THỰC HÀNH TOÁN RỜI RẠC

TÀI LIỆU PHỤC VỤ SINH VIÊN NGÀNH KHOA HỌC DỮ LIỆU

Nhóm Giảng viên biên soạn: TS. Hoàng Lê Minh – Khưu Minh Cảnh – Phạm Trọng Nghĩa – Nguyễn Công Nhựt – Trần Ngọc Việt - Hoàng Thị Kiều Anh – Lê Ngọc Thành – Đỗ Đình Thủ – Nguyễn Hữu Trí Nhật – Lê Công Hiếu – Nguyễn Thị Thanh Bình – Nguyễn Thái Hải – Huỳnh Thái Học và các Giảng viên khác

TP.HCM – Năm 2020

MỤC LỤC

CHƯƠNG 5: QUAN HỆ TRONG TẬP HỢP

Mục tiêu:

- *Tìm hiểu về các loại quan hệ giữa hai ngôi trên một tập hợp*
- *Biểu diễn quan hệ, suy luận bằng toán học và Python*
- *Các ứng dụng sử dụng quan hệ toán học để xử lý dữ liệu*

Nội dung chính:

1. Dẫn nhập: Bài toán nói dối – nói thật: Knights và Knaves

Bài toán dẫn nhập cho thấy các tập dữ liệu luận lý đôi khi có sự liên hệ (quan hệ với nhau). Mở rộng ra, với các dữ liệu khác, sự liên hệ giữa các tập có thể tồn tại nhiều mối quan hệ. Trong phần dẫn nhập này, chúng ta xét đến quan hệ giữa 2 tập sự kiện có chung đặc điểm liên hệ về thời gian như sau:

Hai ông James và Jonathan đều nói dối vào những ngày nhất định.

James nói dối vào thứ Sáu, thứ Bảy và Chủ Nhật, nhưng nói thật vào tất cả những ngày còn lại.

Jonathan nói dối vào thứ Ba, thứ Tư và thứ Năm, nhưng nói thật vào tất cả những ngày còn lại.

Thế thì vào ngày nào trong tuần cả hai đều nói "**Ngày mai, tôi sẽ nói dối?**"

Các nhận xét:

- Nhận xét 1: Gọi nói **dối** là Fasle, nói **thật** là True. Như vậy ta sẽ lập được 1 hàm trả về trị đúng/sai tương ứng với nói thật/dối (như một hàm Bool với "nói dối" được xem như "mạch đảo", nghĩa là ngược lại).
- Nhận xét 2: Hàm Bool sẽ nhận vào 1 giá trị đầu tiên là Sai (nói dối). Tuy nhiên, giá trị sẽ được tính toán phụ thuộc vào giá trị ngày. Cụ thể: Nếu hôm nay là ngày nói **dối** (False) thì nếu ông này nói dối nữa, nghĩa là giá trị đầu vào là False, nghĩa là ông ta đang nói thật (kết quả sẽ là True). Ngược lại, nếu hôm nay là True thì chính xác ông đang đang nói dối (nghĩa là giá trị False như lời ông nói).
- Nhận xét 3: Nghĩa là hàm Bool sẽ có 2 giai đoạn: bước 1: tìm dự báo ngày mai từ giá trị hôm nay và bước 2 là xác định sự đúng đắn từ dự báo và hiện trạng.
- Nhận xét 4: với 1 tuần có 7 ngày và xoay vòng, nếu chọn ngày đầu tiên của tuần là Thứ 2 thì chúng ta cần xét đến ngày Chủ Nhật. Như vậy, với mỗi người James hoặc Jonathan, chúng ta cần mô tả tình trạng nói dối của họ với 7+1 giá trị (để xoay vòng)
- Nhận xét 5: James và Johnathan là 2 bộ dữ liệu khác nhau.

Sinh viên thực hành tìm ngày cả 2 cùng nói câu: "Ngày mai, tôi sẽ nói dối!"!

Bảng "chân trị" của hàm "nói dối":

Sự thật **Hôm nay** *(1)*	Dự báo **Ngày mai** từ câu nói Hôm nay *(2, được suy từ 1)*	Sự thật **Ngày mai** *(3)*	Kết quả so sánh trùng khớp *(giữa 2 và 3)*
Thật	Dối	Thật	Sai
Thật	***Dối***	***Dối***	Đúng
Dối	***Thật***	***Thật***	Đúng
Dối	Thật	Dối	Sai

Bảng chân trị về câu nói của James và Jonathan:

	Thứ 2	Thứ 3	Thứ 4	Thứ 5	Thứ 6	Thứ 7	Chủ nhật	Thứ 2
James	Thật	Thật	Thật	Thật	Dối	Dối	Dối	Thật
Câu nói của James	**Sai**	**Sai**	**Sai**	**Đúng**	**Sai**	**Sai**	**Đúng**	
Jonathan	Thật	Dối	Dối	Dối	Thật	Thật	Thật	Thật
Câu nói của Jonathan	**Đúng**	**Sai**	**Sai**	**Đúng**	**Sai**	**Sai**	**Sai**	

Diễn giải bảng trên: Với James, thứ 2 và thứ 3 đều nói thật. Do đó, nếu thứ 2 James nói: "Ngày mai nói dối" nghĩa là mệnh đề này sai. Tương tự với Chủ nhật và Thứ 2, James nói dối mà nói "Mai nói dối" nghĩa là mệnh đề đó sẽ Đúng.

Đoạn mã Python như sau:

>>> James = [True, True, True, True, False, False, False, True]

>>> Jonathan =[True, True, True, True, False, False, False, True]

>>> ngay_trong_tuan = ["Thu 2", "Thu 3", "Thu 4", "Thu 5", "Thu 6", "Thu 7", "Chu nhat", "Thu 2"]

- *Code chưa rút gọn:*

```
>>> for i in range(0, 6):
        if ((James[i] == True and James[i+1] == False)
            or ((James[i] == False and James[i+1] == True))):
                if ((Jonathan[i] == True and Jonathan[i+1] == False)
                    or ((Jonathan[i] == False and Jonathan[i+1] == True))):
                        print (ngay_trong_tuan[i])
```

…… ← sinh viên điền kết quả tìm được.

- *Code rút gọn:*

Tìm ra quy luật là hôm nay và ngày mai phải khác nhau.

```
>>> for i in range(0,6):
        if ((James[i] and not James[i+1]) and (Jonathan[i] and not Jonathan[i+1])):
            print (ngay_trong_tuan[i])
```

…… ← sinh viên điền kết quả tìm được.

2. [Phần giảng viên hướng dẫn để SV làm bài tập] Quan hệ trên các tập hợp và biểu diễn quan hệ

Dẫn nhập: Thông thường, với các phần tử (như số liệu) được cho, ta gọi chúng là một tập hợp, chúng ta cần tìm mối quan hệ giữa các phần tử đó. Có nhiều loại quan hệ giữa hai phần tử với nhau, như quan hệ toán học chia hết, đồng dư (cùng số dư khi chia cho 1 số), lớn hơn, nhỏ hơn,…

Ví dụ: *Cho ngẫu nhiên một dãy số S có thứ tự gồm 10 số tự nhiên từ 1 đến 99. Sau đó, ứng với mỗi số, hãy tìm số lượng các số phía sau nó lớn hơn nó.*

Sinh viên thực tập các câu lệnh sau:

```
>>> import random
>>> S = random.sample(range(1, 100), 10)  # lấy mẫu 10 số tự nhiên trong [1, 99]
>>> print (S)
……
>>> tS = []      # dãy tS lưu trữ số lượng số nằm ở phía sau Si mà lớn hơn Si, i=0,n-1
>>> for i in range(len(S)):
        tS.append(0)
        for j in range(i, len(S)):
            if S[i] < S[j]:
```

tS[i] = tS[i] + 1

>>> print (tS)

……

Quan hệ trên tập hợp:

- Định nghĩa: Một quan hệ hai ngôi từ tập A đến tập B là tập con của tích Descarte $R \subseteq A \times B$.
- Kí hiệu: a R b [thay cho $(a, b) \in R$].

Ví dụ:

- Quan hệ giữa 2 tập: A là tập sinh viên; B là tập lớp học. Khi đó
$$R = \{(a, b) | \text{ sinh } viên\ a\ trong\ lớp\ học\ b\}$$
- Quan hệ ước số: Tập $A = \{1,2,3,4,5,6\}, R = \{(a, b) | a\ là\ ước\ số\ của\ b\}$.Khi đó:

$R = \{(1,1), (1,2), (1,3), (1,4), (1,5), (1,6), (2,2), (2,4), (2,6), (3,3), (3,6), (4,4), (5,5), (6,6)\}$

Tính chất:

Cho quan hệ R trên A, quan hệ R được gọi là:

- **Phản xạ**: nếu $\forall a \in A, a\ R\ a$.
  Ví dụ: Quan hệ ước số vì mọi số đều là ước số của nó. Ví dụ: 10 là ước số của 10.
- **Đối xứng**: nếu $\forall a \in A, \forall b \in A: (a\ R\ b) \rightarrow (b\ R\ a)$
  Ví dụ: [Quan hệ đồng dư] $a, b, k \in \mathbb{Z}, a \equiv b\ (mod\ k) \rightarrow b \equiv a\ (mod\ k)$
- **Phản xứng**: nếu $\forall a \in A, \forall b \in A: (a\ R\ b) \wedge (b\ R\ a) \rightarrow (a = b)$
  Ví dụ: Quan hệ $\leq$ trên trường $\mathbb{Z}$ là phản xứng vì với 2 số $a, b \in \mathbb{Z}: (a \leq b) \wedge (b \leq a) \rightarrow (a = b)$.
- **Bắc cầu**: nếu $\forall a, b, c \in A, \forall b \in A: (a\ R\ b) \wedge (b\ R\ c) \rightarrow a\ R\ c$
  Ví dụ: Quan hệ $<$ trong trường số $\mathbb{Z}$, i.e. nếu a nhỏ hơn b và b nhỏ hơn c thì a nhỏ hơn c.

Chúng ta có thể biểu diễn quan hệ của các phần tử trong một tập hợp bằng ít nhất 2 cách:

- Phương pháp 1: Sử dụng ma trận, dãy các vector để mô tả các quan hệ trong một tập hợp.
- Phương pháp 2: Thông qua một hàm (module) có 2 tham số là 2 phần tử của tập hợp. Nếu hai phần tử có quan hệ thì hàm (module) sẽ trả về giá trị là True (đúng), ngược lại là False (sai).

Phương pháp 1: **Biểu diễn quan hệ R của tập A** với tập B bằng ma trận

- Biểu diễn bằng ma trận, gọi là ma trận M với các phần tử là m_{ij} với i biểu thị dòng và j biểu thị cột.
- Mỗi dòng là từng phần tử của tập A.

- Mỗi cột là từng phần tử của tập B.
- Hai phần tử có quan hệ được mô tả bằng giá trị 1, ngược lại là giá trị 0.

Sinh viên thực hiện các lệnh thực tập sau:

>>> import numpy as np

Sinh viên tạo 1 tập tin Excel có tên là monhoc.xlsx và lưu vào ổ đĩa D: có nội dung như sau:

Môn học (A1)	**Lập trình cơ bản**	**Đại số**	**Toán rời rạc**	**Vật lý**	**Hóa học**	**Thể dục**
Môn học (A2)	*(Lưu ý: giá trị 1: là môn ở dòng cần được học trước môn ở cột)*					
Lập trình căn bản	0	1	1	0	0	0
Đại số	0	0	0	0	0	0
Toán rời rạc	0	0	0	0	0	1
Vật lý	0	0	0	0	0	0
Hóa học	0	0	0	0	0	0
Thể dục	1	1	1	1	1	0

Từ đó, GV hướng dẫn sinh viên đọc dữ liệu từ Excel thành ma trận các quan hệ theo bảng trên:

```
>>> import pandas
>>> df = pandas.read_excel('D:\\monhoc.xlsx')
>>> print (df.columns)  # liệt kê các cột dữ liệu
……
>>> giatri = df['<tên cột>'].values     # đọc 1 cột dữ liệu
……
```

Sau đó thực hiện các lệnh đọc dữ liệu vào mảng/ma trận của Python/numpy để xử lý tiếp tục các yêu cầu bên dưới, nghĩa là *xem xét các tính chất ma trận*.

Suy ra [*cách kiểm chứng ma trận quan hệ với các tính chất*]:

- Quan hệ phản xạ: ma trận vuông (do $A = B$) và đường chéo chính (\) bằng 1, i.e. $m_{ii} = 1, \forall i$
- Quan hệ đối xứng: các trị 0, 1 đối xứng qua đường chéo chính (\), i.e. $m_{ij} = m_{ji}, \forall i, j$
- Quan hệ phản xứng: ma trận vuông (do $A = B$) và các trị 0, 1 đối xứng qua đường chéo phụ (/), i.e. $m_{ij} = 0 \vee m_{ji} = 0 \rightarrow i \neq j$
- Quan hệ bắc cầu: $m_{ij} = 1, m_{jk} = 1 \rightarrow m_{ik} = 1$

Hãy viết chương trình bằng Python kiểm tra các mối quan hệ (cho trong một ma trận) có bị chồng chéo nhau hay không? Ví dụ: i phụ thuộc j, j phụ thuộc k và k phụ thuộc i.

Gợi ý: SV có thể xem lại Bài toán loan tin (Chương 3, Thực hành Đại số Tuyến tính)

Quan hệ tương đương: là quan hệ khi có các tính chất **phản xạ**, **đối xứng** và **bắc cầu**.

Khái niệm lớp tương đương: Ví dụ lớp tương đương modulo 8 chứa 3.

Quan hệ thứ tự: là quan hệ khi có các tính chất: phản xạ, phản xứng, bắt cầu.

Ví dụ: quan hệ $\leq$ trong số thực, số nguyên; quan hệ ước số trong số nguyên.

Với tập số nguyên dương:

- Quan hệ $\leq$ là quan hệ thứ tự toàn phần trong tập. Vì mọi số đều có quan hệ.
- Quan hệ "ước số" không phải quan hệ thứ tự nhưng không toàn phần vì tồn tại các phần tử không phải ước số của nhau.

Thứ tự từ điển:

Cho $(A, \preccurlyeq)$ và $(B, \preccurlyeq)$ là 2 tập thứ tự toàn phần. Thứ tự từ điển $\prec$ trên $A \times B$ như sau:

$$(a_1, b_1) \prec (a_2, b_2) \; nếu \; (a_1 < a_2) \vee (a_1 = a_2 \wedge b_1 < b_2)$$

Ví dụ:

Tập bảng chữ cái $\Sigma = \{a, b, c\}, với\ a < b < c$ và $\lambda \in \Sigma^*$ là chuỗi rỗng thì thứ tự từ điển là: $\Sigma^* = \{\lambda, a, aa, ab, ac, aaa, aab, aac, aba, abb, \dots, b,\ ba, bb, bc, \dots, c, ca, cb, cc, \dots\}$

Hoặc: $\Sigma = \{0,1\}$ với $0 < 1$ thì $0111110 \prec 10$

PHẦN 3 VÀ 4 SINH VIÊN ĐỌC ĐỂ CÓ KIẾN THỨC (BIẾT VỀ BÀI TOÁN)

3. [Đọc thêm] Bài toán ứng dụng 1: Biểu diễn cơ sở dữ liệu

Trong thực tế, quan hệ giữa các tập hợp được ứng dụng trong các mô hình lưu trữ dữ liệu. Ví dụ: hai tập hợp Sinh viên và Môn học sẽ có mối quan hệ với nhau. Cụ thể là mỗi ***Sinh viên*** sẽ có

quan hệ ***Học*** các ***Môn học***: **Học(Sinh viên, Môn học)**. Với mô tả này, chúng ta sẽ có 2 tập hợp cơ bản là Sinh viên và Môn học. Và dưới đây là một bảng dữ liệu minh họa các mô tả quan hệ Sinh viên và Môn học:

Môn học:	**Lập trình cơ bản**	**Đại số**	**Toán rời rạc**	**Vật lý**	**Hóa học**	**Thể dục**
Sinh viên	*(Lưu ý: giá trị 1: Sinh viên có học môn tương ứng; 0: ngược lại)*					
Văn Hậu	1	1	0	1	1	0
Quang Hải	0	0	1	1	0	1
Xuân Trường	1	1	0	0	1	1
Văn Đức	0	0	1	1	1	1
Đức Chinh	1	0	0	1	1	0
Văn Toàn	1	1	0	1	1	0

Tất nhiên, đối với mỗi tập, những quan hệ trong chính tập đó cũng là một thành phần cần biết. Với dữ liệu thực tế, chúng ta phải xem xét các quan hệ mô tả với sự hợp lý. Ví dụ: với Môn học, chúng ta có thể có quan hệ giữa các môn học là: ***Tiên quyết*** (môn cần học trước), như sau:

Môn học:	**Lập trình cơ bản**	**Đại số**	**Toán rời rạc**	**Vật lý**	**Hóa học**	**Thể dục**
Môn học	*(Lưu ý: giá trị 1: là môn ở dòng cần được học trước môn ở cột)*					
Lập trình căn bản	0	1	1	0	0	0
Đại số	0	0	0	0	0	0
Toán rời rạc	0	0	0	0	0	1
Vật lý	0	0	0	0	0	0
Hóa học	0	0	0	0	0	0
Thể dục	1	1	1	1	1	0

Với bảng trên, chúng ta phải điều chỉnh lại giá trị trong quan hệ Tiên quyết giữa 2 môn Toán rời rạc và Thể dục. Vì ở dòng môn 'Toán rời rạc' yêu cầu học trước môn 'Thể dục'. Tuy nhiên, ở dòng cuối môn 'Thể dục' cũng yêu cầu cần phải học trước môn 'Toán rời rạc'! Điều này có nghĩa là: ***Quan hệ môn Tiên quyết là quan hệ có thứ tự***. Tiên quyết để chỉ đến thời gian học trước và sau. Do vậy, 2 môn trên cần sửa đổi thứ tự để có điều kiện **phản xứng**! (Giả định điều chỉnh vị trí dòng Toán rời rạc, cột Thể dục sang giá trị 0)

Toán rời rạc	0	0	0	0	0	0

Hơn thế nữa, một số quan hệ sẽ mô tả sự liên quan của nhiều tập hợp. Ví dụ: Quan hệ Học được bổ sung thêm Học kỳ để biết được chính xác Sinh viên học môn học đó ở Học kỳ (bao gồm học kỳ, niên khóa), cụ thể quan hệ được tạo thành từ bộ 3 tập hợp: Học(Sinh viên, Môn học, Học kỳ).

Từ dữ liệu trên, chúng ta có thể tìm kiếm được những thông tin cần thiết. Dưới đây là một số truy vấn cơ bản trên dữ liệu mà chúng ta có thể xây dựng được:

- Sinh viên X đã học được những môn gì trong học kì này.
- Kết hợp với môn tiên quyết để suy ra sinh viên X đã học được tất cả những môn gì.

Như vậy, về cơ bản, sinh viên sẽ hiểu được phương pháp lưu trữ dữ liệu là thiết lập những quan hệ của các tập dữ liệu. Hiển nhiên, để thể hiện dữ liệu trong cơ sở dữ liệu, chúng ta còn nhiều kỹ thuật khác để giải quyết các vấn đề gặp phải trong: tìm kiếm, lưu trữ, thống kê,.... Các môn học như thiết kế cơ sở dữ liệu ở các học kỳ sau sẽ giúp các bạn sinh viên nắm rõ hơn. Ví dụ: các mô hình dữ liệu tuân theo các chuẩn 1, 2, 3,…; mô hình dữ liệu Nosql,…

4. [Đọc thêm] Bài toán ứng dụng 2: Hợp lý hóa điều kiện tìm kiếm trên dữ liệu

Việc hiểu rõ về cấu trúc dữ liệu, những quan hệ trong dữ liệu và bằng các biến đổi luận lý, chúng ta dễ dàng thay đổi các điều kiện tìm kiếm trên dữ liệu nhằm tăng tốc tính toán. Ứng dụng dưới đây liên quan đến luật De Morgan nổi tiếng:

$$\overline{A \cup B} = \bar{A} \cap \bar{B}$$

$$\overline{A \cap B} = \bar{A} \cup \bar{B}$$

với $\bar{A} = \{x \in U, x \notin A\}, A \subset U$

Chúng ta xét 2 bài toán sau:

- ***Kiểm tra sự "Tách rời" (disjoint) của hai tập đối tượng:***

Cho 2 đối tượng, mỗi đối tượng là một tập hợp. Phép kiểm hai đối tượng ***tách rời*** được định nghĩa như sau:

$$disjoint(A, B) \leftrightarrow \nexists x: (x \in A \wedge x \in B) \; với \; A, B \; là \; 2 \; tập \; hợp$$

Bài toán minh họa: Cho 3 tập dữ liệu Trẻ em (gọi là tập A), Phụ nữ/Giới nữ (B) và Bệnh nhân sởi (C). Hãy tìm ***tập những người không phải là bệnh nhân sởi mà là trẻ em hoặc phụ nữ***. Biểu thức toán học chúng ta cần tìm như sau:

$$!\,((A \cup B) \cap C)$$

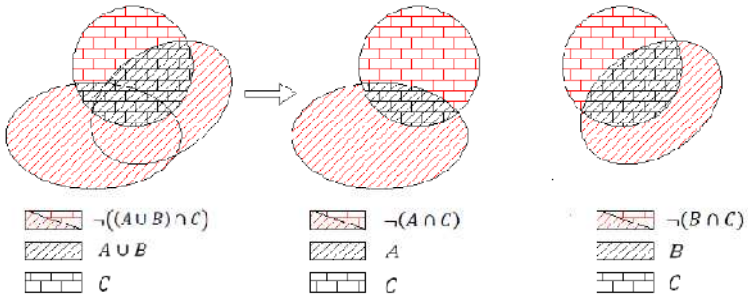
Biến đổi bằng luật phân phối, ta có:

$$!\,\big((A \cup B) \cap C\big) = !\,((A \cap C) \cup (B \cap C))$$

Áp dụng định luật De Morgan:

$$\ldots = !\,\big((A \cap C) \cup (B \cap C)\big) = !\,(A \cap C) \cap !\,(B \cap C)$$

Như vậy, về luận lý, chúng ta thấy rằng hai biểu thức: biểu thức đầu tiên $!(A \cup B) \cap C)$ và biểu thức thứ 2 $!(A \cap C) \cap !(B \cap C)$ là như nhau.



Tuy nhiên, trong thực tế, tùy thuộc vào dữ liệu, việc xử lý các biểu thức sẽ có độ phức tạp khác nhau và dẫn đến tốc độ xử lý khác nhau vì các lí do và giải thích sau:

- Việc tính toán tập $(A \cup B)$ sẽ tốn nhiều thời gian để xử lý do trên thực tế tập trẻ em và tập phụ nữ rất lớn. Khả năng chiếm trên 60% dữ liệu. Sau đó, với lượng dữ liệu đó, chúng ta thực hiện phép $\cap$ với tập C là tập bệnh nhân là một tập không lớn trên thực tế. Việc xử lý phải tuần tự từng bước.
- Hai phép xử lý $(A \cap C)$, $(B \cap C)$ và sau đó là phép lấy phủ định có thể độc lập với nhau nên việc xử lý hoàn toàn tách biệt. Các hệ thống có thể chia thành 2 tiến trình/luồng cùng lúc thực thi.

- ***Phép toán kiểm tra sự "Khác nhau" (difference) của hai tập đối tượng:***

Phép kiểm ***khác nhau*** được định nghĩa như sau:

$$difference(A, B) \leftrightarrow (x \in A \backslash B) \leftrightarrow (x \in A \wedge x \notin B) \text{ với } A, B \text{ là 2 tập hợp}$$

Bài toán minh họa: Cho 3 tập dữ liệu Trẻ em (gọi là tập A), Phụ nữ/Giới nữ (B) và Bệnh nhân sởi (C). Hãy tìm tập những Trẻ em không là Giới nữ mà không nhiễm bệnh sởi. Nghĩa là chúng ta cần tìm tập hợp sau:

$$A \backslash (B \backslash C)$$

Thay thế phép hiệu \ bằng phép $\cap$, ta có:

$$A \backslash (B \backslash C) = A \backslash (B \cap \bar{C})$$

Áp dụng định luật De Morgan cho vế phải đẳng thức trên, ta có:

$$\ldots = A \backslash (B \cap \bar{C}) = (A \backslash B) \cup (A \backslash \bar{C})$$

Áp dụng lần nữa thay thế phép hiệu \ bằng phép $\cap$, ta có:

$$\ldots = (A \backslash B) \cup (A \backslash \bar{C}) = (A \backslash B) \cup (A \cap C)$$

Như vậy, tương tự như trên, về luận lý, chúng ta thấy rằng hai biểu thức: biểu thức $A \backslash (B \backslash C)$ và biểu thức $(A \backslash B) \cup (A \cap C)$ là như nhau. Tuy nhiên, trong thực tế, tùy thuộc vào dữ liệu, việc xử lý các biểu thức sẽ có độ phức tạp khác nhau và dẫn đến tốc độ xử lý khác nhau vì các lí do và giải thích sau:

- Việc tính toán tập $A \backslash (B \backslash C)$ sẽ phải tuần tự thực hiện các bước.
- Hai phép xử lý $(A \backslash B)$ và $(A \cap C)$ có thể xử lý cùng lúc khi thực thi.

PHẦN DƯỚI ĐÂY GIẢNG VIÊN CUNG CẤP KIẾN THỨC CƠ BẢN, SINH VIÊN THỰC HÀNH VÀ THỬ NGHIỆM NHƯ BÀI TẬP VỀ NHÀ:

5. Khái niệm về lập trình logic

*Lưu ý 1: Phần này tập trung **giới thiệu** sinh viên **sự tồn tại và các khái niệm cơ bản nhất về lập trình và ngôn ngữ lập trình logic**. Chi tiết hơn để lập trình logic trên các ngôn ngữ như Prolog cũng như khai thác chi tiết các hàm trong gói thư viện PySWIP sẽ được học ở các môn chuyên ngành hoặc sinh viên có thể tự khám phá thêm. Do đó, phần này là phần bổ sung trong lớp học, chỉ khuyến khích học sinh nghiên cứu, tìm hiểu và tham khảo thêm chứ không bắt buộc.*

Lưu ý 2: Sinh viên nghiên cứu nên tự xem phương pháp cài đặt các gói thư viện phần mềm để làm thêm ở nhà.

5.1. Giới thiệu về lập trình logic và gói PySWIP

- **Khái niệm chương trình logic**

Từ nguyên lý cơ bản: *"Mọi thuật giải có thể biểu diễn bằng tập quy tắc với 3 cấu trúc: tuần tự, chọn và lặp (đệ quy)"*, một nhánh lập trình gọi là lập trình logic được ra đời. Lập trình logic có thể xem như việc thể hiện tập các luật, mỗi luật là sự kết hợp của các mệnh đề mệnh đề và mỗi mệnh đề là một dạng tân từ. Từ đó, người lập trình chỉ việc khai báo, còn việc xử lý luận lý để ra kết quả là do hệ thống.

Ví dụ: người lập trình khai báo luật: tổng 3 góc tam giác bằng 180, sau đó cho biết 2 góc thì hệ thống sẽ suy luận ra giá trị góc thứ 3.

Hiện tại, ngôn ngữ Prolog là ngôn ngữ lâu đời và nổi tiếng về lập trình logic.

- **Các thành phần trong chương trình logic:**

Tân từ có dạng như sau: $f(a_1, a_2, \dots, a_n)$ *là tân từ* $\leftrightarrow f: D_1 \times D_2 \times \dots \times D_n \rightarrow \{True, False\}$

Các D_i là các miền cho trước và $a_i \in D_i$. Mệnh đề (clause) là dạng các tân từ.

Các **luật** (rule) là tập hợp các mệnh đề: $p(\dots) :- q_1(\dots), q_2(\dots), \dots, q_n(\dots)$

Chương trình logic: $\{Rule: p_1(\dots), p_2(\dots), \dots\}$

Ví dụ: 2 mô tả quan hệ Ông với sự trợ giúp của mô tả quan hệ Cha mẹ như sau:

Ông(X, Y) :- Chame(Z, Y)

Ông(X, Y) :- Cha(X, Z), Chame(Z, Y)

Giải thích: ở luật thứ 1: nếu X là ông của Y thì tồn tại một Z là Cha hoặc Mẹ của Y; đồng thời luật thứ 2 mô tả là: X là **Cha** của Z hoặc Z có quan hệ **Cha mẹ** của Y (dấu , ở luật thứ 2).

- **Cấu trúc một chương trình logic:**

Một chương trình logic thường gồm 3 phần:

- **Phần khai báo các mệnh đề** (Predicates), khai báo các hằng, biến, tân từ,...
- **Phần thân chương trình** (Clauses): khai báo cụ thể các quy luật, quy tắc (dữ liệu).
- **Phần đích** (kết luận, Goal).

5.2. Cài đặt gói pyswip để minh họa các suy diễn luận lý trong Python

- **Giới thiệu về gói phần mềm PySWIP:**

PySWIP là một gói phần mềm Python làm cầu nối để thực thi các chương trình ngôn ngữ Prolog với những dòng lệnh theo dạng ngôn ngữ Python. Lõi của PySWIP là gói SWI-Prolog, một mã nguồn mở được phát triển từ năm 1987 được sử dụng nhiều trong web ngữ nghĩa.

Yêu cầu cài đặt:

* Python phiên bản 2.3 hoặc cao hơn.

* ctypes phiên bản 1.0 hoặc cao hơn.

* SWI-Prolog phiên bản 5.6.x hoặc cao hơn (phù hợp với các phiên bản khác).

* Thư viện chia sẻ libpl.

* Trên hệ thống Linux và Win32, trên tất cả các hệ POSIX.

- **Chi tiết cài đặt:**

Cài đặt bằng lệnh pip install. Pip install là 1 lệnh trong thư mục Scripts:

Từ thư mục Anaconda3 được cài đặt, thực hiện lệnh: **Scripts\pip install pyswip**

Ví dụ: Anaconda được cài đặt ở ổ **C:\Anaconda3** thì:

C:\Anaconda3> Scripts\pip install pyswip

```
C:\Anaconda3>scripts\pip install pyswip
Collecting pyswip
  Downloading https://files.pythonhosted.org/packages/60/cd/7defc05e52763f142865
45331b8e865ff829304befcaf22c4001f5706b77/pyswip-0.2.8-py2.py3-none-any.whl
Installing collected packages: pyswip
Successfully installed pyswip-0.2.8
```

Lưu ý: Nếu tập tin pip đã cũ (có phiên bản cập nhật mới thì sử dụng lệnh Python để cập nhật):

- Tập tin pip.exe đã cũ và được yêu cầu cập nhật mới:

```
You are using pip version 9.0.1, however version 19.0.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' comm
and.
```

- Thực hiện việc cập nhật phiên bản mới:

```
C:\Anaconda3>python -m pip install --upgrade pip
Cache entry deserialization failed, entry ignored
Collecting pip
  Using cached https://files.pythonhosted.org/packages/d8/f3/413bab4ff08e1fc4828
dfc59996d721917df8e8583ea85385d51125dceff/pip-19.0.3-py2.py3-none-any.whl
Installing collected packages: pip
  Found existing installation: pip 9.0.1
    Uninstalling pip-9.0.1:
      Successfully uninstalled pip-9.0.1
Successfully installed pip-19.0.3
```

Sinh viên có thể tham khảo thêm thông tin tại: https://github.com/f0ma/pyswip3

Trang web download: https://code.google.com/archive/p/pyswip/downloads

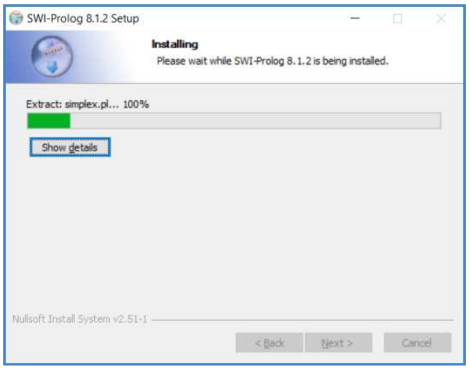
*Lưu ý: Trong Windows, gói pyswip có thể bị lỗi cài đặt (do thiếu thư viện libswipl), như sau:

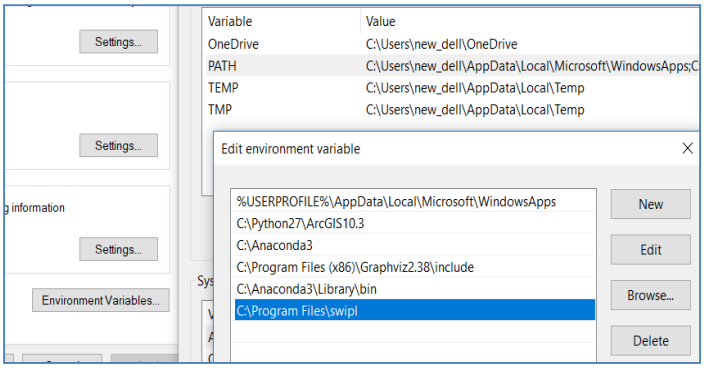
```
Command Prompt

C:\Anaconda3>Scripts\pip install pyswip
pip is configured with locations that require TLS/SSL, however the ssl module in Python is not available.
Requirement already satisfied: pyswip in c:\anaconda3\lib\site-packages (0.2.8)
pip is configured with locations that require TLS/SSL, however the ssl module in Python is not available.
Could not fetch URL https://pypi.org/simple/pip/: There was a problem confirming the ssl certificate: HTT
Max retries exceeded with url: /simple/pip/ (Caused by SSLError("Can't connect to HTTPS URL because the S
```

Khi đó, chúng ta nên kiểm tra việc thiết lập đường dẫn (path) đối với thư mục **Anaconda\Library\bin** và thực hiện việc cài đặt bổ sung gói **SWI-Prolog** tương thích với **Windows** tại địa chỉ: http://www.swi-prolog.org/download/devel (/devel hoặc /stable) cùng với việc thiết lập đường dẫn Path (tại Enviroment Variables cho thư mục cài đặt). Sau đó, chúng ta tắt và khởi động trình IDLE để tải lại các thư viện.

Hiển nhiên, chúng ta có thể xóa gói bằng lệnh **pip uninstall pyswip** và thực hiện cài đặt mới.





5.3. Minh họa sử dụng gói pyswip

Dưới đây là một số ví dụ đơn giản sử dụng gói pyswip.

Lưu ý:

- *Dữ liệu đều phải viết thường, không được viết hoa;*
- *Hạn chế khoảng trắng, ví dụ không có khoảng trắng: phaichet(X):-connguoi(X);*
- *Các từ khóa phải viết như mô tả, như: X, Y, What, Which.*

- **Ví dụ 1: Liệt kê dữ liệu tác giả và tác phẩm**

Sinh viên thực hành nhập các lệnh dưới đây:

>>> **from** pyswip **import** Prolog

>>> tgtph = **Prolog**() # khai báo quan hệ giữa tác giả tác phẩm là một đối tượng Prolog

>>> tgtph.**assertz**('tacgiatacpham(nguyendu,truyenkieu)') # nhập dữ liệu nguyendu, truyenkieu

>>> tgtph.**assertz**('tacgiatacpham(nguyendu,thanhhienthitap)')

>>> tgtph.**assertz**('tacgiatacpham(nguyendu,namtrungtapngam)')

>>> tgtph.**assertz**('tacgiatacpham(nguyendu,bachanhluctap)')

>>> list(tgtph.**query**('tacgiatacpham(nguyendu, **X**)')) # truy vấn kết quả đã nhập

…….. ← sinh viên điền kết quả.

……..

>>> for tg in tgtph.query('tacgiatacpham(X, Y)'): # truy vấn : liệt kê toàn bộ

print(tg["X"], ' viet ra tac pham: ', tg["Y"])

…….. ← sinh viên điền kết quả.

……..

……..

……..

- **Ví dụ 2: Suy luận tam đoạn luận:**

Tam đoạn luận là một cách suy luận trong diễn dịch.

Mệnh đề: 'Mọi người đều phải chết!'

Dữ liệu: Ông Micheal Jackson là người

Suy ra: ông Micheal Jackson phải chết

Sinh viên thực hiện các bước sau:

- Khai báo mệnh đề: phaichet(X) với X là đối tượng connguoi.
- Khai báo dữ liệu connguoi, như: micheal_jackson,… (bằng lệnh **assertz**)
- Tạo ra các truy vấn, nghĩa là kết quả: (Which) là con người? (What) phải chết?

```
>>> from pyswip import Prolog
>>> p = Prolog()
>>> p.assertz('phaichet(X):-connguoi(X)')
>>> p.assertz('connguoi(micheal_jackson)')
>>> p.assertz('connguoi(trinh_cong_son)')
>>> p.assertz('connguoi(vo_tong)')
>>> list(p.query('connguoi(Which)'))
```

…….. ← sinh viên điền kết quả.

```
>>> list(p.query('phaichet(What)'))
```

…….. ← sinh viên điền kết quả.