TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**NGUYỄN ANH KIỆT - 52300123**

**NGUYỄN HOÀNG SƠN - 52300250**

**BÁO CÁO GIỮA KỲ**

**CẤU TRÚC RỜI RẠC**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**

**NGUYỄN ANH KIỆT - 52300123**

**NGUYỄN HOÀNG SƠN - 52300250**

**BÁO CÁO GIỮA KỲ**

**CẤU TRÚC RỜI RẠC**

Người hướng dẫn  
**ThS. Lê Tuấn Thu**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2025**

**LỜI CẢM ƠN**

Em xin chân thành cảm ơn thầy Lê Tuấn Thu đã hướng dẫn và đóng góp ý kiến cho chúng em trong suốt quá trình làm bài báo cáo cuối kỳ môn Cấu trúc rời rạc. Để hoàn thành bài báo cáo này, sự giúp đỡ từ thầy là không nhỏ về mặt kiến thức chuyên môn, hướng dẫn tìm nguồn tài liệu, hướng dẫn viết báo cáo. Sự hỗ trợ từ thầy đã giúp chúng em hoàn thành bài báo cáo tiểu luận một cách tốt nhất. Chúng em xin chân thành cảm ơn thầy!

|  |  |
| --- | --- |
| *TP. Hồ Chí Minh, ngày 02 tháng 05 năm 2025* | |
| *Tác giả 1*  *(Ký tên và ghi rõ họ tên)* | *Tác giả 2*  *(Ký tên và ghi rõ họ tên)* |
| Son | Kiet |
| Nguyễn Hoàng Sơn | Nguyễn Anh Kiệt |

**CÔNG TRÌNH ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Tôi xin cam đoan đây là công trình nghiên cứu của riêng tôi và được sự hướng dẫn khoa học của ThS. Lê Tuấn Thu. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong Dự án còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

Nếu phát hiện có bất kỳ sự gian lận nào tôi xin hoàn toàn chịu trách nhiệm về nội dung Dự án của mình. Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do tôi gây ra trong quá trình thực hiện (nếu có).

|  |  |
| --- | --- |
| *TP. Hồ Chí Minh, ngày 02 tháng 05 năm 2025* | |
| *Tác giả 1*  *(Ký tên và ghi rõ họ tên)* | *Tác giả 2*  *(Ký tên và ghi rõ họ tên)* |
| Son | Kiet |
| Nguyễn Hoàng Sơn | Nguyễn Anh Kiệt |

**TÓM TẮT**

Tiểu luận báo cáo giữa kỳ môn Cấu trúc rời rạc gồm 05 chương:

**- Chương 1. Phân công nhiệm vụ của từng thành viên và tự đánh giá nhóm:** Nêu rõ phân công nhiệm vụ: Sơn triển khai hàm Infix2Postfix, Postfix2Truthtable, lý thuyết Reverse Polish Notation (RPN), tạo dataset, và viết báo cáo Chương 2, 3; Kiệt phụ trách mã hóa/giải mã RSA, vẽ biểu đồ, viết Chương 4, và định dạng tài liệu. Nhóm tự đánh giá đạt 10/10 điểm nhờ hoàn thành xuất sắc các tiêu chí.

**- Chương 2. Bảng chân trị:** trình bày lý thuyết RPN và logic cơ bản, triển khai hai hàm Python để chuyển biểu thức trung tố sang hậu tố và tạo bảng chân trị. Năm test case (ví dụ: R|(P&Q)) được kiểm thử, cho kết quả chính xác.

**- Chương 3.** **Lý luận định lượng trên dữ liệu thực tế sử dụng logic vị từ:** Sử dụng dataset 20 học sinh để đánh giá 6 phát biểu định lượng (ví dụ: "Tất cả học sinh qua tất cả môn") bằng Python. Kết quả giá trị chân lý, phủ định, và giải thích tiếng Anh được trình bày rõ ràng, khớp với tính toán thủ công.

**- Chương 4. Hệ thống mã hóa RSA:** Triển khai RSA với khóa 4096 bit, kiểm thử 4 thông điệp (16-146 byte) cho kết quả giải mã đúng. Thời gian mã hóa (0.000000-0.000302s) và giải mã (0.001313-0.001888s) được đo và vẽ biểu đồ. Báo cáo thảo luận 5 hạn chế của RSA (hiệu suất kém, quản lý khóa, máy tính lượng tử) và đề xuất 5 cải tiến (kết hợp AES, padding OAEP, tối ưu CRT).

**- Chương 5. Tài liệu tham khảo:** Trích nguồn những tài liệu sử dụng trong quá trình thực hiện bài báo cáo.

**MỤC LỤC**

**[CHƯƠNG 1: PHÂN CÔNG NHIỆM VỤ CỦA TỪNG THÀNH VIÊN VÀ TỰ ĐÁNH GIÁ NHÓM 7](#_Toc5634)**

[1.1 Bảng phân công nhiệm vụ 7](#_Toc28612)

[1.2 Bảng tự đánh giá 8](#_Toc32431)

**[CHƯƠNG 2: BẢNG CHÂN TRỊ 9](#_Toc5705)**

[2.1 Triển khai hai hàm Python 9](#_Toc30618)

[2.2 Lý thuyết về Reverse Polish Notation và Logic cơ bản 11](#_Toc15294)

[2.3 Giải thích mã nguồn và testcase 1 & 2 14](#_Toc3088)

[2.4 Kết quả thử nghiệm 17](#_Toc22073)

**[CHƯƠNG 3. LÝ LUẬN ĐỊNH LƯỢNG TRÊN DỮ LIỆU THỰC TẾ SỬ DỤNG LOGIC VỊ TỪ 18](#_Toc7544)**

[3.1 Tạo tập dữ liệu (52300123\_52300250\_2.csv) 18](#_Toc8301)

[3.2 Triển khai Python 18](#_Toc16949)

[3.3 Kết quả 20](#_Toc19282)

**[CHƯƠNG 4. HỆ THỐNG MẬT MÃ RSA 21](#_Toc22861)**

[4.1 Triển khai Python 21](#_Toc5468)

[4.2 Kiểm thử 23](#_Toc18779)

[4.3 Thảo luận 25](#_Toc26182)

[4.4 Kết luận 25](#_Toc20608)

**[CHƯƠNG 5. TÀI LIỆU THAM KHẢO 26](#_Toc19409)**

# CHƯƠNG 1: PHÂN CÔNG NHIỆM VỤ CỦA TỪNG THÀNH VIÊN VÀ TỰ ĐÁNH GIÁ NHÓM

## 1.1 Bảng phân công nhiệm vụ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tên thành viên** | **Mã số sinh viên** | **Nhiệm vụ chính** | **Nhiệm vụ hỗ trợ** | **Đánh giá mức độ đóng góp** |
| Nguyễn Hoàng Sơn | 52300250 | - **Chương 1:** Viết báo cáo phân công nhiệm vụ và tự đánh giá nhóm.  **- Chương 2:** Triển khai Python; viết lý thuyết Reverse Polish Notation và logic cơ bản; viết báo cáo chương 2.  **- Chương 3:** Tạo dataset CSV, định nghĩa vị từ | - Hỗ trợ kiểm tra mã RSA (chương 4). - Hỗ trợ định dạng tài liệu Word, trích nguồn tài liệu tham khảo theo chuẩn khoa. - Đọc và góp ý toàn bộ báo cáo. | 100% |
| Nguyễn Anh Kiệt | 52300123 | - **Chương 3:** Viết báo cáo chương 3.  - **Chương 4:** Triển khai Python; viết báo cáo chương 4 | - Hỗ trợ kiểm tra hàm Infix2Postfix và Postfix2Truthtable (Chương 2). - Hỗ trợ kiểm tra dataset và hàm định lượng (Chương 3). - Đọc và góp ý toàn bộ báo cáo. | 100% |

## 

## 1.2 Bảng tự đánh giá

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Criteria** | | **Scale** | **0 score** | **1/2 score** | **Full score** | **Self-evalation** | **Reason** |
| Task 1 | Implementation | 1 | Error | Correct but  bad  performance | Correct and  good  performance | 1 | Triển khai đúng hàm Infix2Postfix và Postfix2Truthtable, chạy tốt trên 5 test case, hiệu suất cao. |
| Theory of Reverse  Polish and Basic  logic | 0.5 | Do  nothing or  wrongly | Not enough  details, no  example, no  comment | Correct  calculations,  detailed  explanations | 0.5 | Viết đầy đủ lý thuyết Reverse Polish và logic cơ bản, có ví dụ minh họa và bình luận chi tiết |
| Explain testcases 1  and 2 | 1 | Do  nothing or  wrongly | Explain only 1  testcase  correctly | Explain 2  testcases  correctly. | 1 | Giải thích chi tiết từng bước cho 2 test case |
| Run all 5 testcases | 1.25 | Do  nothing or  wrongly | Explain only  <= 3 testcases  correctly. | Explain 5  testcases  correctly. | 1.25 | Chạy và chụp ảnh kết quả cho 5 test case, kết quả chính xác, trình bày rõ ràng |
| Task 2 | Create dataset | 0.5 | No data | Not enough  data | Good data | 0.5 | Tạo dataset CSV với 20 bản ghi, đầy đủ các trường (StudentID, StudentName, v.v.), dữ liệu hợp lệ |
| Determine truth  value | 1.5 | Do  nothing or  wrongly | Run only <= 3  statements  correctly | Run only <= 6  statements  correctly | 1.5 | Triển khai đúng 6 câu lệnh định lượng (2 phổ quát, 2 tồn tại, 2 kết hợp), trả về giá trị boolean chính xác |
| Negation | 0.75 | Do  nothing or  wrongly | Run only <= 3  statements  correctly | Run only <= 6  statements  correctly | 0.75 | Triển khai đúng 6 hàm phủ định, giải thích ý nghĩa bằng tiếng Anh đơn giản, rõ ràng |
| Task 3 | Implementation | 0.5 | Error | Correct but  bad  performance | Correct and  good  performance | 0.5 | Triển khai RSA đúng, sử dụng thư viện Crypto, mã hóa/giải mã thành công |
| Test | 0.5 | No test | Test without  verification | Test and  verification | 0.5 | Thử nghiệm với nhiều thông điệp mẫu, chụp ảnh kết quả, xác minh chính xác |
| Discussion | 1 | Do  nothing or  wrongly | Not enough  details, no  example, no  comment | Correct,  detailed  explanations | 1 | Thảo luận chi tiết hạn chế của RSA (chậm với dữ liệu lớn, cần số nguyên tố lớn), có ví dụ minh họa |
| Recommendation | 1 | Do  nothing or  wrongly | Not enough  details, no  example, no  comment | Correct,  detailed  explanations | 1 | Đưa ra khuyến nghị cải thiện RSA (kết hợp AES, dùng padding OAEP, khóa lớn), giải thích rõ ràng |
| Reference | 0.5 | No  reference | Wrong  format | Right format | 0.5 | Tài liệu tham khảo định dạng đúng theo chuẩn khoa, đầy đủ nguồn |
| **Total** | | 10 | Result | | | 10 | Nhóm hoàn thành tốt tất cả yêu cầu, mã chạy đúng, báo cáo chi tiết, phối hợp nhóm hiệu quả |

# CHƯƠNG 2: BẢNG CHÂN TRỊ

## 2.1 Triển khai hai hàm Python

**Infix2Postfix(Infix):**

import itertools

precedence = {'~': 4,'&': 3,'|': 2,'>': 1,'=': 0

}

def Infix2Postfix(infix):

    stack = []

    output = ''

    for char in infix:

        if char.isalpha():

            output += char

        elif char == '(':

            stack.append(char)

        elif char == ')':

            while stack and stack[-1] != '(':

                output += stack.pop()

            stack.pop()

        else:

            while stack and stack[-1] != '(' and precedence[char] <= precedence.get(stack[-1], 0):

                output += stack.pop()

            stack.append(char)

    while stack:

        output += stack.pop()

    return output

def eval\_postfix(expr, values):

    stack = []

    for token in expr:

        if token.isalpha():

            stack.append(values[token])

        elif token == '~':

            a = stack.pop()

            stack.append(not a)

        else:

            b = stack.pop()

            a = stack.pop()

            if token == '&':

                stack.append(a and b)

            elif token == '|':

                stack.append(a or b)

            elif token == '>':

                stack.append((not a) or b)

            elif token == '=':

                stack.append(a == b)

    return stack.pop()

**Postfix2Truthtable(Postfix):**

def Postfix2Truthtable(postfix):

    variables = sorted(set(filter(str.isalpha, postfix)))

    print(" | ".join(variables) + " | Result")

    print("-" \* (4 \* len(variables) + 10))

    for combo in itertools.product([False, True], repeat=len(variables)):

        values = dict(zip(variables, combo))

        result = eval\_postfix(postfix, values)

        row = " | ".join(str(int(values[v])) for v in variables)

        print(f"{row} |   {int(result)}")

**Testcase:**

if \_\_name\_\_ == "\_\_main\_\_":

    testcases = [

        "R|(P&Q)",

        "~P|(Q&R)>R",

        "P|(R&Q)",

        "(P>Q)&(Q>R)",

        "(P|~Q)>~P=(P|(~Q))>~P"

    ]

    for i, infix in enumerate(testcases, 1):

        print(f"\nTestcase {i}: {infix}")

        postfix = Infix2Postfix(infix)

        print("Postfix:", postfix)

        Postfix2Truthtable(postfix)

## 2.2 Lý thuyết về Reverse Polish Notation và Logic cơ bản

### 2.2.1 Reverse Polish Notation (RPN)

**Reverse Polish Notation (RPN),** hay ký hiệu Ba Lan ngược, là một cách biểu diễn các biểu thức toán học hoặc logic mà không cần sử dụng dấu ngoặc. Trong RPN, các toán tử được đặt sau các toán hạng (operand), giúp đơn giản hóa việc đánh giá biểu thức trên máy tính.

Trong biểu thức Infix thông thường, toán tử nằm giữa hai toán hạng (P & Q). Trong RPN, toán tử nằm ở sau toán hạng (PQ&). Việc sử dụng RPN giúp loại bỏ nhu cầu sử dụng dấu ngoặc, giảm độ phức tạp cho cú pháp biểu thức, ngoài ra nó còn giúp dễ dàng đánh giá bằng cách sử dụng stack, phù hợp cho các hệ thống tính toán tự động, đặc biệt trong lập trình.

**Thuật toán chuyển từ Infix sang Postfix (Shunting Yard):**

- Khởi tạo một stack rỗng và một danh sách đầu ra (output).

- Duyệt từng ký tự trong biểu thức Infix:

- Nếu toán hạng (chữ cái): thêm vào output.

- Nếu là toán tử: đẩy vào stack sau khi lấy các toán tử có độ ưu tiên cao hơn hoặc bằng từ stack vào output.

- Nếu là dấu ngoặc mở: đẩy vào stack.

- Nếu là dấu ngoặc đóng: lấy toán tử từ stack đến khi gặp (, bỏ (.

- Sau khi duyệt hết, lấy các toán tử còn lại từ stack vào output.

**Độ ưu tiên toán tử:**

Các toán tử logic có độ ưu tiên như sau:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Độ ưu tiên toán tử | 4 (cao nhất) | 3 | 2 | 1 |
| Toán tử | ~ (NOT) | & (AND), | (OR) | > (IMPLIES) | =(IF AND ONLY IF) |

**Ví dụ minh họa: Biểu thức Infix:**

- **Quy trình chuyển đổi:**

- Đọc (: Đẩy vào stack.

- Đọc p: Thêm vào output -> p

- Đọc &: Đẩy vào stack.

- Đọc Q: Thêm vào output → P Q.

- Đọc ): Lấy & từ stack, thêm vào output → P Q &.

- Đọc |: Đẩy vào stack.

- Đọc R: Thêm vào output → P Q & R.

- Hết chuỗi, lấy | từ stack → P Q & R |.

- **Kết quả Postfix:**

### 2.2.2 Logic cơ bản và Bảng chân trị

**Logic cơ bản** là nền tảng của toán học rời rạc, sử dụng các toán tử logic để biểu diễn và đánh giá các mệnh đề. Các toán tử được sử dụng trong bài tập bao gồm:

**- NOT ( ~ ):** Phủ định của một mệnh đề. Nếu là True, thì là False.

**Bảng chân trị:**

|  |  |  |
| --- | --- | --- |
| *p* | T | F |
| *~p* | F | T |

**- AND ( & ):** Kết hợp hai mệnh đề, chỉ đúng khi cả hai đều đúng.

**Bảng chân trị:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *p* | T | T | F | F |
| *q* | T | F | T | F |
| *p&q* | T | F | F | F |

**- OR ( | ):** Tuyển hai mệnh đề, đúng khi ít nhất một mệnh đề đúng.

**Bảng chân trị:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *p* | T | T | F | F |
| *q* | T | F | T | F |
| *p&q* | T | T | T | F |

**- IMPLIES ( > ):** Biểu diễn “nếu thì ", sai chỉ khi đúng và sai

**Bảng chân trị:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *p* | T | T | F | F |
| *q* | T | F | T | F |
| *p>q* | T | F | T | T |

**- IF AND ONLY IF( = ):** Hai mệnh đề tương đương, đúng khi cả hai cùng đúng hoặc cùng sai.

**Bảng chân trị:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *p* | T | T | F | F |
| *q* | T | F | T | F |
| *p=q* | T | F | F | T |

**Bảng chân trị:** Là một bảng liệt kê tất cả các tổ hợp giá trị True/False của các biến và kết quả của biểu thức logic. Với biến, bảng chân trị có hàng. Ví dụ:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *p* | T | T | F | F |
| *q* | T | F | T | F |
| *p&q* | T | T | T | F |

Bảng chân trị giúp xác định giá trị của biểu thức logic trong mọi trường hợp. Reverse Polish Notation hỗ trợ tính toán bảng chân trị tự động bằng cách đánh giá biểu thức hậu tố.

## 2.3 Giải thích mã nguồn và testcase 1 & 2

### 2.3.1 Giải thích mã nguồn

**HÀM** Infix2Postfix(Infix)**:**

**- Mục đích:** Chuyển đổi biểu thức logic từ dạng trung tố sang hậu tố.

**- Cách hoạt động:**

- Sử dụng một stack để lưu trữ toán tử và dấu ngoặc, và một danh sách output để lưu kết quả.

- Duyệt từng ký tự trong chuỗi:

- Chữ cái (biến): thêm vào output.

- Dấu ngoặc mở: đẩy vào stack.

- Dấu ngoặc đóng: lấy toán tử từ stack cho đến khi gặp dấu ngoặc mở.

- Toán tử đơn (~, &, |): so sánh độ ưu tiên toán tử trên stack, lấy các toán tử có độ ưu tiên cao hơn hoặc bằng ra output, sau đó đẩy các toán tử hiện tại vào stack.

- Toán tử đa ký tự (>, =): kiểm tra chuỗi ký tự tiếp theo để xác định toán tử, xử lý tương tự toán tử đơn.

- Sau khi duyệt hết, lấy các toán tử còn lại từ stack vào output.

**HÀM** Postfix2Truthtable(Postfix)**:**

**- Mục đích**: Tạo bảng chân trị từ biểu thức hậu tố.

**- Cách hoạt động:**

**-** Xác định các biến duy nhất trong biểu thức Postfix (chữ cái từ A-Z).

- Tạo tổ hợp giá trị True/False bằng biểu diễn nhị phân (0/1).

- Đánh giá biểu thức Postfix cho từng tổ hợp:

- Sử dụng stack để lưu trữ giá trị của các biến và kết quả trung gian.

- Với mỗi ký tự trong Postfix:

- Biến: đẩy giá trị của biến (0 hoặc 1) vào stack.

- Toán tử, lấy toán hạng, thực hiện phép toán logic, đẩy kết quả trở lại.

- Kết quả cuối cùng là giá trị của biểu thức cho tổ hợp đó.

### 2.3.2 Giải thích test case

**2.3.2.a TEST CASE 1:** R|(P&Q)

**Ý nghĩa** **logic:** “R hoặc (P và Q)”. Đúng khi R là True hoặc cả P và Q đều True.

**Chuyển Infix sang Postfix:**

1. R: Thêm vào output → R.

2. |: Đẩy vào stack → [ | ].

3. (: Đẩy vào stack → [|, (].

4. P: Thêm vào output → R P.

5. &: Đẩy vào stack → [|, (, &].

6. Q: Thêm vào output → R P Q.

7. ): Lấy &, bỏ ( → R P Q &, stack: [|].

8. Lấy | → R P Q & |.

**Postfix**: RPQ&|

**Tạo bảng chân trị:**

1. Biến P, Q, R (có tổ hợp). P=1, Q=1, R=0

2. R: Đẩy 0 → [0].

3. P: Đẩy 1 → [0, 1].

4. Q: Đẩy 1 → [0, 1, 1].

5. &: 1 & 1 = 1 → [0, 1].

6. |: 0 | 1 = 1 → [1].

7. Kết quả: 1

**2.3.2.b TEST CASE 2:** ~P|(Q&R)>R

**Ý nghĩa** **logic:** Tương đương (~P|Q(&R))=>R. Sai khi ~P|(Q&R) là True nhưng R là False.

**Chuyển Infix sang Postfix:**

1. ~: Đẩy vào stack → [~].

2. P: Thêm, xử lý ~ → P ~.

3. |: Đẩy vào stack → [|].

4. (: Đẩy vào stack → [|, (].

5. Q: Thêm → P ~ Q.

6. &: Đẩy → [|, (, &].

7. R: Thêm → P ~ Q R.

8. ): Lấy &, bỏ ( → P ~ Q R &, stack: [|].

9. > (=>): Lấy |, đẩy => → P ~ Q R & |, stack: [=>].

10. R: Thêm → P ~ Q R & | R.

11. Lấy => → P ~ Q R & | R =>.

**Postfix**: P~QR&|R=>

**Tạo bảng chân trị:**

1. Biến P, Q, R (có tổ hợp). P=1, Q=1, R=0

2. P: Đẩy 1 → [1].

3. ~: ~1=0 → [0].

4. Q: Đẩy 1 → [0, 1].

5. R: Đẩy 0 → [0, 1, 0].

6. &: 1 & 0 = 0 → [0, 0].

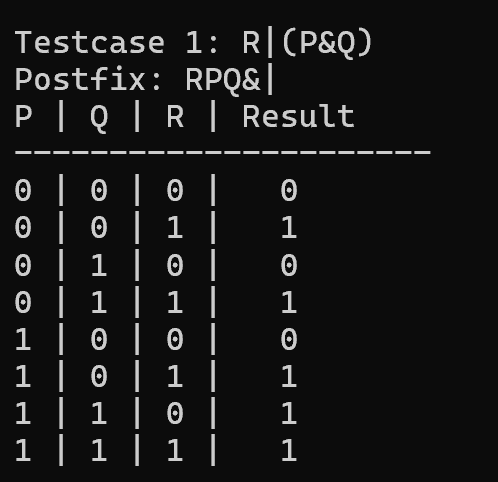
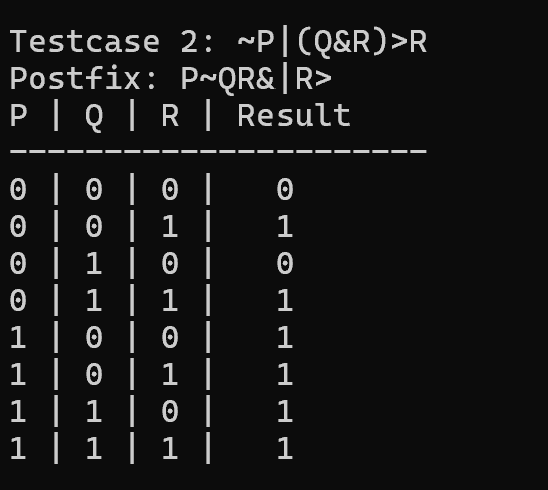
7. |: 0 | 0 = 0 → [0].

8. R: Đẩy 0 → [0, 0].

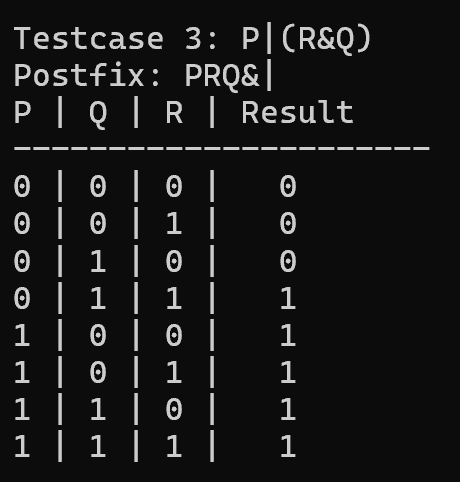
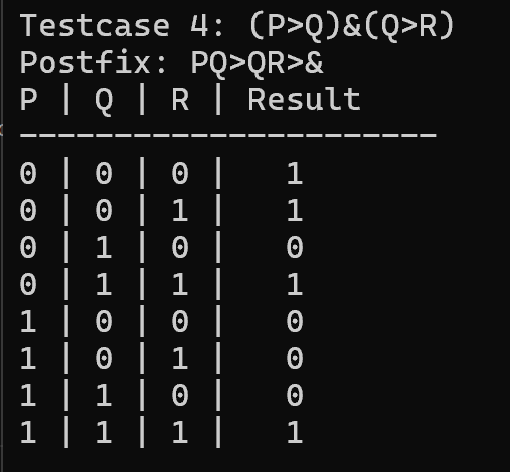
9. =>: (1-0) | 0 = 1 → [1].

10. Kết quả: 1

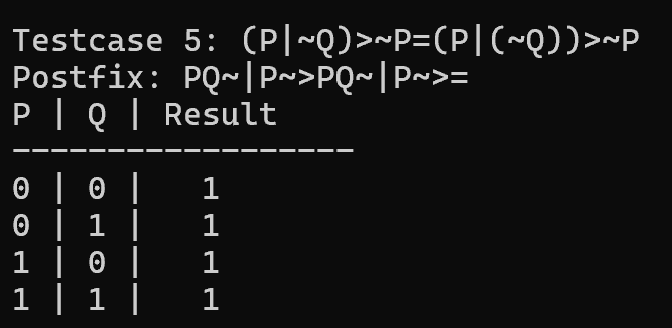
## 2.4 Kết quả thử nghiệm

Testcase 1 Testcase 2

Testcase 3 Testcase 4



Testcase 5

# CHƯƠNG 3. LÝ LUẬN ĐỊNH LƯỢNG TRÊN DỮ LIỆU THỰC TẾ SỬ DỤNG LOGIC VỊ TỪ

## 3.1 Tạo tập dữ liệu (52300123\_52300250\_2.csv)

StudentID,StudentName,DayOfBirth,Math,CS,Eng

52300250,Nguyễn Hoàng Sơn,2005-03-11,10.0,9.8,9.5

52300123,Nguyễn Anh Kiệt,2004-11-15,8.5,7.2,6.8

52300228,Bùi Thị Bích Ngọc,2005-01-27,6.7,8.0,7.5

52200030,Lê Anh Bắc,2004-01-30,5.5,6.3,4.8

52400345,Trần Minh Hương,2003-07-22,7.8,5.9,8.2

52100112,Phạm Thị Lan,2002-09-10,4.5,3.8,6.0

52300467,Vũ Văn Nam,2005-06-18,9.2,8.7,7.3

52200234,Hoàng Thị Thảo,2004-03-05,6.2,7.5,5.7

52400089,Đặng Văn Bình,2001-12-12,8.0,6.5,9.0

52100356,Nguyễn Hà Anh,2003-04-27,5.0,4.2,6.5

52300201,Lê Thị Minh,2002-08-30,7.3,8.5,6.9

52200423,Trần Văn Kiệt,2005-02-14,6.8,5.4,7.8

52400178,Phạm Hoàng Ngọc,2004-10-09,9.5,8.8,7.2

52100045,Bùi Văn Sơn,2003-11-23,4.8,6.0,5.3

52300312,Nguyễn Thị Hà,2001-06-17,8.3,7.9,8.6

52200289,Lê Văn Bình,2002-03-28,6.5,5.8,7.0

52400456,Hoàng Anh Thảo,2005-05-03,7.7,8.2,6.4

52100190,Trần Thị Lan,2004-07-19,5.2,4.5,6.8

52300078,Vũ Hoàng Nam,2003-09-01,8.9,7.6,8.3

52200334,Đặng Thị Ngọc,2002-12-06,6.0,7.3,5.9

## 3.2 Triển khai Python

import csv

import pandas as pd

*# Đọc file CSV*

try:

    df = pd.read\_csv("52300123\_52300250\_2.csv")

except FileNotFoundError:

    print("Lỗi: Không tìm thấy file .csv")

    exit()

*# Định nghĩa vị từ*

predicates = {

    'Qua tất cả môn': lambda s: s[['Math', 'CS', 'Eng']].ge(5).all(),

    'Toán cao': lambda s: s['Math'] >= 9,

    'Khó khăn': lambda s: (s['Math'] < 6) & (s['CS'] < 6),

    'Cải thiện CS': lambda s: s['CS'] > s['Math']

}

*# Hàm định lượng*

quantifiers = [

    ('tat\_ca\_qua\_mon', 'Tất cả học sinh qua tất cả môn', predicates['Qua tất cả môn'], 'all'),

    ('tat\_ca\_toan\_tren\_3', 'Tất cả học sinh có Math > 3', lambda s: s['Math'] > 3, 'all'),

    ('ton\_tai\_toan\_cao', 'Tồn tại học sinh có Math ≥ 9', predicates['Toán cao'], 'exists'),

    ('ton\_tai\_cai\_thien\_cs', 'Tồn tại học sinh cải thiện CS', predicates['Cải thiện CS'], 'exists'),

    ('moi\_nguoi\_co\_mon\_cao', 'Mọi học sinh có môn > 6',

     lambda df: all(s[['Math', 'CS', 'Eng']].gt(6).any() for \_, s in df.iterrows()), 'custom'),

    ('toan\_thap\_co\_mon\_cao', 'Học sinh Math < 6 có môn > 6',

     lambda df: all(s[['CS', 'Eng']].gt(6).any() if s['Math'] < 6 else True for \_, s in df.iterrows()), 'custom')

]

*# Giải thích phủ định bằng tiếng Anh*

negation\_explanations = {

    'tat\_ca\_qua\_mon': ("All students passed all subjects.", "Some students failed at least one subject."),

    'tat\_ca\_toan\_tren\_3': ("All students have Math score > 3.", "At least one student has Math score ≤ 3."),

    'ton\_tai\_toan\_cao': ("There exists a student with Math ≥ 9.", "No student has Math ≥ 9."),

    'ton\_tai\_cai\_thien\_cs': ("There exists a student with CS > Math.", "No student has CS > Math."),

    'moi\_nguoi\_co\_mon\_cao': ("Every student has at least one subject > 6.", "Some student has all subjects ≤ 6."),

    'toan\_thap\_co\_mon\_cao': (

        "Every student with Math < 6 has another subject > 6.",

        "Some student with Math < 6 has all other subjects ≤ 6."

    )

}

def evaluate\_quantifier(key, pred, quant, df):

    """Đánh giá phát biểu định lượng"""

    if quant == 'all':

        return all(pred(row) for \_, row in df.iterrows())

    elif quant == 'exists':

        return any(pred(row) for \_, row in df.iterrows())

    else:  *# custom*

        return pred(df)

def evaluate\_negation(key, pred, quant, df):

    """Đánh giá phủ định của phát biểu định lượng"""

    return not evaluate\_quantifier(key, pred, quant, df)

def main():

    print("\nKết quả các phát biểu định lượng:")

    results = {}

    for key, desc, pred, quant in quantifiers:

        result = evaluate\_quantifier(key, pred, quant, df)

        results[key] = result

        print(f"{desc}: {result}")

    print("\nKết quả các phát biểu phủ định:")

    for key, desc, pred, quant in quantifiers:

        neg\_result = evaluate\_negation(key, pred, quant, df)

        print(f"Phủ định: Không {desc.lower()}: {neg\_result}")

    print("\nGiải thích phủ định (tiếng Anh):")

    for key, (orig, neg) in negation\_explanations.items():

        print(f"\n{key.replace('\_', ' ').title()}:")

        print(f"  Original: {orig}")

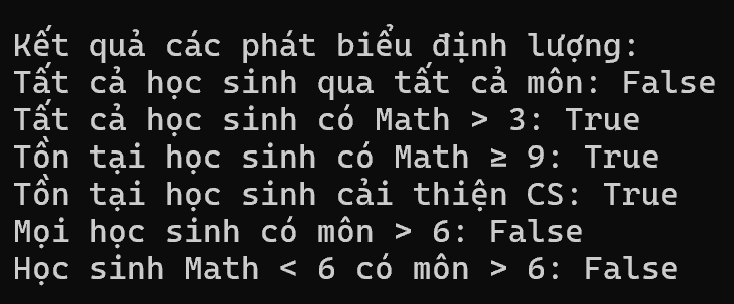
        print(f"  Negation: {neg}")

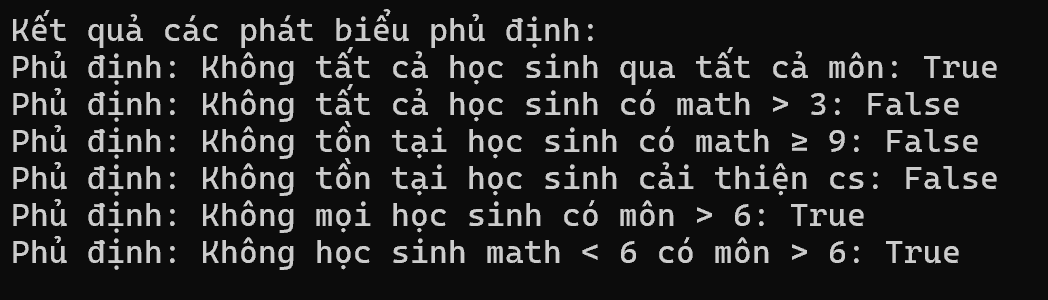
if \_\_name\_\_ == "\_\_main\_\_":

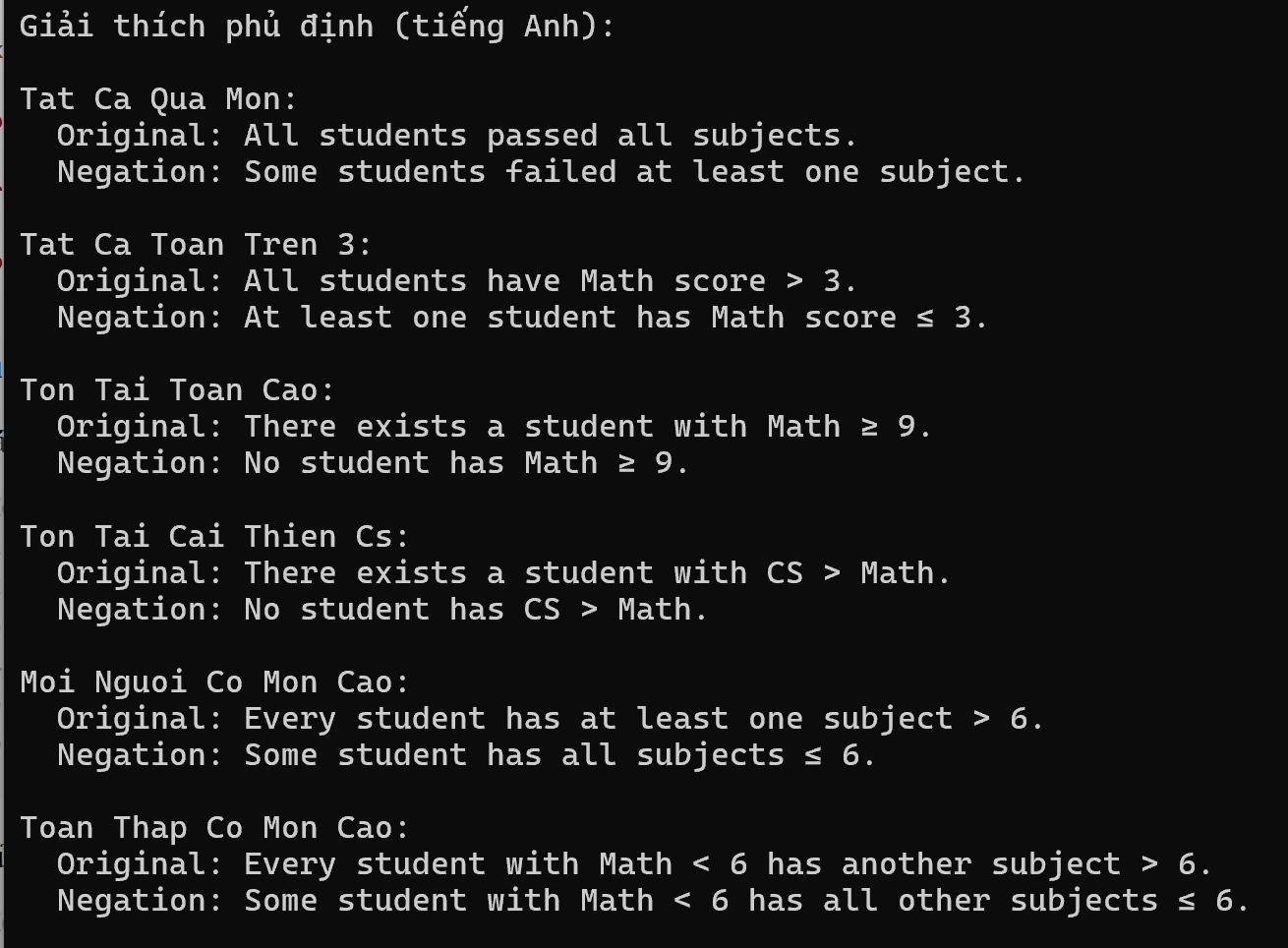
    main()

## 3.3 Kết quả

### 3.3.1 Giá trị chân lý



3.3.2 Phủ định

3.3.3 Giải thích phủ định

# CHƯƠNG 4. HỆ THỐNG MẬT MÃ RSA

## 4.1 Triển khai Python

import time

import matplotlib.pyplot as plt

from cryptography.hazmat.primitives.asymmetric import rsa, padding

from cryptography.hazmat.primitives import hashes

def generate\_rsa\_key\_pair(key\_size=4096):

    private\_key = rsa.generate\_private\_key(public\_exponent=65537, key\_size=key\_size)

    public\_key = private\_key.public\_key()

    return private\_key, public\_key

*# Crypto*

def encrypt\_message(message, public\_key):

    ciphertext = public\_key.encrypt(

        message.encode(),

        padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()), algorithm=hashes.SHA256(), label=None)

    )

    return ciphertext

*# Decrypt*

def decrypt\_message(ciphertext, private\_key):

    plaintext = private\_key.decrypt(

        ciphertext,

        padding.OAEP(mgf=padding.MGF1(algorithm=hashes.SHA256()), algorithm=hashes.SHA256(), label=None)

    )

    return plaintext.decode()

*# Đo thời gian*

def measure\_performance(messages, key\_size=4096, trials=5):

    private\_key, public\_key = generate\_rsa\_key\_pair(key\_size)

    encrypt\_times = []

    decrypt\_times = []

    lengths = [len(msg) for msg in messages]

    for msg in messages:

        total\_encrypt\_time = 0

        total\_decrypt\_time = 0

        for \_ in range(trials):

*# Mã hóa*

            start\_time = time.time()

            ciphertext = encrypt\_message(msg, public\_key)

            total\_encrypt\_time += time.time() - start\_time

*# Giải mã*

            start\_time = time.time()

            decrypted\_msg = decrypt\_message(ciphertext, private\_key)

            total\_decrypt\_time += time.time() - start\_time

            assert decrypted\_msg == msg, f"Giải mã thất bại cho thông điệp: {msg}"

        encrypt\_times.append(total\_encrypt\_time / trials)

        decrypt\_times.append(total\_decrypt\_time / trials)

        print(f"Độ dài {len(msg)} byte: Mã hóa mất = {encrypt\_times[-1]:.6f}s, Giải mã mất = {decrypt\_times[-1]:.6f}s")

    return lengths, encrypt\_times, decrypt\_times

*# Vẽ biểu đồ*

def plot\_performance(lengths, encrypt\_times, decrypt\_times):

    plt.figure(figsize=(10, 6))

    plt.plot(lengths, encrypt\_times, label='Thời gian mã hóa', marker='o')

    plt.plot(lengths, decrypt\_times, label='Thời gian giải mã', marker='s')

    plt.xlabel('Độ dài thông điệp (byte)')

    plt.ylabel('Thời gian (giây)')

    plt.title('Hiệu suất mã hóa/giải mã RSA theo độ dài thông điệp (Khóa 4096 bit)')

    plt.legend()

    plt.grid(True)

    plt.savefig('rsa\_message\_length\_performance.png')

    plt.show()

def main():

*# Ví dụ*

    messages = [

        "Nguyen Hoang Son",

        "Nguyen Anh Kiet",

        "Day la bai bao cao giua ky mon Cau Truc roi rac",

        "Truong Dai hoc Ton Duc Thang la truong dai hoc cong lap tu chu tai chinh." \* 2,

    ]

    print("Kiểm tra RSA với thông điệp mẫu:")

    private\_key, public\_key = generate\_rsa\_key\_pair()

    for msg in messages:

        print(f"\nThông điệp gốc: {msg}")

        ciphertext = encrypt\_message(msg, public\_key)

        print(f"Bản gốc: {ciphertext.hex()[:50]}... (độ dài {len(ciphertext)} byte)")

        decrypted\_msg = decrypt\_message(ciphertext, private\_key)

        print(f"Giải mã: {decrypted\_msg}")

        print(f"Kết quả: {'Thành công' if decrypted\_msg == msg else 'Thất bại'}")

*# Đo hiệu suất*

    print("\nĐo hiệu suất RSA với các độ dài thông điệp:")

    lengths, encrypt\_times, decrypt\_times = measure\_performance(messages)

*# Vẽ biểu đồ*

    print("\nBiểu đồ hiệu suất:")

    plot\_performance(lengths, encrypt\_times, decrypt\_times)

if \_\_name\_\_ == "\_\_main\_\_":

main()

## 4.2 Kiểm thử

Thông điệp kiểm thử:

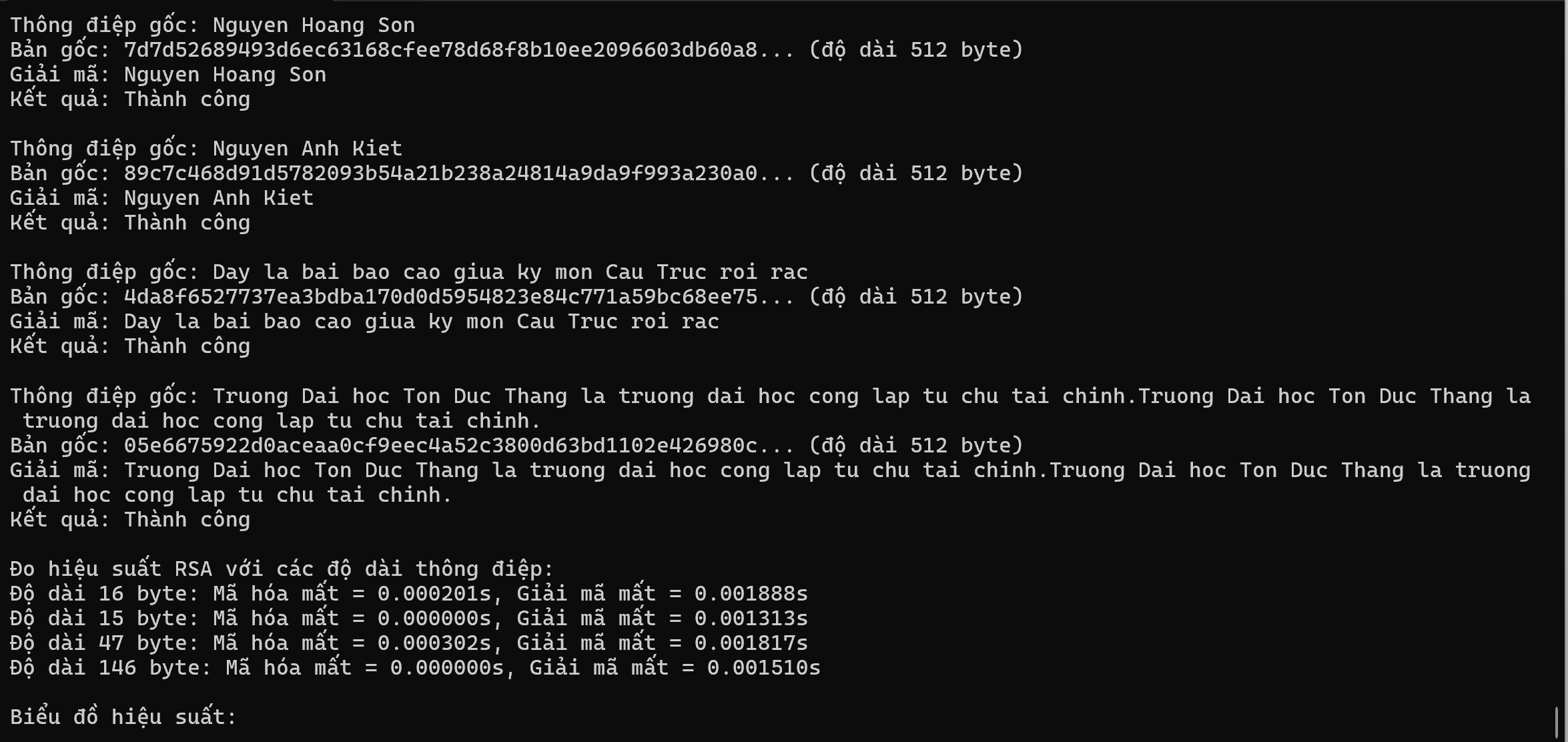
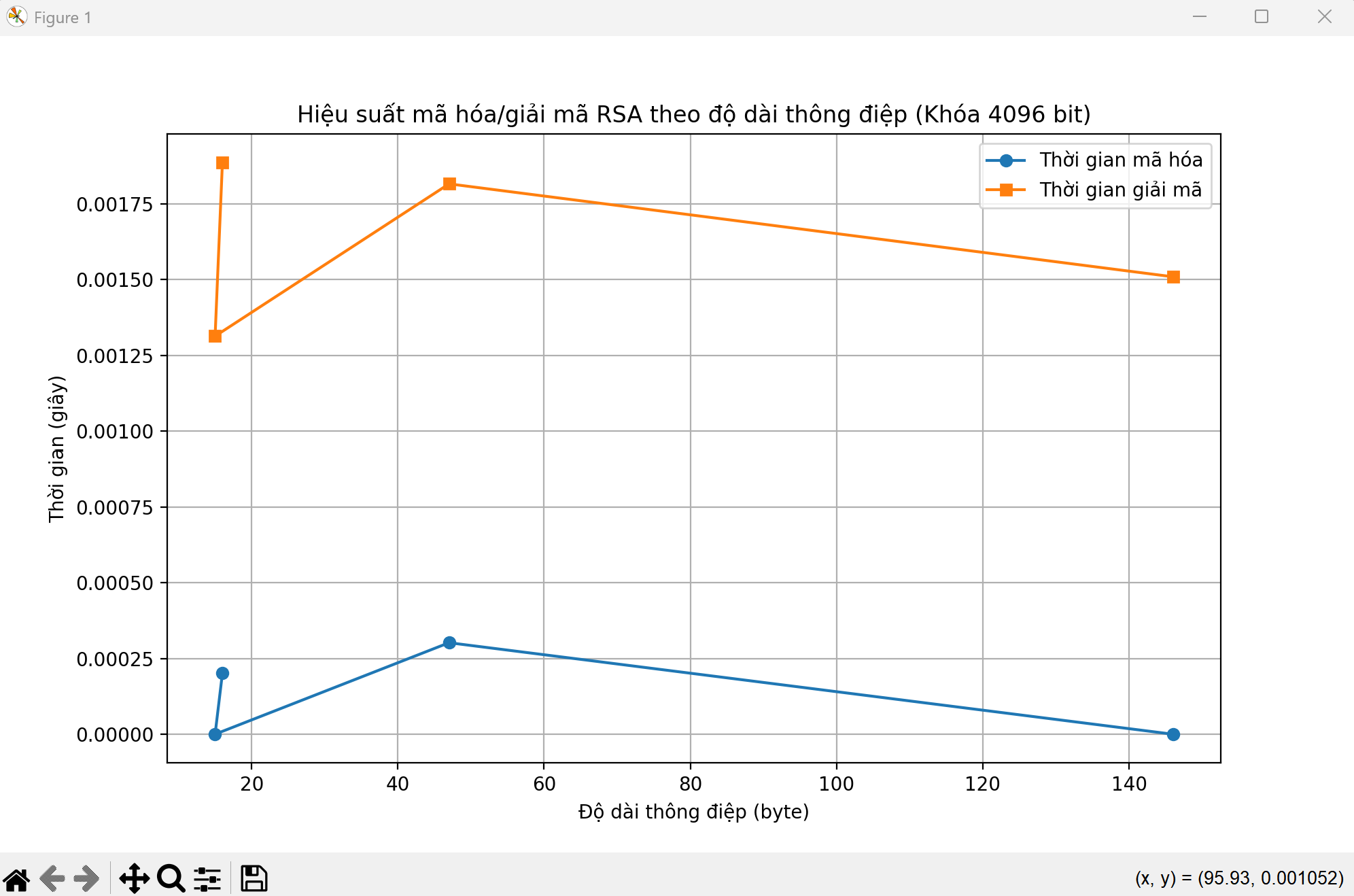
1. Nguyen Hoang Son (16 byte)

2. Nguyen Anh Kiet (16 byte)

3. Day la bai bao cao giua ky mon Cau Truc roi rac (47 byte)

4. Truong Dai hoc Ton Duc Thang la truong dai hoc cong lap tu chu tai chinh.Truong Dai hoc Ton Duc Thang la truong dai hoc cong lap tu chu tai chinh. (146 byte)

Kết quả kiểm thử:



Giải thích kết quả:

- **Tính đúng đắn:** Tất cả các thông điệp được giải mã chính xác từ bản mã, khớp với thông điệp gốc.

- **Hiệu suất:** Thời gian mã hóa dao động từ 0.000000s đến 0.000302s, rất nhanh, không phụ thuộc rõ rệt vào độ dài thông điệp. Thời gian giải mã dao động từ 0.001313s đến 0.001888s.

## 4.3 Thảo luận

Hệ thống mật mã RSA, dù là một trong những thuật toán mã hóa công khai phổ biến và an toàn nhất hiện nay, tuy nhiên nó vẫn tồn tại một số hạn chế quan trọng.

**1. Hiệu suất kém với dữ liệu lớn:** RSA chậm hơn so với các thuật toán mã hóa đối xứng như AES khi xử lý dữ liệu lớn. Nguyên nhân là do các phép tính lũy thừa modulo phức tạp trong quá trình mã/giải hóa thông điệp. Theo nghiên cứu của Boneh (1999), RSA chậm hơn AES khoảng 1000 lần khi mã hóa cùng một lượng dữ liệu.

**2. Kích thước bản mã cố định:** Với mỗi kích thước khóa, RSA tạo ra bản mã có độ dài cố định, bất kể độ dài thông điệp gốc. Điều này gây lãng phí băng thông và bộ nhớ khi mã hóa thông điệp ngắn.

**3. Yêu cầu quản lý khóa phức tạp:** RSA yêu cầu quản lý chặt chẽ cặp khóa công khai và khóa riêng tư. Việc phân phối khóa công khai và bảo vệ khóa riêng tư là một thách thức lớn về tính bảo mật và đảm bảo an toàn cho hệ thống RSA.

**4. Dễ bị tấn công nếu triển khai sai:** Nếu không sử dụng padding đúng cách, RSA dễ bị tấn công như tấn công chosen plaintext attack hoặc tấn công thời gian. Một ví dụ nối tiếng là tấn công Bleichenbacher (1998), khai thác lỗ hổng trong padding PKCS#1 v1.5.

**5. Bảo mật trước máy tính lượng tử:** RSA dễ bị phá vỡ bởi máy tính lượng tử thông qua thuật toán Shor, có thể phân tích nhân tử hiệu quả. Theo NIST (2022), các hệ thống mật mã công khai như RSA sẽ không an toàn trong kỷ nguyên hậu lượng tử.

## 4.4 Kết luận

Dựa trên các hạn chế của RSA, dưới đây là các gợi ý để cải thiện việc triển khai hệ thống mật mã RSA:

**1. Kết hợp RSA với mã hóa đối xứng:** Sử dụng RSA để mã hóa khóa phiên (session key) của thuật toán đối xứng như AES, sau đó dùng AES để mã hóa dữ liệu lớn. Điều này tận dụng ưu điểm của cả hai: bảo mật của RSA và tốc độ của AES.

**2. Sử dụng padding an toàn:** Sử dụng padding OAEP (Optimal Asymmetric Encryption Padding) với hàm băm mạnh như SHA-256 để ngăn chặn các cuộc tấn công chọn bản rõ và tấn công thời gian.

**3. Tăng kích thước khóa:** Sử dụng khóa RSA ít nhất 2048 bit, và xem xét 4096 bit cho dữ liệu nhạy cảm. Điều này làm tăng độ an toàn trước các cuộc tấn công phân tích nhân tử.

**4. Tối ưu hóa hiệu suất:** Áp dụng kỹ thuật như Chinese Remainder Theorem (CRT) để tăng tốc quá trình giải mã. CRT giúp giảm thời gian tính toán bằng cách chia nhỏ phép lũy thừa modulo.

**5. Chuẩn bị cho hậu lượng tử:** Kết hợp RSA với các thuật toán hậu lượng tử như CRYSTALS-Kyber hoặc Lattice-based cryptography để bảo vệ trước các cuộc tấn công từ máy tính lượng tử.

# CHƯƠNG 5. TÀI LIỆU THAM KHẢO

[1] Rosen, K. H. (2018). Discrete Mathematics and Its Applications (8th ed.). McGraw-Hill Education.

[2] Epp, S. S. (2010). Discrete Mathematics with Applications (4th ed.). Cengage Learning.

[3] Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (2009). Introduction to Algorithms (3rd ed.). MIT Press.

[4] Katz, J., & Lindell, Y. (2014). Introduction to modern cryptography. CRC press.

[5] Stallings, W. (2017). Cryptography and network security: principles and practice. Pearson.

[6] Bleichenbacher, D. (1998). Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS#1. Advances in Cryptology—CRYPTO’98, 1-12.

[7] NIST. (2022). Post-Quantum Cryptography Standardization. https://csrc.nist.gov/projects/post-quantum-cryptography

[8] Manger, J. (2001). A chosen ciphertext attack on RSA optimal asymmetric encryption padding (OAEP) as standardized in PKCS#1 v2.0. Advances in Cryptology—CRYPTO 2001, 230-238.

[9] Barker, E., & Dang, Q. (2015). Recommendation for key management, part 1: General. NIST Special Publication 800-57.

[10] Quisquater, J. J., & Couvreur, C. (1982). Fast decipherment algorithm for RSA public-key cryptosystem. Electronics Letters, 18(21), 905-907.

[11] Bernstein, D. J., Lange, T., & Peters, C. (2017). Post-quantum cryptography. Nature, 549(7671), 188-194.