

**TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM  
TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG  
KHOA CÔNG NGHỆ THÔNG TIN**



**BÀI TẬP LỚN MÔN THỰC HÀNH CẤU TRÚC RỜI RẠC**

# **BA LAN NGƯỢC CHO PHÉP TÍNH LOGIC**

*Người hướng dẫn:* **GV. TRẦN HỒNG TÀI**

*Người thực hiện:* **TRẦN HOÀNG CUNG – 518H0602**

**Lớp : 18H50303**

**Khoá : 22**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

## MỤC LỤC

|   |    |
|---|----|
| LỜI CẢM ƠN.....   | 2  |
| CHƯƠNG 1 – GIỚI THIỆU.....  | 3  |
| 1.1 Thành viên:.....  | 3  |
| 1.2 Phân công công việc.....                                      | 3  |
| CHƯƠNG 2 – CƠ SỞ LÝ THUYẾT.....                                   | 4  |
| 2.1 Ký pháp Ba Lan(Reverse Polish notation):.....                 | 4  |
| 2.2 Các Logic cơ bản được sử dụng để tính toán bản chân trị:..... | 5  |
| CHƯƠNG 3 – THỰC NGHIỆM.....                                       | 7  |
| 3.1 Testcase 1.....   | 7  |
| 3.2 TestCase 2:.....  | 14 |
| CHƯƠNG 4 – KẾT QUẢ.....   | 24 |
| 4.1 Testcase.txt:.....  | 24 |
| 4.2 Test Case 1:.....   | 24 |
| 4.3 Test Case 2:.....   | 24 |
| 4.4 Test Case 3:.....   | 25 |
| 4.5 Test Case 4:.....   | 25 |
| 4.6 Test Case 5:.....   | 25 |
| TÀI LIỆU THAM KHẢO.....   | 26 |

## LỜI CẢM ƠN

Qua quá trình học tập và làm việc với thầy và các bạn trong bộ môn Cấu Trúc Rời Rạc, em đã học được rất nhiều kiến thức và kinh nghiệm cho bản thân. Em xin chân thành cảm ơn sự nhiệt tình của thầy để em có được kiến thức và tạo điều kiện cho em được tìm hiểu thêm nhiều thứ mới lạ. Do trình độ còn hạn hẹp nên trong quá trình làm bài tập lớn không tránh khỏi sai sót. Em mong nhận được sự chỉ dẫn và góp ý của thầy. Em xin chân thành cảm ơn

## CHƯƠNG 1 – GIỚI THIỆU

### 1.1 Thành viên

-Họ và Tên: *Trần Hoàng Cung*.

-Mã số sinh viên: *518H0602*

### 1.2 Phân công công việc

-*Tuần 1*: Tìm hiểu đề bài, định hướng cách giải quyết các vấn đề trong đề bài, tham khảo các tài liệu liên quan đến đề bài trên Internet hoặc tài liệu học tập;

-*Tuần 2*: Giải quyết vấn đề đầu tiên của bài toán về hàm Infix2Postfix(Infix), thực hiện giải Reverse Polish notation bằng phương pháp thủ công, xong thực hiện viết code bằng ngôn ngữ Python, kiểm tra và sửa lỗi.

-*Tuần 3*: Giải quyết hàm Postfix2TruthTable(Postfix), thực hiện tìm Truth Table từ đầu ra của hàm Infix2Postfix(Infix) bằng phương pháp thủ công, tiếp theo thực hiện code bằng ngôn ngữ Python, kiểm tra và sửa lỗi.

-*Tuần 4*: Kiểm tra tổng thể sources code bằng testcase, kiểm tra và sửa lỗi, thực hiện viết báo cáo

## CHƯƠNG 2 – CƠ SỞ LÝ THUYẾT

### 2.1 Ký pháp Ba Lan(Reverse Polish notation):

**Ký pháp Ba Lan** (tiếng Anh: *Polish notation*), còn gọi là ký pháp tiền tố (tiếng Anh: *prefix notation*), là một cách viết một biểu thức đại số rất thuận lợi cho việc thực hiện các phép toán. Đặc điểm cơ bản của cách viết này là không cần dùng đến các dấu ngoặc và luôn thực hiện từ trái sang phải.

#### Chuyển đổi từ biểu thức bình thường sang ký pháp Ba Lan:

Việc tính giá trị một biểu thức viết dưới dạng phép toán sau rất thuận tiện, tuy nhiên, theo thói quen thông thường, việc nhập biểu thức đó vào lại không dễ, người ta thường nhập vào một công thức dưới dạng thông thường (phép toán giữa) rồi dùng chương trình chuyển đổi nó sang dạng phép toán sau. Chúng ta hãy xét biểu thức trong ví dụ sau:

$$input = R|(P \& Q)$$

Ký hiệu biểu thức ghi dưới dạng phép toán sau là *output*. Trong quá trình chuyển đổi ta dùng một *stack* *S* để lưu các phần tử trong *output* chưa sử dụng đến. Khi đọc từ trái sang phải biểu thức *input* lần lượt có:

1. Đọc và ghi nhận giá trị R, ghi giá trị R vào *output*. Vậy *output* = "R".
2. Đọc toán tử "|". Đưa toán tử này vào stack *S*: *S* = "|"
3. Đọc dấu ngoặc mở "(", đưa dấu ngoặc này vào stack: *S* = "|(".
4. Đọc hạng tử P, đưa P vào *output*: *output* = "R P"
5. Đọc toán tử "&", đặt "&" vào stack: *S* = "|(&"
6. Đọc hạng tử "Q", đưa Q vào cuối *output*: *output* = "R P Q"
7. Đọc dấu ngoặc đóng ")". Lần lượt lấy các toán tử ở cuối stack ra khỏi stack đặt vào cuối *output* cho đến khi gặp dấu ngoặc mở "(" trong stack thì giải phóng nó: *S* = "|"; *output* = "R P Q &"
8. Đã đọc hết biểu thức *input*, lần lượt lấy các phần tử cuối trong stack đặt vào *output* cho đến hết. *output* = "R P Q & |".

Thuật toán chuyển từ ký pháp trung tố sang ký pháp tiền tố hoặc hậu tố. Một biểu thức chỉ gồm các phép toán hai ngôi bất kỳ luôn có thể được tính mà không cần dùng dấu ngoặc. Các phép toán ở trước nếu có độ ưu tiên (ưu tiên bởi toán tử hoặc bởi dấu ngoặc) thấp hơn một phép toán ở sau được đẩy vào một ngăn xếp (*stack*), chỉ khi nào các phép toán ưu tiên hơn ở sau được tính xong, các phép toán ở trước mới được xử lý.

## 2.2 Các Logic cơ bản được sử dụng để tính toán bản chân trị:

Trong toán học, khi có hai số, người ta dùng các phép toán số học (cộng, trừ, nhân, chia,...) tác động vào chúng để nhận được những số mới. Tương tự, khi có mệnh đề, người ta dùng các phép logic tác động vào chúng để nhận được những mệnh đề mới. Dưới đây ta trình bày định nghĩa và các tính chất cơ bản của các phép toán này.

### Phép Phủ Định (NOT):

*Phủ định* của mệnh đề  $a$  là một mệnh đề, ký hiệu là  $\neg a$ , đúng khi  $a$  sai và sai khi  $a$  đúng.

**Bảng giá trị chân trị của phép phủ định**

| $a$ | $\neg a$ |
|-----|----------|
| 1   | 0        |
| 0   | 1        |

### Phép Hội (AND):

Hội của hai mệnh đề  $a, b$  là một mệnh đề, đọc là  $a$  và  $b$ , ký hiệu  $a \wedge b$  (hoặc  $a.b$ ), đúng khi cả hai mệnh đề  $a, b$  cùng đúng và sai trong các trường hợp còn lại.

**Bảng giá trị chân trị của phép hội**

| $a$ | $b$ | $a \wedge b$ |
|-----|-----|--------------|
| 1   | 1   | 1            |
| 1   | 0   | 0            |
| 0   | 1   | 0            |
| 0   | 0   | 0            |

### Phép Tuyển (OR)

Tuyển của hai mệnh đề  $a, b$  là một mệnh đề đọc là  $a$  hoặc  $b$ , ký hiệu là  $a \vee b$  (hoặc  $a+b$ ), sai khi cả hai mệnh đề cùng sai và đúng trong trường hợp còn lại.

**Bảng giá trị chân trị của phép tuyển**

| <b>a</b> | <b>b</b> | <b>a v b</b> |
|----------|----------|--------------|
| 1        | 1        | 1            |
| 1        | 0        | 0            |
| 0        | 1        | 0            |
| 0        | 0        | 0            |

**Phép Kéo Theo (IMPLIES)**

*a kéo theo b* là một mệnh đề, ký hiệu là  $a \rightarrow b$ , chỉ sai khi *a đúng và b sai* và đúng trong các trường hợp còn lại.

**Bảng giá trị chân trị của phép kéo theo**

| <b>a</b> | <b>b</b> | <b>a <math>\rightarrow</math> b</b> |
|----------|----------|-------------------------------------|
| 1        | 1        | 1                                   |
| 1        | 0        | 0                                   |
| 0        | 1        | 1                                   |
| 0        | 0        | 1                                   |

**Phép Tương Đương(IF AND ONLY IF)**

*a tương đương b* là một mệnh đề, ký hiệu là  $a \leftrightarrow b$ , nếu cả hai mệnh đề *a* và *b* cùng đúng hoặc cùng sai

**Bảng giá trị chân trị của phép tương đương**

| <b>a</b> | <b>b</b> | <b>a <math>\leftrightarrow</math> b</b> |
|----------|----------|---|
| 1        | 1        | 1                                       |
| 1        | 0        | 0                                       |
| 0        | 1        | 0                                       |
| 0        | 0        | 1                                       |

## CHƯƠNG 3 – THỰC NGHIỆM

### 3.1 Testcase 1.

**Input:** 'R|(P&Q)'

**Infix2Posfix(Infix):**

Infix='R|(P&Q)'

Postfix=''

Stack=[]

Tạo hàm trả mức độ ưu tiên của toán tử:

```
def Uutien(inp):
    if inp=='~':
        return 5
    if inp=='&':
        return 4
    if inp=='|':
        return 3
    if inp=='>':
        return 2
    if inp=='=':
        return 1
    else:
        return 0
```

For i in Infix:

| i | Action                                   | Postfix | Stack |
|---|--|---------|-------|
| R | Đưa vào Postfix                          | R       | null  |
|   | Stack=null<br>->Đặt vào Stack            | R       |       |
| ( | Đặt vào Stack                            | R       | (     |
| P | Đưa vào Postfix                          | R P     | (     |
| & | -Stack!=null<br>-Uutien('&')>Uutien('(') | R P     | (&    |



|                               |  |         |    |
|-------------------------------|--|---------|----|
|                               | ->Đưa vào Stack  |         |    |
| Q                             | Đưa vào Postfix  | R P Q   | (& |
| )                             | -Đưa các phần tử trong stack ra Postfix đến khi gặp '('<br>-Xóa '(' khỏi Stack | R P Q & |    |
| End(đã kết thúc vòng lặp for) | Stack != null<br>->Đưa các phần tử còn lại trong Stack ra Posfix               | R P Q & |    |

-->Postfix ='RPQ&|'

### ***Postfix2TruthTable(Postfix):***

-Postfix='RPQ&|'

-Alpha=[],truth\_table=[],stack=[]

-Tạo hàm tính phép kéo theo:

```
def implies(a,b):
    if a:
        return b
    else:
        return True
```

-Đưa các kí tự là toán hạng vào trong Alpha:

```
for i in Postfix:
    if i.isalpha():
        Alpha.append(i)
```

-->Alpha=['R','P','Q']

-Loại bỏ các toán hạng trùng lặp trong Alpha bằng cách dùng Alpha=set(Alpha):

-->Alpha={'R','P','Q'}

-Tính số lượng toán hạng có trong Alpha:

amountOfAlpha=len(Alpha).

-->amountOfAlpha=3

-Tạo bảng giá trị True/False dựa trên số lượng toán hạng đã tính được

```
table=list(itertools.product([False,True],repeat=amountOfAlpha))
```

-->table=[(False, False, False), (False, False, True), (False, True, False), (False, True, True), (True, False, False), (True, False, True), (True, True, False), (True, True, True)]

```
container=[a for a in Alpha]
```

-->Container=['R','P','Q']

-Sắp xếp Container theo đúng bảng chữ cái:

```
Container.sort()
```

-->Container=['P','Q','R']

-Tạo các giá trị Logic cho các toán hạng đã có và đưa vào truth\_table:

```
for i in Container:
    position=Container.index(i)
    value=[a[position] for a in table]
    truth_table.append(value)
```

| i        | 'P'   | 'Q'  | 'R'  |
|----------|---|--|--|
| position | 0   | 1  | 2  |
| value    | [False,False,False,False, True, True, True, True] | [False,False,True,True, False,False,True,True] | [False,True,False,True, False,True,False,True] |

-->truth\_table=[[False, False, False, False, True, True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True]]

-Hàm trả về giá trị của toán hạng được đưa vào variable(alpha):



|   |   |  |   |
|---|---|--|---|
|   |   |  | True]]  |
| Q | Đưa giá trị của Q được trả về từ hàm variable('Q') vào stack  | [[False, True, False, True, False, True], [False, False, False, True, True, True], [False, False, True, True, True, True], [False, False, True, True, True, True]] | [[False, False, False, True, True, True], [False, False, True, True, True, True], [False, True, False, True, False, True], [False, True, True, True, True, True]] |
| & | <p>T1=stack.pop()<br/> -&gt;T1=[False, False, True, True, False, False, True, True]</p> <p>T2=stack.pop()<br/> -&gt;T2=[False, False, False, True, True, True, True, True].</p> <p>AND=[T2[i] and T1[i] for i in range(len(T1))].</p> <p>-&gt;AND=[False,</p> | [[False, True, False, True, False, True], [False, False, False, True, True, True], [False, False, True, True, True, True]]   | [[False, False, True, True, True, True], [False, False, True, True, True, True], [False, True, False, True, False, True], [False, True, True, True, True, True]]  |

|  |   |   |   |
|--|---|---|---|
|  | False, False,<br>False, False,<br>False, True, True]<br>Đưa AND vào<br>stack và<br>truth_table.   |   |   |
|  | T1=stack.pop()<br>->T1=[False,<br>False, False,<br>False, False,<br>False, True, True]<br>T2=stack.pop()<br>->T2=[False,<br>True, False, True,<br>False, True, False,<br>True]<br>OR=[T2[i] or<br>T1[i] for i in<br>range(len(T1))]<br>Đưa OR vào stack<br>và truth_table | [[False, True,<br>False, True, False,<br>True, True, True]] | [[False, False,<br>False, False, True,<br>True, True, True],<br>[False, False,<br>True, True, False,<br>False, True, True],<br>[False, True,<br>False, True, False,<br>True, False,<br>True],[False,<br>False, False,<br>False, False,<br>False, True,<br>True],[False,<br>True, False, True,<br>False, True, True,<br>True]] |

|                                |   |  |  |
|--------------------------------|---|--|--|
| End (đã kết thúc vòng lặp for) | <code>stack!=null</code><br><code>-&gt;Loại bỏ các phần tử thừa trong stack</code><br><code>(stack.pop()).</code> |  | <pre>[[False, False, False, False, True, True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True],[False, False, False, False, False, False, True, True],[False, True, False, True, False, True, True, True]]</pre> |
|--------------------------------|---|--|--|

-Kết thúc vòng lặp ta được:

`->truth_table=[[False, False, False, False, True, True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True],[False, False, False, False, False, False, True, True],[False, True, False, True, False, True, True, True]].`

-Đưa ra kết quả: `Truthtable=list(zip(*truth_table))`

**Output:** Truthtable=[(False, False, False, False, False), (False, False, True, False, True), (False, True, False, False, False), (False, True, True, False, True), (True, False, False, False, False), (True, False, True, False, True), (True, True, False, True, True), (True, True, True, True, True)] .

### 3.2 TestCase 2:

$\sim P|(Q\&R)>R$

***Infix2Postfix(Infix):***

Infix='~P|(Q&R)>R'

Postfix=''

Stack=[]

Tạo hàm trả mức độ ưu tiên của toán tử:

```
def Uutien(inp):
    if inp=='~':
        return 5
    if inp=='&':
        return 4
    if inp=='|':
        return 3
    if inp=='>':
        return 2
    if inp=='=':
        return 1
    else:
        return 0
```

For i in Infix:

| i | Action                        | Stack | Postfix |
|---|-------------------------------|-------|---------|
| ~ | Stack=null<br>->Đưa vào stack | ~     |         |
| P | Đưa đến Postfix               | ~     | P       |
|   | -Stack!=null                  |       | P~      |

|                                     |   |     |          |
|-------------------------------------|---|-----|----------|
|                                     | -Uutien(' ')<Uutien('~')<br>->Đưa '~' đến Postfix<br>-Stack==null<br>->Đưa   vào stack                |     |          |
| (                                   | Đưa vào stack   | (   | P~       |
| Q                                   | Đưa đến Posfix  | (   | P~Q      |
| &                                   | -stack !=null<br>-Uutien('&')>Uutien('(')<br>->Đưa vào stack  | ( & | P~Q      |
| R                                   | Đưa đến posfix  | ( & | P~QR     |
| )                                   | -Đưa các phần tử ra Posfix đến<br>khi gặp '('<br>-Xóa ( khỏi stack                                    |     | P~QR&    |
| >                                   | -Stack!=null<br>-Uutien(' ')<Uutien('~')<br>->Đưa '~' đến Postfix<br>Stack==null<br>->Đưa   vào stack | >   | P~QR&    |
| R                                   | Đưa đến Postfix   | >   | P~QR& R  |
| End(đã kết<br>thúc vòng<br>lặp for) | Stack != null<br>->Đưa các phần tử còn lại trong<br>stack ra Posfix                                   |     | P~QR& R> |

->Postfix ='P~QR&|R>'

***Postfix2Truthtable(Postfix):***

-Postfix='P~QR&|R>'

-Alpha=[],truth\_table=[],stack=[]



-Tạo hàm tính phép kéo theo:

```
def implies(a,b):
    if a:
        return b
    else:
        return True
```

-Đưa các kí tự là toán hạng vào trong Alpha:

```
for i in Postfix:
    if i.isalpha():
        Alpha.append(i)
```

-->Alpha=['P','Q','R','R']

-Loại bỏ các toán hạng trùng lặp trong Alpha bằng cách dùng Alpha=set(Alpha):

-->Alpha={'P','Q','R'}

-Tính số lượng toán hạng có trong Alpha:

```
amountOfAlpha=len(Alpha).
```

-->amountOfAlpha=3

-Tạo bảng giá trị True/False dựa trên số lượng toán hạng đã tính được

```
table=list(itertools.product([False,True],repeat=amountOfAlpha))
```

-->table=[(False, False, False), (False, False, True), (False, True, False), (False, True, True), (True, False, False), (True, False, True), (True, True, False), (True, True, True)]

```
Container=[a for a in Alpha]
```

-->Container=['P','Q','R']

-Sắp xếp Container theo đúng bảng chữ cái:

```
Container.sort()
```

-->Container=['P','Q','R']

-Tạo các giá trị Logic cho các toán hạng đã có và đưa vào truth\_table:

```
for i in Container:
    position=Container.index(i)
    value=[a[position] for a in table]
    truth_table.append(value)
```

| i        | 'P'   | 'Q'  | 'R'  |
|----------|---|--|--|
| position | 0   | 1  | 2  |
| value    | [False,False,False,False, True, True, True, True] | [False,False,True,True, False,False,True,True] | [False,True,False,True, False,True,False,True] |

-->truth\_table=[[False, False, False, False, True, True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True]]

-Hàm trả về giá trị của toán hạng được đưa vào variable(alpha):

```
def variable(alpha):
    for i in Container:
        if(alpha==i):
            position=Container.index(i)
            value=[a[position] for a in table]
            return value
```

-for i in Postfix:

| i | Action   | stack  | truth-table  |
|---|--|--|--|
| P | Đưa giá trị của P được trả về từ hàm variable('P') vào stack | [[False, False, False, False, True, True, True, True]] | [[False, False, False, False, True, True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True], |

|   |  |  |  |
|---|--|--|--|
|   |  |  | True]]   |
| ~ | <p>T=stack.pop()<br/>         -&gt;T=[[False, False, False, False, True, True, True]]<br/>         NOT=[not T[i] for i in range(len(T))]<br/>         -&gt;NOT=[True, True, True, True, True, False, False].<br/>         Đưa NOT vào stack và truth_table</p> | [[True, True, True, True, False, False, False, False]]   | [[False, False, False, False, True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True], [True, True, True, True, False, False, False, False]] |
| Q | Đưa giá trị của Q được trả về từ hàm variable('Q') vào stack   | [[True, True, True, True, False, False, False, False], [False, False, True, True, False, False, True, True]] | [[False, False, False, False, True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True], [True, True, True, True, False, False, False, False]] |
| R | Đưa giá trị của R được trả về từ hàm   | [[True, True, True, True, False, False, False, False]]   | [[False, False, False, False, True, True, True, True]]   |

|   |   |  |  |
|---|---|--|--|
|   | valiable('R') vào stack   | False, False],[False, False, True, True, False, False, True, True],[False, True, False, True, False, True, False, True]] | True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True], [True, True, True, True, False, False, False, False]]   |
| & | <p>T1=stack.pop()<br/> -&gt;T1=[False, True, False, True, False, True, False, True]<br/> T2=stack.pop()<br/> -&gt;T2=[False, False, True, True, False, False, True, True].<br/> AND=[T2[i] and T1[i] for i in range(len(T1))].<br/> -&gt;AND=[False, False, False, True, False, False, False, True]<br/> Đưa AND vào stack và truth_table</p> | [[True, True, True, True, False, False, False, False],[False, False, True, False, False, False, True, True]]             | [[False, False, False, False, True, True, True, True], [False, False, True, False, False, True, True, True], [False, True, False, True, False, True, False, True], [True, True, True, True, False, False, False, False], [False, False, False, True, False, True, False, True], [True, True, True, True, False, False, False, False], [False, False, False, True, False, True, False, True]] |
|   | T1=stack.pop()  | [[True, True, True,  | [[False, False,  |

|   |  |   |   |
|---|--|---|---|
|   | ->T1=[False, False,<br>False, True, False,<br>False, False, True]<br>T2=stack.pop()<br>->T2=[True, True,<br>True, True, False,<br>False, False, False].<br>OR=[T2[i] and<br>T1[i] for i in<br>range(len(T1))].<br>->OR=[True, True,<br>True, True, False,<br>False, False, True]<br>Đưa OR vào stack<br>và truth_table | True, False, False,<br>False, True]]  | False, False, True,<br>True, True, True],<br>[False, False, True,<br>True, False, False,<br>True, True], [False,<br>True, False, True,<br>False, True, False,<br>True],[True, True,<br>True, True, False,<br>False, False,<br>False],[False, False,<br>False, True, False,<br>False, False,<br>True],[True, True,<br>True, True, False,<br>False, False, True]] |
| R | Đưa giá trị của R<br>được trả về từ hàm<br>valiable('R') vào<br>stack  | [[True, True, True,<br>True, False, False,<br>False, True],[False,<br>True, False, True,<br>False, True, False,<br>True]] | [[False, False,<br>False, False, True,<br>True, True, True],<br>[False, False, True,<br>True, False, False,<br>True, True], [False,<br>True, False, True,<br>False, True, False,<br>True],[True, True,<br>True, True, False,<br>False, False,   |

|                 |   |  |  |
|-----------------|---|--|--|
|                 |   |  | False],[False, False, False, True, False, False, False, True],[True, True, True, True, False, False, False, True]]   |
| >               | <p>T1=stack.pop()<br/> -&gt;T1=[False, True, False, True, False, True, False, True]<br/> T2=stack.pop()<br/> -&gt;T2=[True, True, True, True, False, False, True].<br/> IF=[implies(T1[i],T2[i]) for i in range(len(T1))].<br/> -&gt;IF=[False, True, False, True, True, True, True, True]<br/> Đưa IF vào stack và truth_table</p> | [[False, True, False, True, True, True, True, True]] | [[False, False, False, False, True, True, True, True],[False, False, True, False, False, True, True], [False, True, False, True, False, True, False, True],[True, True, True, True, False, False, False, False],[False, False, False, True, False, False, False, False],[True, True, True, True, False, False, False, False],[True, True, True, True, True, True, True, True]] |
| End(đã kết thúc | stack!=null   |  | [[False, False,  |

|               |   |  |   |
|---------------|---|--|---|
| vòng lặp for) | ->Loại bỏ các phần tử thừa trong stack (stack.pop()). |  | False, False, True, True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True], [True, True, True, True, False, False, False, False], [False, False, False, True, False, False, False, True], [True, True, True, True, False, False, False, True], [False, True, False, True, True, True, True, True]] |
|---------------|---|--|---|

-Kết thúc vòng lặp ta được:

->Truth\_table=[[False, False, False, False, True, True, True, True], [False, False, True, True, False, False, True, True], [False, True, False, True, False, True, False, True], [True, True, True, True, False, False, False, False], [False, False, False, True, False, False, False, True], [True, True, True, True, False, False, False, True], [False, True, False, True, True, True, True, True]]

-Đưa ra kết quả:

```
Truthtable=list(zip(*truth_table))
```

**Output:** Truthtable=[(False, False, False, True, False, True, False), (False, False, True, True, False, True, True), (False, True, False, True, False, True, False), (False, True, True, True, True, True, True), (True, False, False, False, False, False, True), (True, False, True, False, False, False, True), (True, True, False, False, False, False, True), (True, True, True, False, True, True, True)].



## CHƯƠNG 4 – KẾT QUẢ

### 4.1 Testcase.txt:

```
R | (P&Q)
~P | (Q&R) > R
P | (R&Q)
(P > Q) & (Q > R)
(P | ~Q) > ~P = (P | (~Q)) > ~P
```

### 4.2 Test Case 1: $R|(P\&Q)$

Infix2Postfix(Infix):

```
RPQ&|
```

Postfix2Truthtable(Postfix):

|       |       |       |       |       |
|-------|-------|-------|-------|-------|
| False | False | False | False | False |
| False | False | True  | False | True  |
| False | True  | False | False | False |
| False | True  | True  | False | True  |
| True  | False | False | False | False |
| True  | False | True  | False | True  |
| True  | True  | False | True  | True  |
| True  | True  | True  | True  | True  |

### 4.3 Test Case 2: $\sim P|(Q\&R)>R$

-Infix2Postfix(Infix):

```
RPQ&|
```

-Postfix2Truthtable(Postfix):

|       |       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|-------|
| False | False | False | True  | False | True  | False |
| False | False | True  | True  | False | True  | True  |
| False | True  | False | True  | False | True  | False |
| False | True  | True  | True  | True  | True  | True  |
| True  | False | False | False | False | False | True  |
| True  | False | True  | False | False | False | True  |
| True  | True  | False | False | False | False | True  |
| True  | True  | True  | False | True  | True  | True  |

### 4.4 Test Case 3: $P|(R\&Q)$

-Infix2Postfix(Infix):

PRQ&|

-Postfix2TruthTable(Postfix):

|       |       |       |       |       |
|-------|-------|-------|-------|-------|
| False | False | False | False | False |
| False | False | True  | False | False |
| False | True  | False | False | False |
| False | True  | True  | True  | True  |
| True  | False | False | False | True  |
| True  | False | True  | False | True  |
| True  | True  | False | False | True  |
| True  | True  | True  | True  | True  |

#### 4.5 Test Case 4: $(P \supset Q) \& (Q \supset R)$

-Infix2Postfix(Infix):

PQ $\supset$ QR $\supset$ &

-Postfix2TruthTable(Postfix):

|       |       |       |       |       |       |
|-------|-------|-------|-------|-------|-------|
| False | False | False | True  | True  | True  |
| False | False | True  | True  | True  | True  |
| False | True  | False | True  | False | False |
| False | True  | True  | True  | True  | True  |
| True  | False | False | False | True  | False |
| True  | False | True  | False | True  | False |
| True  | True  | False | True  | False | False |
| True  | True  | True  | True  | True  | True  |

#### 4.6 Test Case 5: $(P | \sim Q) \supset \sim P = (P | (\sim Q)) \supset \sim P$

-Infix2Postfix(Infix):

PQ $\sim$  | P $\sim$   $\supset$  PQ $\sim$  | P $\sim$   $\supset$  =

-Postfix2TruthTable(Postfix):

|       |       |       |       |       |       |       |       |       |       |      |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|------|
| False | False | True  | True  | True  | True  | True  | True  | True  | True  | True |
| False | True  | False | False | True  | True  | False | False | True  | True  | True |
| True  | False | True  | True  | False | False | True  | True  | False | False | True |
| True  | True  | False | True  | False | False | False | True  | False | False | True |

## TÀI LIỆU THAM KHẢO

[https://en.wikipedia.org/wiki/Reverse\\_Polish\\_notation](https://en.wikipedia.org/wiki/Reverse_Polish_notation)

[https://en.wikipedia.org/wiki/Shunting-yard\\_algorithm](https://en.wikipedia.org/wiki/Shunting-yard_algorithm)

<https://toidicode.com/cac-ham-xu-ly-chuoi-trong-python-368.html>

[https://vi.wikipedia.org/wiki/K%C3%ADp\\_h%C3%A1p\\_Ba\\_Lan](https://vi.wikipedia.org/wiki/K%C3%ADp_h%C3%A1p_Ba_Lan)

[https://vi.wikipedia.org/wiki/M%E1%BB%87nh\\_%C4%91%E1%BB%81\\_to%C3%A1n\\_h%E1%BB%8Dc](https://vi.wikipedia.org/wiki/M%E1%BB%87nh_%C4%91%E1%BB%81_to%C3%A1n_h%E1%BB%8Dc)