đại hóa hàm mục tiêu nào đó.

1. **Lịch sử hình thành khoa học về AI:**

- Giai đoạn cổ điển (1950 – 1965):

* Game playing (trò chơi):
* Dựa trên kỹ thuật **State Space Search** (SSS) với trạng thái (State) là các tình huống của trò chơi. Đáp án cần tìm là các trạng thái thắng hay con đường dẫn tới trạng thái thắng, áp dụng với các trò chơi loại đối kháng.
* Ví dụ: Trò chơi đánh cờ vua.
* Có 2 kỹ tuật tìm kiếm cơ bản:
* Kỹ thuật **generate and test:** chỉ tìm được 1 đáp án/ chưa chắc tối ưu.
* Kỹ thuật **Exhaustive search** (vét cạn): Tìm tất cả các nghiệm, chọn lựa phương pháp tốt nhất.
* Theorem Proving (Chứng minh định lý):
* Dựa trên tập tiên đề cho trước, chuong trình sẽ thực hiện chuỗi các suy diễn để đạt được biểu thức cần chứng minh.
* Ví dụ: Chứng minh các định lý tự động, giải toán,…
* Vẫn đưa ra các kỹ thuật SSS nhưng khó khăn hơn do mức độ và quan hệ của các phép suy luận: song song, đồng thời, bắc cầu,..

- Giai đoạn viễn vông (1965 – 1975):

* Tham vọng làm cho máy hiểu được con người thông qua các ngôn ngữ tự nhiên.
* Biểu diễn tri thức giữa người & máy bằng ngôn ngữ tự nhiên.
* Semantic Networks (Mạng ngữ nghĩa).
* Conceptial Graph (Đồ thị khái niệm).
* Frame (Khung).
* Script (Kịch bản).
* Kết quả không mấy khả quan.

- Giai đoạn hiện đại (Từ 1975):

* Xác định lại mục tiêu mang tính thực tiễn.
* Tìm ra **lời giải tốt nhất** trong thòi gian có thể chấp nhận.
* Không nhất thiết phải tìm ra **lời giải tối ưu**.
* Ví dụ:
* Nerual Networks.
* Hiden Markov Models.
* Machine Learning.

1. **Các thành tựu quan trọng ủa AI:**

**-** Tìm kiếm thông tin (Informtion retrieval):

* Trợ lý ảo: Siri, Google now, Cortana, Bixby

- Giao tiếp người máy:

* Giọng nói, cử chỉ, xử lý ngôn ngữ tự nhiên.

- Giải trí (Entertainment):

* Âm nhạc, phim ảnh, trò chơi, tin tức, mạng xã hội.

- Giao thông (Transportation):

* Xe tự lái, giám sát tuân thủ giao thông, dự đoán nhu cầu đi xe.

- Giáo dục và học tập (Education and Learning):

* Tài liệu học tập, quá trình học tập, phổ biến kiến thức.

- Thương mại điện tử:

* Gợi ý sản phẩm / dịch vụ.
* Dự đoán nhu cầu.
* Kế hoạch khuyến mãi.

- An toàn hệ thống (System security):

* Phát hiện virus, phát hiện tấn công mạng, lọc thư rác.

- Tiếp thị và quảng cáo (Marketing and advertisement):

* Giọng nói, cử chỉ, xử lý ngôn ngữ tự nhiên.

1. **So sánh giũa lập trình hệ thống và lập trình AI:**

|  |  |
| --- | --- |
| **Lập trình hệ thống** | **Lập trình AI** |
| Dữ liệu + Thuật toán = Chương trình | Tri thức + Điều khiển = Chương trình |
| Xủ lí dữ liệu | Xử lý dữ liệu định tính (các ký hiệu đặc trưng) |
| Dữ liệu trong bộ nhớ được đánh địa chỉ số | Xử lý dụa trên tri thức cho phép dùng các giải thuật Heuristic, các cơ chế suy diễn |
| Xử lý theo các thuật toán | Trí thức được cấu trúc để trong bộ nhớ làm việc theo kí hiệu |
| Định hướng xử lý các đại lượng định lượng só | Định hướng xủ lý các đại lượng định tính (logic), các ký hiệu tượng trưng và danh sách |
| Xử lý tuần tự theo mẻ | Xử lý theo chế độ tương tác (hội thoại ngôn ngữ tự nhiên) |
| Không giải thích trong quá trình thực hiện | Có giải thích các hành vi của hệ thống trong quá trình thực hiện |
| Kết quả chính xác, không được mắc lỗi | Kết quả tốt, cho phép mắc lỗi |

1. **Những vấn đề chưa được giải quyết:**

**-** Chương trình chưa tự sinh ra được Heuristic.

**-** Chưa có khả năng xử lý song song của con người.

**-** Chưa có khả năng diễn giải một vấn đề theo nhiều phương pháp khác nhau như con người.

**-** Chưa có khả năng xử lý thông tin trong môi trường liên tục như con người.

**-** Chua có khả năng học như con người.

**-** Chưa có khả năng tự thích nghi với môi trường.

1. **Vấn đề cốt lõi của AI:**

**-** Biểu diễn (Representation).

- Lập luận (Reasoning).

- Học (Learning).

- Tương tác (Interaction).

A picture containing diagram, circle, drawing, sketch

Description automatically generated

1. **Các công nghệ mở về AI:**

**-** TensorFlow:

* OS: Linux, Mac, Windows, Android.
* Ngôn ngữ khai thác: Python, C++, Java.

**-** Caffe:

* OS: Linux, Mac, Windows.
* Ngôn ngữ khai thác: Python, Matlab.

**-** PyTorch:

* OS: Linux, Mac, Windows, Android, iOS, Raspbian.
* Ngôn ngữ khai thác: Python, C++.

**-** Theano (Deep Learning):

* OS: Linux, Mac, Windows.
* Ngôn ngữ khai thác: Python.

**-** CNTK[[1]](#footnote-1):

* OS: Linux, Windows.
* Ngôn ngữ khai thác: Python, C++, C#.

**-** Deeplearning4j:

* OS: Linux, Mac, Windows, Android.
* Ngôn ngữ khai thác: Java, Scala, Cloujure, Python.

**-** Weka:

* OS: Mọi hệ điều hành có cài máy ảo (JVM[[2]](#footnote-2)).
* Ngôn ngữ khai thác: Java.

# **PYTHON CƠ BẢN**

1. **Khái niệm:**
2. **Môi trường cài đặt**
3. **Khái niệm cơ bản trong Python**

* **Thao tác đơn giản trong Python**

**A screen shot of a computer

Description automatically generated with low confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer code

Description automatically generated with low confidence**

**A screenshot of a computer program

Description automatically generated with low confidence**

* **Thao tác chuỗi trong Python**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with low confidence**

* **Tuples trong Python**

**A screen shot of a computer code

Description automatically generated with low confidence**

**A screen shot of a computer code

Description automatically generated with low confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screen shot of a computer program

Description automatically generated with low confidence**

**A screenshot of a computer program

Description automatically generated with low confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

* **List trong Python**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with low confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

* **Dictionaries trong Python**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A picture containing text, screenshot, font

Description automatically generated**

**A picture containing text, screenshot, font

Description automatically generated**

**A screenshot of a computer program

Description automatically generated with low confidence**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A picture containing text, font, screenshot, design

Description automatically generated**

* **Set trong Python**

**A screenshot of a computer

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer code

Description automatically generated with low confidence**

**A screen shot of a computer program

Description automatically generated with low confidence**

* **Conditions trong Python**

**A picture containing text, screenshot, font, design

Description automatically generated**

* **Loops trong Python**

**A picture containing text, screenshot, font, graphics

Description automatically generated**

1. **Hàm trong Python**

**A picture containing text, screenshot, font, software

Description automatically generated**

1. **Class và Object trong Python**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**A screen shot of a computer program

Description automatically generated with low confidence**

**A picture containing text, font, screenshot

Description automatically generated**

**A screen shot of a computer program

Description automatically generated with medium confidence**

**A screenshot of a computer program

Description automatically generated with medium confidence**

**put(item)**: Phương thức này được sử dụng để thêm một phần tử vào hàng đợi ưu tiên. Nó nhận đối số **item**, là phần tử bạn muốn thêm vào hàng đợi ưu tiên. Trong trường hợp của bạn, **item** là một đối tượng **color**. Khi thêm một phần tử vào hàng đợi ưu tiên, phần tử sẽ được đặt vào vị trí phù hợp dựa trên các yếu tố ưu tiên đã được xác định bởi phương thức **\_\_lt\_\_**.

**get()**: Phương thức này được sử dụng để lấy phần tử ưu tiên đầu tiên từ hàng đợi ưu tiên. Nếu hàng đợi không rỗng, phương thức **get** sẽ trả về phần tử ưu tiên đầu tiên và loại bỏ nó khỏi hàng đợi. Trong trường hợp của bạn, phần tử ưu tiên là đối tượng **color** với trọng số nhỏ nhất.

**\_\_lt\_\_(self, other)**: Đây là một phương thức so sánh nằm trong lớp **color**. Nó được sử dụng để xác định quy tắc so sánh giữa hai đối tượng **color**. Trong trường hợp này, **\_\_lt\_\_** được định nghĩa để so sánh theo trường **weight**. Phương thức trả về **True** nếu trọng số của đối tượng hiện tại nhỏ hơn trọng số của đối tượng khác (**self.weight < other.weight**), và **False** trong trường hợp ngược lại.

# **BÀI TOÁN VÀ PHƯƠNG PHÁP TÌM KIẾM LỜI GIẢI**

1. **Tổng quan về tìm kiếm**

* Tìm kiếm là một thuật toán lấy đầu vào là một bài toán và trả về kết quả là một lời giải cho bài toán đó, thường là sau khi cân nhắc hàng loạt lời giải có thể có.
* Tập hợp tất cả các lời giải có thể có đối với một bài toán gọi là không gian tìm kiếm.
* Phương pháp sơ đẳng nhất của tìm kiếm là thuật toán thử sai (brute-force search) – đây là phương pháp đơn giản và trực quan nhất.

1. **Không gian trạng thái**

* Không gian trạng thái trong tìm kiếm:
* Là tập hợp tất cả các trạng thái có thể có của bài toán.
* Các trạng thái mới được sinh ra sau khi thực hiện các toán tử di chuyển trạng thái.
* Được biểu diễn bởi cây, đồ thị,..
* Các phương pháp tìm kiếm:
* Tìm kiếm không có thông tin (tìm kiếm mù)
* Tìm kiếm có thông tin
* Tìm kiếm đối kháng

1. **Biểu diễn các bài toán trong không gian trạng thái**

* Biểu diễn trong không gian trạng thái – Trò chơi 8 số
* Trạng thái đầu {1, 4, 3, 7, null, 6, 5, 8, 2}
* Trạng thía đích {1, 2, 3, 4, 5, 6, 7, 8, null}
* Toán tử:
* Up
* Down
* Left
* Right
* Note
* Trạng thái sẽ thay đổi phụ thuộc vào toán tử.
* Tùy thuộc vào trạng thái mà sẽ có những toán tử không thực hiện được.

**A diagram of a crossword puzzle

Description automatically generated with low confidence**

* Biểu diễn trong không gian trạng thái – Bài toán người đưa thư

**A diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of a diagram of

Description automatically generated with medium confidence**

* Các bài toán khác:
* Tìm đường đi trong mê cung
* Bài toán tìm kiếm thông tin trên internet
* Bài toán 8 quân hậu

1. **Đồ thị tìm kiếm và cây tìm kiếm**

* Đồ thị tìm kiếm:

**A picture containing diagram, screenshot, design

Description automatically generated**

* Cây tìm kiếm:

**A picture containing circle, diagram, design, art

Description automatically generated**

1. **Các chiến lược tìm kiếm trên không gian trạng thái**

* Tìm kiếm hướng từ dữ liệu (Data-Driven Search)
* Việc tìm kiếm đi từ dữ liệu đến mục tiêu
* Thích hợp khi:
* Tất cả hoặc một phần dữ liệu được cho từ đầu
* Có nhiều mục tiêu nhưng chỉ có một số ít các phép toán có thể áp dụng cho một trạng thái bài toán
* Rất khó đưa ra một mục tiêu hay giả thuyết ngay lúc đầu
* Tìm kiếm hướng mục tiêu (Goal-Driven Search)
* Việc tìm kiếm đi từ mục tiêu trở về dữ liệu
* Thích hợp khi:
* Có thể đưa ra mục tiêu hoặc giả thuyết ngay từ lúc đầu
* Có nhiều phép toán có thể áp dụng trên 1 trạng thái → bùng nổ số lượng trạng thái
* Các dữ liệu của bài toán không được cho trước, nhưng hệ thông phải đạt được trong quá trình tìm kiếm

1. **Tìm kiếm trên không gian trạng thái**

* Tìm kiếm rộng

**A picture containing circle, diagram, sketch, white

Description automatically generated**

* Tìm kiếm rộng là phương pháp tìm kiếm mà ưu tiên những đỉnh gần với đỉnh xuất phát
* Thứ tự tìm kiếm: A → B → C → D → E → F → G → H → I → J → K → L → M → N → O
* **Thuật toán tìm kiếm mở rộng:**

**B1**: Cho đỉnh xuất phát vào Open[[3]](#footnote-3).

**B2**: Nếu Open rỗng thì tìm kiếm thất bại, kết thuc việc tìm kiếm.

**B3**: Lấy đỉnh đầu trong Open ra và gọi đó là θ. ***Cho 𝜽 vào Closed.***

**B4**: Nếu θ là đỉnh đích thì tìm kiếm thành công, kết thúc việc tìm kiếm.

**B5**: Tìm tất cả các đỉnh con của ***θ không thuộc Open và Closed*** cho vào cuối của Open.

**B6**: Trở lại bước 2.

* Quá trình biến đổi của Open:

Open = {**A**}

Open = {**B**, **C**, **D**}

Open = {**C**, **D**, **E**, **F**}

Open = {**D**, **E**, **F**, **G**, **H**}

Open = {**E**, **F**, **G**, **H**, **I**, **J**}

…

* Tổng số đỉnh tối đa cần xét là: 1 + b + b2 + … + bd

Trong đó:

b là số nhánh tối đa của một trạng thái.

d là độ sâu

Độ phức tạp về thời gian và không gian của thuật toán là **O(bd)**

* Code mô phỏng:
* from collections import defaultdict
* data = defaultdict(list)
* data['A'] = ['B', 'C', 'D']
* data['B'] = ['E', 'F']
* data['C'] = ['G', 'H']
* data['D'] = ['I', 'J']
* data['F'] = ['K', 'L', 'M']
* data['H'] = ['N', 'O']
* class Node:
* def \_\_init\_\_(self, name, w = 0, par = None): # w = weight
* self.name = name
* self.par = par
* self.w = w
* def display(self):
* print(self.name, self.par)
* def \_\_lt\_\_(self, other):
* return self.w < other.w
* def equal(O, G):
* return O.name == G.name
* def checkArray(tmp, Open):
* for x in Open:
* if equal(x, tmp):
* return True
* return False
* def path(O):
* print(O.name)
* if O.par != None:
* path(O.par)
* else:
* return
* def BFS(S = Node('A'), G = Node('M')):
* Open = []
* Closed = []
* Open.append(S)
* while True:
* if len(Open) == 0:
* print('Tim kiem that bai')
* return
* O = Open.pop(0)
* Closed.append(O)
* print(O.name)
* if equal(O, G):
* print('Tim thay')
* path(O)
* return
* for x in data[O.name]:
* tmp = Node(x)
* tmp.par = O
* ok1 = checkArray(tmp, Open)
* ok2 = checkArray(tmp, Closed)
* if not ok1 and not ok2:
* Open.append(tmp)
* BFS()

# Các phương pháp tìm kiếm Heuristic

1. **Tìm kiếm Heuristic**- Là tìm kiếm theo kinh nghiệm.

- Ví dụ: Khi không rõ đường đi, thường thì mọi người nhắm về hướng cần tới để đi (Ưu tiên đường đi mà có khoảng cách chim ay nhỏ nhất)

- Heuristic là tìm kiếm sử dụng kiến thức phỏng đoán chi phí từ đỉnh đang xét đến đích.

1. **Hàm đánh giá trong tìm kiếm Heuristic**

**-** Trong tìm kiếm Heuristic, với mỗi trạng thái u, xác định một hàm h(u) – hàm đánh giá, hàm này được sử dụng để đánh giá mức độ “tốt” của trạng thía u so với trạng thái đích.

**-** Các giái đoạn cơ bản của Heuristic:  
+ Tìm biểu diễn thích hợp mô tả các trạng thái và các toán tử.

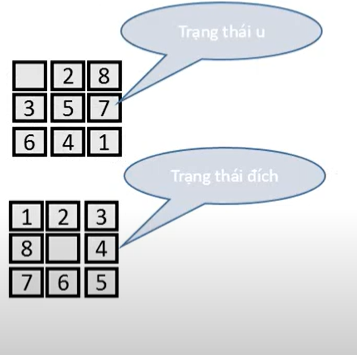
+ Xây dựng hàm đánh giá.  
+ Thiết kế chiến lược chọn trạng thái để phát triển ở mỗi bước.

**- Ví dụ 1:** Trong bài toán tìm kiếm dường đi trên bản đồ, có thể xây dựng hàm đánh giá:

+ Đường chim bay từ thành phố này sang thành phố khác, hoặc

+ Sử dụng khoảng cách thực tiễn trên đường đi giũa các thành phố, hoặc

+ Sử dụng vả khaongr cách và một số trọng số kác ảnh hưởng tới việc tìm kiếm (đóng vai trò làm tăng thời gian di chuyển giữa các thành phố),…

- **Ví dụ 2:** xét bài toán 8 số ta có thể xây dựng hàm đánh giá như sau:

+ Hàm H1, với mỗi trạng thái u thì H1(u) là số quân sai vị trí.

+ Hàm H2, là tổng khoảng cách Mahattan (|x2 – x1|, |y2 – y1|) giữa vị trí quân ở trạng thái u với vị trí trạng thái đích.

+ Lúc này:

\* H1(u) = 7 (chỉ xét từ 1 đến 8)

\* H2(u) = 4 + 0 + 3 + 2 + 2 + 1 + 3 + 3 = 18

1. **Tìm kiếm chi phí cực tiểu (uniform cost search)**

**-** Khi một bài toán có rất nhiều nghiệm thì thường chọn những phương án có “chi phí thấp nhất”  
- Với các thuật toán tìm kiếm mù (DFS và BFS) thì không đảm bảo được việc tối ưu cho bài toán “Chi phí thấp nhất”

**- Ví dụ:** Tìm đường đi từ thành phố A sang thành phố B thỏa mãn nhanh nhất và rẻ nhất.

**- Uniform cost search:** Đảm bảo tìm được “Chi phí thấp nhất”

1. CNTK (Microsoft Cognitiive Toolkit): Là bộ công cụ mã nguồn mở thương mại, có thể được dùng để trainning thuật toán deep learning. [↑](#footnote-ref-1)
2. JVM (Java Virtual Machine): Là một máy ảo Java, hiểu nôm na là trình thông dịch của Java. [↑](#footnote-ref-2)
3. Open hoạt động theo quy tắc FIFO [↑](#footnote-ref-3)