

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



BÁO CÁO CUỐI KỲ

CẤU TRÚC RỜI RẠC

Người hướng dẫn: **TS. NGUYỄN THỊ HUỲNH TRÂM**

Người thực hiện: **NGUYỄN VŨ GIA PHƯƠNG – 52200205**

MAI NGUYỄN PHƯƠNG TRANG – 52200051

LÊ CÔNG TUẤN – 52200033

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

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



BÁO CÁO CUỐI KỲ

CẤU TRÚC RỜI RẠC

Người hướng dẫn: **TS. NGUYỄN THỊ HUỲNH TRÂM**

Người thực hiện: **NGUYỄN VŨ GIA PHƯƠNG – 52200205**

MAI NGUYỄN PHƯƠNG TRANG – 52200051

LÊ CÔNG TUẤN – 52200033

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Lời đầu tiên, chúng em xin gửi lời cảm ơn chân thành đến Trường Đại Học Tôn Đức Thắng và khoa Công Nghệ Thông Tin đã đưa môn học cấu trúc rời rạc vào chương trình giảng dạy. Đặc biệt là giảng viên bộ môn, cô Nguyễn Thị Huỳnh Trâm, người đã truyền cảm hứng và những kiến thức bổ ích cho chúng em, người đã giải đáp những thắc mắc còn tồn đọng trong chúng em.

Môn cấu trúc rời rạc là môn học thú vị, trường và khoa đã sắp xếp vô cùng hợp lý giữa các tiết lý thuyết và thực hành, từ những kiến thức ở lớp lý thuyết, cô Nguyễn Thị Huỳnh Trâm đã giúp chúng em hiểu rõ trình tự thực thi của từng bài toán. Từ đó giúp nâng cao tư duy lập trình của chúng em, củng cố kiến thức toán học mà chúng em còn mơ hồ. Mặc dù chúng em đã cố gắng hết sức nhưng chắc chắn bài làm khó có thể tránh khỏi những thiếu sót và nhiều chỗ còn chưa chính xác, kính mong cô xem xét và góp ý để bài của chúng em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

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 TS. Nguyễn Thị Huỳnh Trâm. 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 22 tháng 5 năm 2024

Tác giả

Nguyễn Vũ Gia Phương

Mai Nguyễn Phương Trang

Lê Công Tuấn

TÓM TẮT

Mục tiêu nghiên cứu: giải quyết một số vấn đề liên quan về môn Cấu trúc rời rạc.

Nội dung chính:

- Thuật toán Euclid và bổ đề Bezout.
- Recurrence relation.
- Tập hợp.
- Mối quan hệ nhị phân (quan hệ hai ngôi).
- Kiểm tra chu trình trong thuật toán Kruskal.
- Chu trình Euler.
- Tô màu bản đồ.
- Nghịch đảo Modulo.
- Tìm hiểu về cách hoạt động của mã RSA, nghiên cứu các vấn đề xung quanh RSA.

MỤC LỤC

DANH MỤC HÌNH VẼ	x
DANH MỤC BẢNG BIỂU	xii
DANH MỤC CÁC CHỮ VIẾT TẮT.....	xiii
CHƯƠNG 1. EUCLID’S ALGORITHM AND BÉZOUT’S IDENTITY.....	1
1.1 Thuật toán Euclid và bổ đề Bézout:	1
1.2 Tính $\gcd(2024, 1033)$ và $\text{lcm}(2024, 1033)$	2
1.2.1 Tính $\gcd(2024, 1033)$	2
1.2.2 Tính $\text{lcm}(2024, 1033)$	3
1.3 Tính $2024x + 1033y = \gcd(2024, 1033)$	3
CHƯƠNG 2. RECURRENCE RELATION.....	4
2.1 Hệ thức truy hồi tuyến tính thuần nhất bậc 2 với hệ số hằng:	4
CHƯƠNG 3. SET	5
3.1 Tạo tập chứa ký tự từ tên, không phân biệt chữ hoa chữ thường và không dấu:..	5
3.2 Tìm giao, hợp, hiệu không đối xứng, hiệu đối xứng của Γ và Δ :	5
3.2.1 Giao của Γ và Δ (<i>intersect</i>):.....	5
3.2.2 Hội của Γ và Δ (<i>union</i>):	5
3.2.3 Hiệu của Γ và Δ và ngược lại (<i>non-symmetric difference</i>):.....	5
3.2.4 Hiệu đối xứng của Γ và Δ (<i>symmetric difference</i>):	5
CHƯƠNG 4. RELATIONS.....	6
4.1 Phản xạ (Reflexive):.....	6
4.2 Đối xứng (Symmetric):	6
4.3 Phản đối xứng (Anti-symmetric):	6

4.4 Bắc cầu (Transitive):	7
CHƯƠNG 5. KRUSKAL'S ALGORITHM	8
5.1 Giới thiệu thuật toán Union-Find (Disjoint Set):	8
5.1.1 Sự khác biệt giữa thuật Kruskal với thuật toán Kruskal áp dụng Union-Find:	
8	
5.1.2 Ưu nhược điểm của thuật toán Union-Find:	9
5.1.3 Sơ lược các bước thực hiện:	9
5.2 Ví dụ:	10
5.2.1 Đồ thị ví dụ:	10
5.2.2 Lập danh sách cạnh (Edge List):	10
5.2.3 Lập bảng Union-Find:	11
5.2.4 Kết luận:	14
CHƯƠNG 6. EULERIAN CIRCUIT	15
6.1 Chứng minh đồ thị có Eulerian circuit hoặc Eulerian path:	15
6.2 Thuật toán Hierholzer:	17
6.2.1 Giới thiệu sơ lược thuật toán Hierholzer:	17
6.2.2 Thực hiện thuật toán Hierholzer:	17
6.2.3 Hiệu quả trong code:	18
6.2.4 Ưu nhược điểm của thuật toán Hierholzer:	18
6.2.5 Ví dụ:	19
6.3 Áp dụng thuật toán Hierholzer tìm Eulerian circuit của đồ thị khi có initial circuit R1:	22
CHƯƠNG 7. MAP COLORING	35
7.1 Mô hình hoá biểu đồ bằng đồ thị:	35

7.2 Tô màu đồ thị:	39
CHƯƠNG 8. FINDING AN INVERSE MODULO	45
8.1 Cơ sở lý thuyết	45
8.1.1 <i>Phép tính đồng dư modulo</i>	45
8.1.2 <i>Số nguyên tố cùng nhau và phân tử nghịch đảo</i>	45
8.1.3 <i>Áp dụng thuật toán Euclid mở rộng vào bài toán tìm phân tử nghịch đảo</i> ...	45
8.1.4 <i>Ví dụ áp dụng</i>	46
8.2 Hiện thực chương trình	47
8.3 Kiểm thử & xác thực	47
8.3.1 <i>Kiểm tra chương trình bằng dữ liệu mẫu</i>	48
8.3.2 <i>Kết quả chạy</i>	48
8.3.3 <i>Hậu kiểm kết quả</i>	48
CHƯƠNG 9. RSA CRYPTOSYSTEM	50
9.1 Hệ thống mã hóa RSA	50
9.1.1 <i>Các khái niệm toán học cơ bản</i>	50
9.1.2 <i>Cơ bản về mã RSA</i>	50
9.1.3 <i>Vận hành RSA</i>	51
9.2 Hiện thực	52
9.3 Kiểm thử và xác minh	53
9.3.1 <i>Kiểm thử</i>	53
9.3.2 <i>Xác minh</i>	54
9.4 Phân tích hiệu quả và bảo mật	54
9.4.1 <i>Hiệu quả</i>	54

9.4.2 Bảo mật	55
9.5 Thảo luận về các mối đe dọa bảo mật và hạn chế của hệ mật RSA.....	55
9.5.1 Các mối đe dọa bảo mật	55
9.5.2 Hạn chế	56
9.6 Khuyến nghị cải thiện	56
TÀI LIỆU THAM KHẢO	57

DANH MỤC HÌNH VẼ

Hình 5.1 – Đồ thị ví dụ cho thuật toán Union-Find trong Kruskal.....	10
Hình 6.1 – Đồ thị gốc.....	15
Hình 6.2 – Đồ thị có hướng.....	16
Hình 6.3 – Đồ thị ví dụ cho thuật toán Hierholzer.....	19
Hình 6.4 – Đồ thị ví dụ cho thuật toán Hierholzer 1.1.....	20
Hình 6.5 – Đồ thị ví dụ cho thuật toán Hierholzer 1.2.....	20
Hình 6.6 – Đồ thị ví dụ cho thuật toán Hierholzer 1.3.....	21
Hình 6.7 – Áp dụng thuật toán vào đồ thị gốc.....	22
Hình 6.8 Áp dụng thuật toán vào đồ thị gốc 1.1.....	22
Hình 6.9 – Áp dụng thuật toán vào đồ thị gốc 1.2.....	23
Hình 6.10 – Áp dụng thuật toán vào đồ thị gốc 1.3.....	23
Hình 6.11 – Áp dụng thuật toán vào đồ thị gốc 1.4.....	24
Hình 6.12 – Áp dụng thuật toán vào đồ thị gốc 1.5.....	24
Hình 6.13 – Áp dụng thuật toán vào đồ thị gốc 1.6.....	25
Hình 6.14 – Áp dụng thuật toán vào đồ thị gốc 1.7.....	25
Hình 6.15 – Áp dụng thuật toán vào đồ thị gốc 1.8.....	26
Hình 6.16 – Áp dụng thuật toán vào đồ thị gốc 1.9.....	26
Hình 6.17 – Áp dụng thuật toán vào đồ thị gốc 1.10.....	27
Hình 6.18 – Áp dụng thuật toán vào đồ thị gốc 1.11.....	27
Hình 6.19 – Áp dụng thuật toán vào đồ thị gốc 1.12.....	28
Hình 6.20 – Áp dụng thuật toán vào đồ thị gốc 1.13.....	28
Hình 6.21 – Áp dụng thuật toán vào đồ thị gốc 1.14.....	29

Hình 6.22 – Áp dụng thuật toán vào đồ thị gốc 1.15	29
Hình 6.23 – Áp dụng thuật toán vào đồ thị gốc 1.16	30
Hình 6.24 – Áp dụng thuật toán vào đồ thị gốc 1.17	30
Hình 6.25 – Áp dụng thuật toán vào đồ thị gốc 1.18	31
Hình 6.26 – Áp dụng thuật toán vào đồ thị gốc 1.19	31
Hình 6.27 – Áp dụng thuật toán vào đồ thị gốc 1.20	32
Hình 6.28 – Áp dụng thuật toán vào đồ thị gốc 1.21	32
Hình 6.29 – Áp dụng thuật toán vào đồ thị gốc 1.22	33
Hình 6.30 – Áp dụng thuật toán vào đồ thị gốc 1.23	33
Hình 6.31 – Áp dụng thuật toán vào đồ thị gốc 1.24	34
Hình 8.1 – Code hiện thực tìm modulo nghịch đảo n	47
Hình 8.2 – Code kiểm tra chương trình bằng dữ liệu mẫu	48
Hình 8.3 – Kết quả chạy của bài toán $13^{-1} \bmod 97$	48
Hình 8.4 – Code hậu kiểm kết quả thực thi.....	48
Hình 8.5 – Kết quả chạy hậu kiểm của bài toán $13^{-1} \bmod 13$	49
Hình 8.6 – Kết quả chạy hậu kiểm của bài toán $8^{-1} \bmod 12$	49
Hình 8.7 – Kết quả chạy hậu kiểm của bài toán $13^{-1} \bmod 97$	49
Hình 9.1 – Code hiện thực quy trình tạo khóa công khai và khóa bí mật.....	52
Hình 9.2 – Kiểm thử & mã hóa và giải mã thông điệp	53
Hình 9.3 – Kết quả quy trình mã hóa & giải mã.....	54
Hình 9.4 – Code xác minh lại tính đúng đắn của thuật toán.....	54
Hình 9.5 – Kết quả xác minh quy trình giải mã.....	54

DANH MỤC BẢNG BIỂU

Bảng 5.1 – Danh sách cạnh ban đầu	10
Bảng 5.2 – Danh sách cạnh đã sắp xếp	11
Bảng 5.3 – Bảng Union-Find 1.1	11
Bảng 5.4 – Bảng Union-Find 1.2	12
Bảng 5.5 – Bảng Union-Find 1.2	12
Bảng 5.6 – Bảng Union-Find 1.3	13
Bảng 5.7 – Bảng Union-Find 1.3	13
Bảng 7.1 – Danh sách các tỉnh có trong biểu đồ.....	36
Bảng 7.2 – Danh sách bảng màu.....	40
Bảng 7.3 – Các bước tô màu đồ thị.....	43

DANH MỤC CÁC CHỮ VIẾT TẮT

ƯCLN	Ước chung lớn nhất
GCD	Greatest common divisor
BCNN	Bội chung nhỏ nhất

CHƯƠNG 1. EUCLID'S ALGORITHM AND BÉZOUT'S IDENTITY

MSSV: 52200033 $\Rightarrow m = 33$, thay $m = 33$ vào:

- $\gcd(2024, 1000 + m)$,
- $\text{lcm}(2024, 1000 + m)$,
- $2024x + (1000 + m)y = \gcd(2024, 1000 + m)$

Ta được:

- $\gcd(2024, 1033)$,
- $\text{lcm}(2024, 1033)$,
- $2024x + 1033y = \gcd(2024, 1033)$

1.1 Thuật toán Euclid và bổ đề Bézout:

Theo thuật toán Euclid, để tìm ước chung lớn nhất $\gcd(a, b)$ dựa vào 2 ý sau:

$$(i) \gcd(a, 0) = a$$

$$(ii) \gcd(a, b) = \gcd(b, r)$$

(trong đó r là phần dư của phép chia a/b)

- Với (i), $\gcd(a, 0) = a$: Khi một trong hai số trong cặp $(a, 0)$ là 0, thì ước chung lớn nhất của cặp số đó chính là số còn lại trong cặp đó.
- Theo định lý 4.1.1 Quotient-remainder [1], Cho a và b là hai số nguyên bất kỳ, với $b \neq 0$. Khi đó, tồn tại duy nhất hai số nguyên q và r thỏa mãn:

$$a = bq + r; \quad 0 \leq r < b \quad (1.1)$$

- Trong đó:
 - q : được gọi là thương khi chia a cho b .
 - r : được gọi là số dư khi chia a cho b .
- Với (ii), vì $a = bq + r$ bất kỳ ước chung c nào của a, b cũng phải chia hết cho r theo Định lý 4.1.1 đã giải thích ở trên. Ngoài ra, bất kỳ ước chung nào của b, r cũng phải chia hết cho a vì cùng một lý do. Vì vậy (a, b) và (b, r) có cùng một tập hợp các ước chung, và do đó \gcd của chúng phải bằng nhau.

- Thuật toán Euclid: lặp lại quá trình tìm kiếm số dư r và thay $a = b$, $b = r$ của phép chia a, b cho đến khi $r \leq 0$ thì tìm được ước chung lớn nhất của a và b .
- Từ đó, ta có công thức tính BCNN của a, b :

$$lcm(a, b) = \frac{ab}{gcd(a, b)} \quad (1.2)$$

- Bổ đề Bézout (Thuyết 4.5.3 [1]) khẳng định rằng: Với hai số nguyên không âm a, b có ƯCLN là d , tồn tại hai số nguyên x và y sao cho:

$$ax + by = d \quad (1.3)$$

- Áp dụng thuật toán Euclid, ta tìm được một cặp (x, y) mẫu, từ đó, ta tìm được công thức tổng quát các cặp hệ số còn lại có dạng:

$$\left(x + \frac{kb}{d}, y - \frac{ka}{d}\right) \quad (1.4)$$

- Trong đó:
 - (x, y) : cặp nghiệm tìm được từ thuật toán Euclid
 - a, b : là 2 số nguyên ban đầu mà ta cần tìm ƯCLN của chúng
 - d là ƯCLN của 2 số a và b
 - $k \in \mathbb{Z}^*$

1.2 Tính $gcd(2024, 1033)$ và $lcm(2024, 1033)$

1.2.1 Tính $gcd(2024, 1033)$

Theo công thức (1.1), $gcd(2024, 1033)$ được tính như sau:

- $2024 = 1033 \times 1 + 991 \quad \leftarrow gcd(1033, 991)$
- $1033 = 991 \times 1 + 42 \quad \leftarrow gcd(991, 42)$
- $991 = 42 \times 23 + 25 \quad \leftarrow gcd(42, 25)$
- $42 = 25 \times 1 + 17 \quad \leftarrow gcd(25, 17)$
- $25 = 17 \times 1 + 8 \quad \leftarrow gcd(17, 8)$
- $17 = 8 \times 2 + 1 \quad \leftarrow gcd(8, 1)$
- $8 = 1 \times 8 + 0 \quad \leftarrow gcd(1, 0)$

Do đó, $gcd(2024, 1033) = 1$

1.2.2 Tính $\text{lcm}(2024, 1033)$

Theo công thức (1.2), $\text{lcm}(2024, 1033)$ được tính như sau:

$$\text{lcm}(a, b) = \frac{2024 \times 1033}{1} = 2090792$$

Do đó, $\text{lcm}(2024, 1033) = 2090792$

1.3 Tính $2024x + 1033y = \text{gcd}(2024, 1033)$

- Để tính $2024x + 1033y = \text{gcd}(2024, 1033) = 1$, ta cần đệ quy lại các bước tính UCLN ở bước trên, từ đó, suy ra công thức tổng quát của phương trình:

$$2024x + 1033y = 1$$

$$\begin{aligned} 1 &= 17 - 8 \times 2 = 17 + 8 \times (-2) \\ &= 17 + (25 - 17 \times 1) \times (-2) = 25 \times (-2) + 17 \times 3 \\ &= 25 \times (-2) + (42 - 25) \times 3 = 42 \times 3 + 25 \times (-5) \\ &= 42 \times 3 + (991 - 42 \times 23) \times (-5) = 991 \times (-5) + 42 \times 118 \\ &= 991 \times (-5) + (1033 - 991) \times 118 = 1033 \times 118 + 991 \times (-123) \\ &= 1033 \times 118 + (2024 - 1033) \times (-123) = 2024 \times (-123) + 1033 \times (241) \end{aligned}$$

Vậy $1 = 2024 \times (-123) + 1033 \times (241)$

- Theo công thức (1.3), ta có nghiệm của phương trình $2024x + 1033y = 1$ là $x = -123, y = 241$.

- Thay $a = 2024, b = 1033, d = 1, x = -123, y = 241$ vào công thức (1.4), ta có công thức nghiệm tổng quát của phương trình $2024x + 1033y = 1$:

$$\left(-123 + \frac{1033k}{1}, 241 - \frac{2024k}{1} \right), k \in \mathbb{Z}^* \quad (1.5)$$

- Lần lượt thay $k = 1, k = 2, k = 3, k = 4, k = 5$ vào công thức nghiệm (1.5), ta được các cặp nghiệm: $(910, -1783), (1943, -3807), (2976, -5831), (4009, -7855), (5042, -9879)$

Vậy 5 cặp nghiệm (x, y) là $(910, -1783), (1943, -3807), (2976, -5831), (4009, -7855), (5042, -9879)$

CHƯƠNG 2. RECURRENCE RELATION

2.1 Hệ thức truy hồi tuyến tính thuần nhất bậc 2 với hệ số hằng:

Tìm số hạng tổng quát của $a_n = 8.a_{n-1} - 15.a_{n-2}$ với: $a_0 = 5, a_1 = m$.
(Với m là 2 số cuối của MSSV nhỏ nhất trong nhóm là 52200033 thì $a_1 = m = 33$)

Giải:

Chuỗi a_n thỏa mãn giả thuyết của hệ thức truy hồi tuyến tính thuần nhất bậc 2 với hệ số hằng: **A = 8** và **B = -15**. Khi đó, phương trình đặc trưng của chuỗi a_n là:

$$t^2 - 8t + 15 = 0 \quad (1)$$

- Trong đó:

- $a = 1$
- $b = -8$
- $c = 15$

- Theo (1), ta có: $\Delta = b^2 - 4ac = (-8)^2 - 4.1.15 = 4 > 0$. Với $\Delta > 0$, phương trình có 2 nghiệm phân biệt thỏa mãn phương trình đặc trưng (1) là:

$$t = \frac{-b \pm \sqrt{\Delta}}{2a} \begin{cases} t_1 = \frac{-b + \sqrt{\Delta}}{2a} = \frac{8 + \sqrt{4}}{2} = 5 \\ t_2 = \frac{-b - \sqrt{\Delta}}{2a} = \frac{8 - \sqrt{4}}{2} = 3 \end{cases} \quad (2)$$

- Khi đó 2 nghiệm này thỏa mãn phương trình nghiệm tổng quát là:

$$a_n = C(t_1)^n + D(t_2)^n = C(5)^n + D(3)^n \quad (3)$$

- Với $a_0 = 5$ và $a_1 = 33$:

$$\bullet \quad a_0 = C(5)^0 + D(3)^0 = 1.C + 1.D = C + D = 5 \quad (4)$$

$$\bullet \quad a_1 = C(5)^1 + D(3)^1 = 5C + 3D = 33 \quad (5)$$

Nghiệm của hệ phương trình (4) và (5): $C = 9$ và $D = -4$

- Thay C, D vào phương trình (3): $a_n = C(5)^n + D(3)^n$

$$a_n = 9.(5)^n + (-4).(3)^n$$

Vậy với dãy số a_n thì số hạng tổng quát là: $a_n = 9.(5)^n + (-4).(3)^n$

CHƯƠNG 3. SET

3.1 Tạo tập chứa ký tự từ tên, không phân biệt chữ hoa chữ thường và không dấu:

- Tập Δ ứng với tên “Tôn Đức Thắng”: $\Delta = \{A, C, D, G, H, N, O, T, U\}$
- Tập Γ ứng với tên “Lê Công Tuấn”: $\Gamma = \{L, E, C, O, N, G, T, U, A\}$
 $= \{A, C, E, G, L, N, O, T, U\}$

3.2 Tìm giao, hợp, hiệu không đối xứng, hiệu đối xứng của Γ và Δ :

3.2.1 Giao của Γ và Δ (intersect):

Phép giao của tập hợp Γ và tập hợp Δ là tập hợp tạo bởi tất cả các phần tử vừa thuộc Γ vừa thuộc Δ : $\Gamma \cap \Delta = \{A, C, G, N, O, T, U\}$

3.2.2 Hội của Γ và Δ (union):

Hội của 2 tập hợp Γ và Δ là tập hợp tất cả các phần tử thuộc Γ hoặc thuộc Δ hoặc cả hai: $\Gamma \cup \Delta = \{A, C, D, E, G, H, L, N, O, T, U\}$

3.2.3 Hiệu của Γ và Δ và ngược lại (non-symmetric difference):

- Gọi U là tập hợp cha của tập hợp Γ và tập hợp Δ . Hiệu của tập hợp Γ và tập hợp Δ thuộc tập vũ trụ U là tập hợp tạo bởi tất cả các phần tử vừa thuộc Γ mà không thuộc Δ : $\Gamma \setminus \Delta = \{E, L\}$
- Ngược lại, hiệu của tập hợp Δ và tập hợp Γ thuộc tập vũ trụ U là tập hợp tạo bởi tất cả các phần tử vừa thuộc Δ mà không thuộc Γ : $\Delta \setminus \Gamma = \{D, H\}$

3.2.4 Hiệu đối xứng của Γ và Δ (symmetric difference):

Phép hiệu đối xứng giữa hai tập hợp Γ và Δ là phép toán mà kết quả là tập hợp của các phần tử mà không thuộc cả Γ và cũng không thuộc Δ . Nói cách khác, hiệu đối xứng giữa Γ và Δ là tập hợp của các phần tử chỉ thuộc một trong hai tập hợp Γ hoặc Δ , nhưng không thuộc cả hai: $\Gamma \oplus \Delta = (\Gamma \setminus \Delta) \cup (\Delta \setminus \Gamma) = \{D, E, H, L\}$

CHƯƠNG 4. RELATIONS

Cho R là một quan hệ nhị phân được định nghĩa trên tập hợp các số nguyên như sau: $\forall a, b \in N (aRb \leftrightarrow 33|(a \cdot b))$

4.1 Phản xạ (Reflexive):

- Để chứng minh rằng quan hệ R có tính phản xạ, ta cần chứng minh:

$$\forall a \in N, a R a$$

- Theo định nghĩa của R , điều này nghĩa là:

$$\forall a \in N, 33|(a \cdot a)$$

- Ta có: $33|(a \cdot a) = 33|a^2$

$$\leftrightarrow a = \sqrt{33} \text{ hoặc } a = -\sqrt{33}$$

- Nhưng $a \in N$ nên loại cả hai kết quả.

→ Do đó, quan hệ R không có tính phản xạ.

4.2 Đối xứng (Symmetric):

- Để chứng minh rằng quan hệ R có tính đối xứng, ta cần chứng minh:

$$\forall a, b \in N, aRb \rightarrow bRa$$

- Theo định nghĩa của R , điều này nghĩa là:

$$\forall a, b \in N, 33|(a \cdot b) \rightarrow 33|(b \cdot a)$$

- Ta có: $a \cdot b = b \cdot a$ (do tính giao hoán của phép nhân)

→ $a \cdot b$ và $b \cdot a$ đều chia hết cho 33

→ Do đó, quan hệ R có tính đối xứng.

4.3 Phản đối xứng (Anti-symmetric):

- Để chứng minh rằng quan hệ R có tính phản đối xứng, ta cần chứng minh:

$$\forall a, b \in N, (aRb \wedge bRa) \rightarrow a = b$$

- Theo định nghĩa của R , điều này nghĩa là:

$$\forall a, b \in N, (33|(a \cdot b) \wedge 33|(b \cdot a)) \rightarrow a = b$$

- Ta có: $a.b = b.a$ (do tính giao hoán của phép nhân)
 $\rightarrow a.b$ và $b.a$ đều chia hết cho 33
- Nhưng không chắc chắn $a = b$. Ví dụ, 33 chia hết cho 11×3 và 3×11 nhưng 3 không bằng 11
 \rightarrow Do đó, quan hệ R không có tính phản đối xứng.

4.4 Bắt cầu (Transitive):

- Để chứng minh rằng quan hệ R có tính bắc cầu, ta cần chứng minh:

$$\forall a, b, c \in N, (aRb \wedge bRc) \rightarrow aRc$$
- Theo định nghĩa của R, điều này nghĩa là:

$$\forall a, b, c \in N, (33|(a.b) \wedge 33|(b.c)) \rightarrow 33|(a.c)$$
- Giả sử $a = 3, b = 11, c = 9$. Ta có:
 - $a.b = 33$ chia hết cho 33
 - $b.c = 99$ chia hết cho 33
 - $a.c = 27$ **KHÔNG** chia hết cho 33 \rightarrow Do đó, quan hệ R không có tính bắc cầu.

CHƯƠNG 5. KRUSKAL'S ALGORITHM

5.1 Giới thiệu thuật toán Union-Find (Disjoint Set):

- Thuật toán Kruskal là một thuật toán tham lam được sử dụng để tìm cây khung nhỏ nhất (MST) trong một đồ thị vô hướng. Nó hoạt động bằng cách duy trì một tập hợp các cạnh được sắp xếp theo trọng số tăng dần và lần lượt thêm các cạnh vào cây khung cho đến khi tất cả các nút được kết nối.
- Thuật toán Union-Find là một thuật toán để thực hiện các thao tác trên cấu trúc dữ liệu rời rạc. Nó cung cấp hai thao tác chính:
 - Find(x, y): Xác định xem hai nút x và y có thuộc cùng một tập hợp hay không.
 - Union(x, y): Hợp nhất hai tập hợp chứa nút x và y thành một tập hợp duy nhất.

5.1.1 Sự khác biệt giữa thuật Kruskal với thuật toán Kruskal áp dụng Union-Find:

- Thuật toán Kruskal:
 - Sử dụng một danh sách hoặc mảng đơn giản để duy trì các tập hợp nút rời rạc.
 - Cập nhật danh sách hoặc mảng này khi thêm các cạnh vào MST, phản ánh việc hợp nhất các thành phần liên thông.
- Thuật toán Kruskal áp dụng Union-Find:
 - Sử dụng cấu trúc dữ liệu Union-Find phức tạp hơn để quản lý các tập hợp nút rời rạc.
 - Cấu trúc dữ liệu này cung cấp các hoạt động hiệu quả cho cả việc tìm xem hai nút có thuộc cùng một tập hợp hay không (tìm kiếm) và hợp nhất hai tập hợp (liên kết).

5.1.2 Ưu nhược điểm của thuật toán Union-Find:

- Ưu điểm:
 - Hiệu quả: Thuật toán Union-Find có độ phức tạp thời gian trung bình là $O(\log n)$, với n là số lượng nút trong đồ thị.
 - Dễ dàng triển khai: Thuật toán Union-Find có thể được triển khai dễ dàng bằng cách sử dụng các cấu trúc dữ liệu như mảng hoặc danh sách liên kết.
 - Giảm thiểu sử dụng bộ nhớ: Thuật toán Union-Find hiệu quả hơn về bộ nhớ so với thuật toán Kruskal thông thường, đặc biệt là cho các đồ thị lớn.
- Nhược điểm:
 - Độ phức tạp: Thuật toán Union-Find có độ phức tạp thời gian trung bình là $O(\log n)$, cao hơn so với thuật toán Kruskal thông thường là $O(n)$. Tuy nhiên, sự khác biệt này thường không đáng kể ở các đồ thị lớn.
 - Việc triển khai thuật toán Union-Find có thể phức tạp hơn so với thuật toán Kruskal thông thường.

5.1.3 Sơ lược các bước thực hiện:

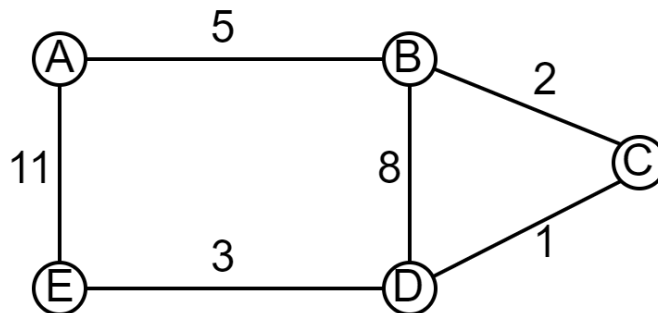
- Khởi tạo:
 - Sắp xếp tất cả các cạnh của đồ thị theo thứ tự tăng dần của trọng số.
 - Khởi tạo một bảng cấu trúc Union-Find để theo dõi các tập hợp của các đỉnh.
- Chọn cạnh:
 - Lặp qua các cạnh theo thứ tự đã sắp xếp.
 - Với mỗi cạnh, sử dụng Union-Find để kiểm tra xem hai đỉnh của cạnh đó có nằm trong cùng một thành phần liên thông (một nhóm) hay không.

- Nếu hai đỉnh không nằm trong cùng một thành phần liên thông, thêm cạnh đó vào cây khung và hợp nhất hai tập hợp lại (sử dụng hàm union của Union-Find khi code).
- Kết thúc:
- Lặp lại bước 2 cho đến khi cây khung có đủ $n-1$ cạnh (với n là số đỉnh trong đồ thị).

5.2 Ví dụ:

5.2.1 Đồ thị ví dụ:

Giả sử, ta có đồ thị sau:



Hình 5.1 Đồ thị ví dụ cho thuật toán Union-Find trong Kruskal

5.2.2 Lập danh sách cạnh (Edge List):

- Lập danh sách cạnh gồm cạnh và trọng số của chúng:

Cạnh	Trọng số
A - B	5
B - C	2
A - E	11
E - D	3
B - D	8
D - C	1

Bảng 5.1 Danh sách cạnh ban đầu

- Sắp xếp cạnh theo thứ tự giảm dần của trọng số:

Cạnh	Trọng số
D - C	1
B - C	2
E - D	3
A - B	5
B - D	8
A - E	11

Bảng 5.2 Danh sách cạnh đã sắp xếp

5.2.3 Lập bảng Union-Find:

- Khởi tạo bảng Union-Find, bảng bao gồm đỉnh, đỉnh đại diện (đỉnh cha) và hạng của đỉnh:

Đỉnh	Đỉnh đại diện	Hạng
A	A	0
B	B	0
C	C	0
D	D	0
E	E	0

Bảng 5.3 Bảng Union-Find 1.1

- Bảng Union-Find ban đầu, đỉnh đại diện của mỗi đỉnh sẽ là chính nó và thứ hạng sẽ ở mức ban đầu là 0. Sau đó, ta sẽ lặp qua các cạnh đã sắp xếp theo thứ tự trong danh sách cạnh.

5.2.3.1 Xét cạnh (D - C, 1):

Đỉnh	Đỉnh đại diện	Hạng
A	A	0
B	B	0
C	C	1
D	C	0
E	E	0

Bảng 5.4 Bảng Union-Find 1.2

- Xét cạnh D – C với trọng số 1, đỉnh C và đỉnh D đang cùng hạng 0 nên ta có thể chọn một trong hai để làm đỉnh đại diện khi đó hạng của đỉnh được chọn sẽ là hạng 1.
- Trường hợp này, nhóm em chọn đỉnh C nên hạng đỉnh C là hạng 1 và đỉnh đại diện là chính nó. Đỉnh D có hạng 0 và đỉnh đại diện là đỉnh C.

5.2.3.2 Xét cạnh (B - C, 2):

Đỉnh	Đỉnh đại diện	Hạng
A	A	0
B	C	0
C	C	1
D	C	0
E	E	0

Bảng 5.5 Bảng Union-Find 1.2

Xét cạnh B – C với trọng số 2, đỉnh C là đỉnh đại diện hạng 1 có hạng lớn hơn đỉnh B nên đỉnh đại diện của B là đỉnh C và hạng của đỉnh B là 0.

5.2.3.3 Xét cạnh (E - D, 3):

Đỉnh	Đỉnh đại diện	Hạng
A	A	0
B	C	0
C	C	1
D	C	0
E	C	0

Bảng 5.6 Bảng Union-Find 1.3

Xét cạnh E – D với trọng số 3, đỉnh D có đỉnh ảnh đại diện là đỉnh C hạng 1 lớn hơn hạng đỉnh E nên đỉnh E sẽ có đỉnh đại diện là đỉnh C và hạng 0.

5.2.3.4 Xét cạnh (A - B, 5):

Đỉnh	Đỉnh đại diện	Hạng
A	C	0
B	C	0
C	C	1
D	C	0
E	C	0

Bảng 5.7 Bảng Union-Find 1.3

Xét cạnh A – B với trọng số 5, đỉnh B có đỉnh ảnh đại diện là đỉnh C hạng 1 lớn hơn hạng đỉnh A nên đỉnh A sẽ có đỉnh đại diện là đỉnh C và hạng 0.

Đến đây, cây khung đã có đủ $n - 1$ cạnh (4 cạnh 5 đỉnh).

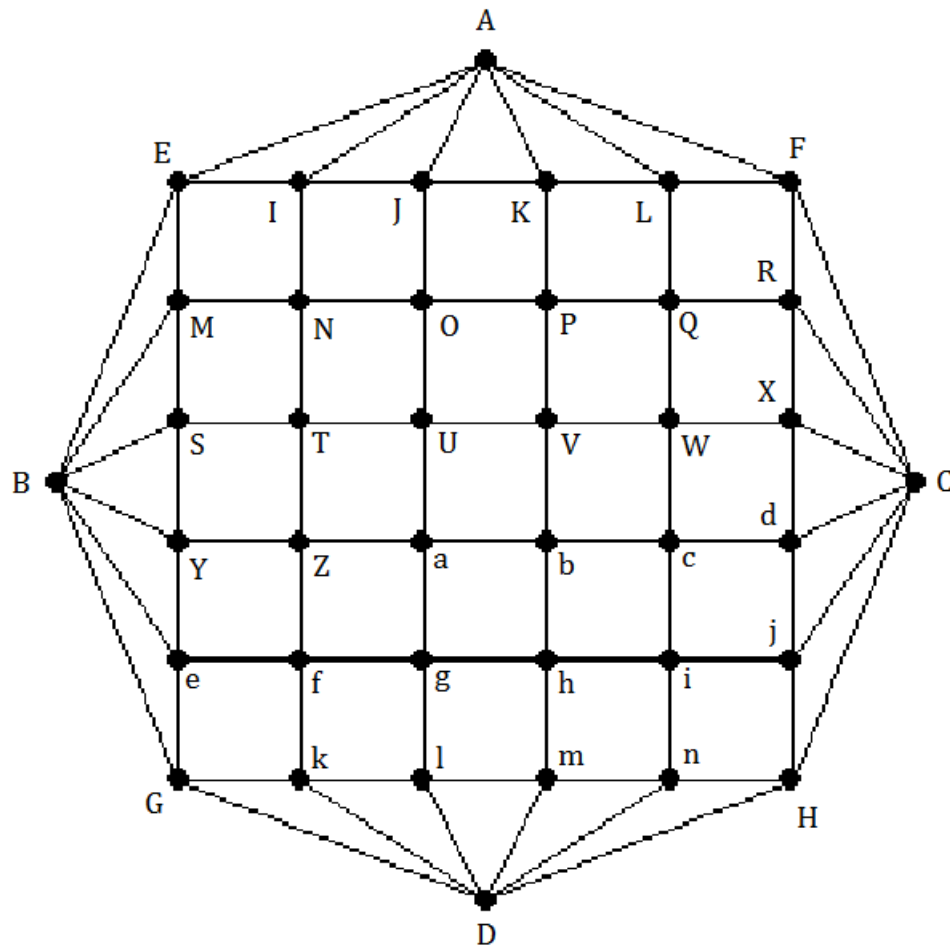
5.2.4 Kết luận:

Cây khung nhỏ nhất chứa các cạnh:

- D – C (trọng số 1)
- B – C (trọng số 2)
- E – D (trọng số 3)
- A – B (trọng số 5)

Tổng trọng số của cây khung nhỏ nhất là $1 + 2 + 3 + 5 = 9$.

CHƯƠNG 6. EULERIAN CIRCUIT



Hình 6.1 Đồ thị gốc

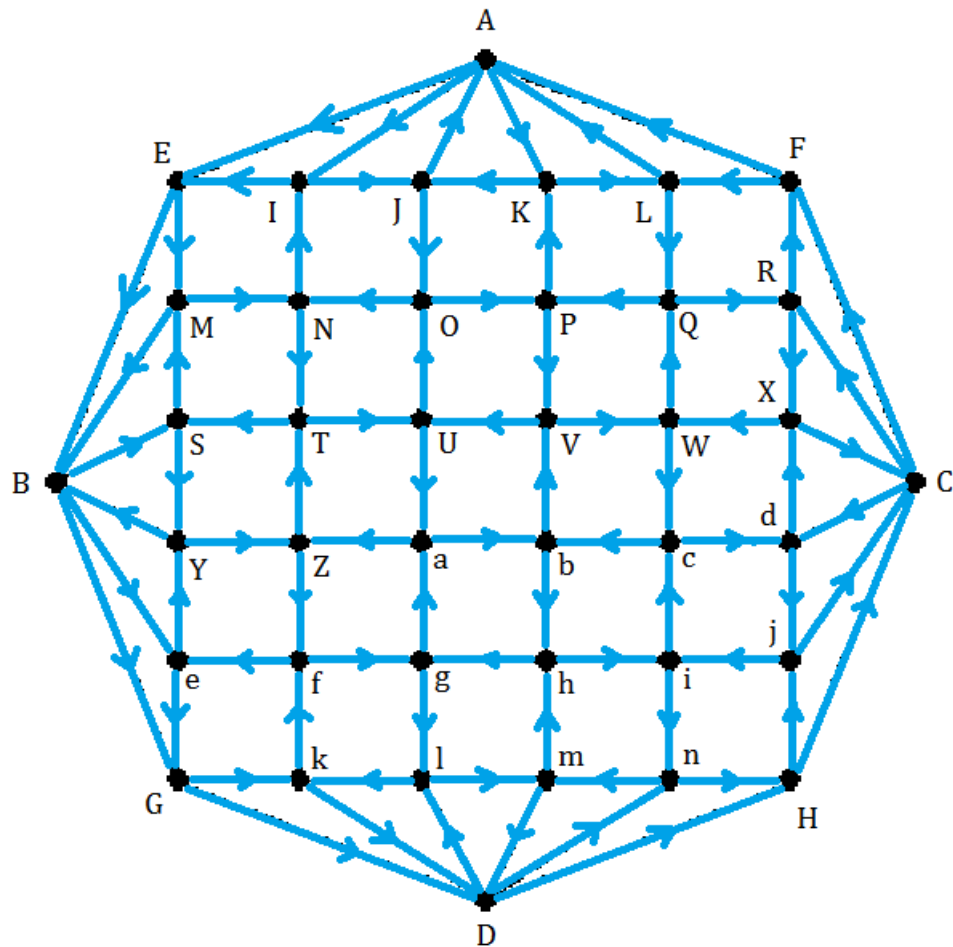
6.1 Chứng minh đồ thị có Eulerian circuit hoặc Eulerian path:

- Từ hình 6.1 ta có:

- Các đỉnh A, B, C, D đều có độ là 6
- Các đỉnh còn lại E, I, J, K, L, F, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z, a, b, c, d, e, f, g, h, i, j, G, k, l, m, n, H là các đỉnh có độ là 4.

→ Độ của các đỉnh đều là độ chẵn.

→ Loại trường hợp đồ thị có Eulerian path (Eulerian path yêu cầu có đỉnh có độ lẻ và nhiều nhất hai đỉnh có độ lẻ).



Hình 6.2 Đồ thị có hướng

- Đồ thị kết nối được tất cả các cạnh với các đỉnh với đường đi như sau: A E B G D H C F A I E M B S Y B e G k D l m D n H j C d X C R F L A K J O P K L Q R X W c d j i n m h i c b V W Q P V U a b h g l k f e Y Z f g a Z T S M N T U O N I J A

→ Tất cả các cạnh đều được duyệt qua một lần, duyệt qua tất cả các đỉnh và đỉnh đầu và đỉnh kết thúc cùng tại đỉnh A.

Kết luận: Đồ thị đang xét có Eulerian circuit.

6.2 Thuật toán Hierholzer:

6.2.1 Giới thiệu sơ lược thuật toán Hierholzer:

Thuật toán Hierholzer là một thuật toán được tạo ra bởi Hierholzer vào năm 1873. Đây là một thuật toán hiệu quả hơn thuật toán Fleury để tìm Euler cycle, cụ thể là tìm Eulerian path hoặc Eulerian circuit trong đồ thị vô hướng và có hướng.

6.2.2 Thực hiện thuật toán Hierholzer:

Trước hết, đồ thị áp dụng thuật toán phải là đồ thị Eulerian và là đồ thị liên thông (connected). Riêng đối với đồ thị có hướng, tất cả các đỉnh phải có mức độ đỉnh chẵn và tổng số lượng cạnh vào (incoming edges) phải bằng tổng số lượng cạnh đi ra (outgoing edges).

6.2.2.1 Ý tưởng:

Sử dụng các sub-cycle trong đồ thị. Tạo ra sub-cycle và thêm các sub-cycle vào các đỉnh đã trong path đã tồn tại, quá trình này được lặp đến khi duyệt qua hết tất cả các cạnh của đồ thị.

6.2.2.2 Thực hiện thuật toán:

- **Bước 1:** Chọn một đỉnh bắt đầu bất kì (ký hiệu là v).
- **Bước 2:** Bắt đầu từ đỉnh v , đi theo từng cạnh cho đến khi quay lại chính đỉnh v . Lưu ý, trên đường duyệt nên kết thúc tại đỉnh v bởi vì mọi đỉnh đều có bậc chẵn (số cạnh nối với một đỉnh là số chẵn). Khi đó, sẽ đảm bảo được khi duyệt một đỉnh khác, luôn có một cạnh chưa duyệt nối với đỉnh đó.
- **Bước 3:** Nối các cạnh lại tạo thành một circuit nhưng circuit này có thể chưa bao gồm tất cả các cạnh và các đỉnh.
- **Bước 4:** Tìm các đường đi nhánh của circuit đã tìm ở bước 3. Kiểm tra xem có đỉnh (ví dụ đỉnh w) nào thuộc đường đi hiện tại nhưng vẫn còn các cạnh nối với các đỉnh khác chưa được duyệt qua hay không. Nếu có, tạo sub-cycle là một đường đi nhánh mới từ đỉnh w , đi theo các cạnh chưa duyệt qua cho

đến khi trở lại đỉnh w . Sau đó, nối sub-cycle với circuit chính đã tìm ở bước 3.

- Lặp lại bước 4 cho đến khi đã duyệt qua tất cả các cạnh của đồ thị. Khi ban đầu đồ thị là đồ thị liên thông (connected) thì việc lặp này đảm bảo đi qua tất cả các cạnh của đồ thị đó.

6.2.3 Hiệu quả trong code:

- Khi ta sử dụng danh sách liên kết kép (doubly linked list) để quản lý các cạnh chưa được duyệt tại mỗi đỉnh, danh sách các đỉnh trên đường đi hiện tại có các cạnh chưa được duyệt.
- Các thao tác của thuật toán như tìm các cạnh chưa duyệt của mỗi đỉnh, tìm đỉnh bắt đầu mới cho sub-cycle đường đi nhánh và kết nối hai đường đi chung một đỉnh có thể thực hiện trong một hằng số thời gian nên thuật toán có độ phức tạp tuyến tính $O(|E|)$ với $|E|$ là số lượng cạnh trong đồ thị.
- Cách khắc phục lỗi bị dừng tại một đỉnh không phải đỉnh ban đầu: Thuật toán Hierholzer cũng có thể được triển khai bằng deque (double-ended queue). Trong quá trình, thuật toán bị dừng tại đỉnh khác (nghĩa là deque hiện tại đang là một vòng kín), khi đó xoay deque lại bằng xóa phần tử cuối và thêm phần tử đã xóa vào đầu cho đến khi thoát khỏi hết bị dừng tại đỉnh khác. Sau đó, tiếp tục thuật toán cho đến khi duyệt hết tất cả các cạnh. Cách làm này có độ phức tạp tuyến tính vì số lần xoay deque không bao giờ lớn hơn $|E|$ với $|E|$ là số lượng cạnh trong đồ thị.

6.2.4 Ưu nhược điểm của thuật toán Hierholzer:

- Ưu điểm:
 - Thuật toán Hierholzer có cách thức hoạt động đơn giản, dễ dàng để hiểu và triển khai.
 - Thuật toán có độ phức tạp thời gian tuyến tính $O(|E|)$, với $|E|$ là số lượng cạnh trong đồ thị. Điều này nghĩa là thuật toán hoạt động nhanh chóng

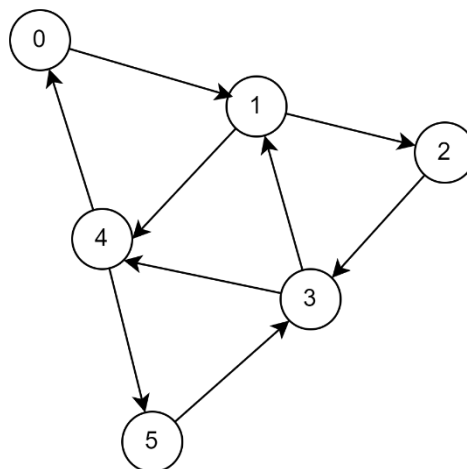
và hiệu quả, đặc biệt cho các đồ thị có số lượng cạnh lớn nên có hiệu quả về mặt thời gian.

- Thuật toán có thể được áp dụng cho nhiều loại đồ thị khác nhau, miễn là đồ thị có mạch Euler nên có tính linh hoạt cao.

- Nhược điểm:

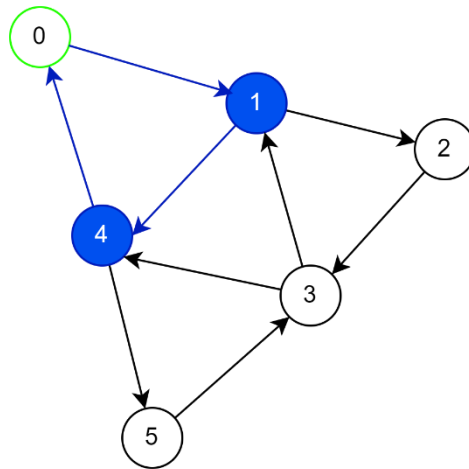
- Thuật toán Hierholzer chỉ có thể được sử dụng để tìm mạch Euler trong đồ thị. Nếu đồ thị không có mạch Euler, thuật toán sẽ không hoạt động.
- Đồ thị cần phải liên thông, nghĩa là có đường đi giữa bất kỳ hai đỉnh nào. Nếu đồ thị không liên thông, thuật toán sẽ không thể tìm được mạch Euler.
- Nếu đồ thị có nhiều mạch Euler, thuật toán có thể chỉ tìm ra một trong số đó.

6.2.5 Ví dụ:



Hình 6.3 Đồ thị ví dụ cho thuật toán Hierholzer

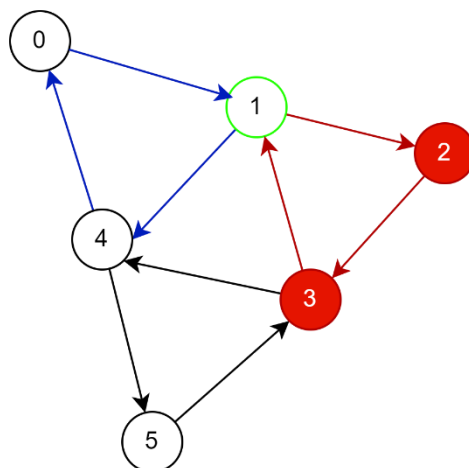
6.2.5.1 Tìm circuit chính:



Hình 6.4 Đồ thị ví dụ cho thuật toán Hierholzer 1.1

- Tìm circuit chính của đồ thị bằng cách chọn đại một đỉnh. Trường hợp này chọn đỉnh 0 để bắt đầu.
- Khi đó, tạo ra một sub-cycle: 0, 1, 4, 0. Thêm sub-cycle vào mạch chính. Sau đó xét, các đường rẽ nhánh kế tiếp, do nối với đỉnh 0 là đỉnh 1 nên ta sẽ duyệt sub-cycle bắt đầu bằng đỉnh 1.

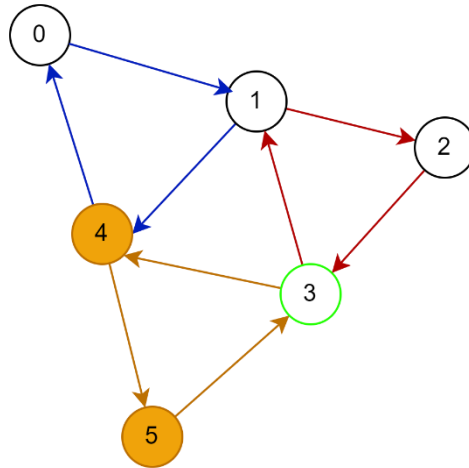
6.2.5.2 Tìm các đường rẽ nhánh:



Hình 6.5 Đồ thị ví dụ cho thuật toán Hierholzer 1.2

- Duyệt sub-cycle mới: 1, 2, 3, 1. Ta sẽ thêm sub-cycle mới vào mạch chính (mạch chính hiện tại: 0, 1, 4, 0) vào ở vị trí đỉnh 1 trong mạch chính.

- Mạch chính mới: 0, 1, 2, 3, 1, 4, 0.
- Sau đó, duyệt tiếp đỉnh 3 vì mỗi đỉnh 3 là còn đỉnh có đỉnh kề chưa được duyệt qua.



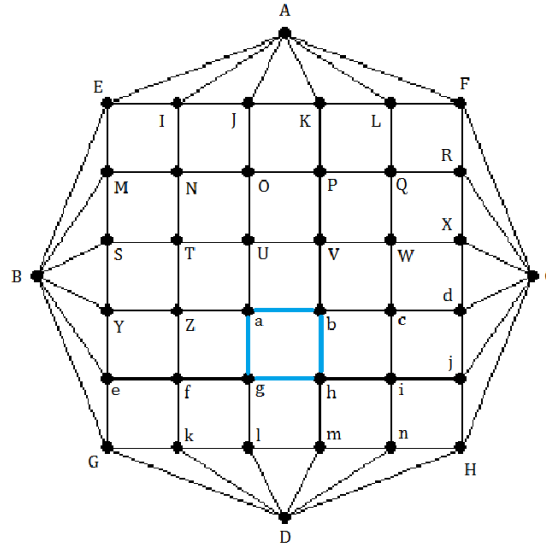
Hình 6.6 Đồ thị ví dụ cho thuật toán Hierholzer 1.3

- Duyệt sub-cycle mới: 3, 4, 5, 3. Ta sẽ thêm sub-cycle mới vào mạch chính vào ở vị trí đỉnh 3 trong mạch chính và đã duyệt qua được tất cả các cạnh của đồ thị.
- Mạch chính mới: 0, 1, 2, 3, 4, 5, 3, 1, 4, 0.

6.2.5.3 Kết luận:

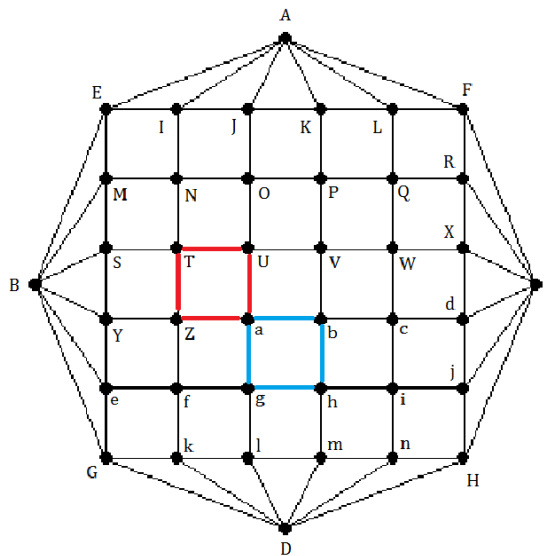
Vậy đồ thị có Eulerian Path là 0, 1, 2, 3, 4, 5, 3, 1, 4, 0.

6.3 Áp dụng thuật toán Hierholzer tìm Eulerian circuit của đồ thị khi có initial circuit R1:



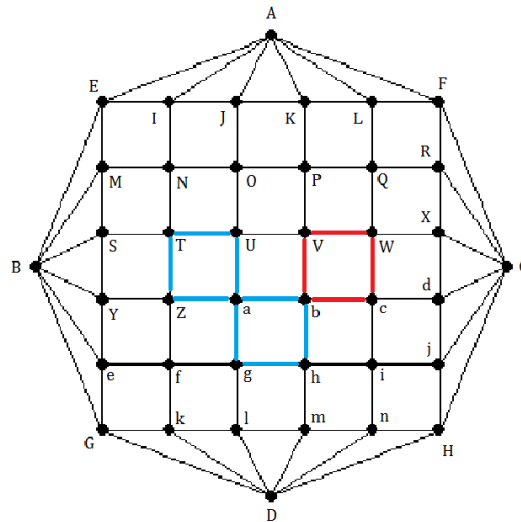
Hình 6.7 – Áp dụng thuật toán vào đồ thị gốc

- Mã số sinh viên: 52200033 \rightarrow R1 = a, b, h, g, a



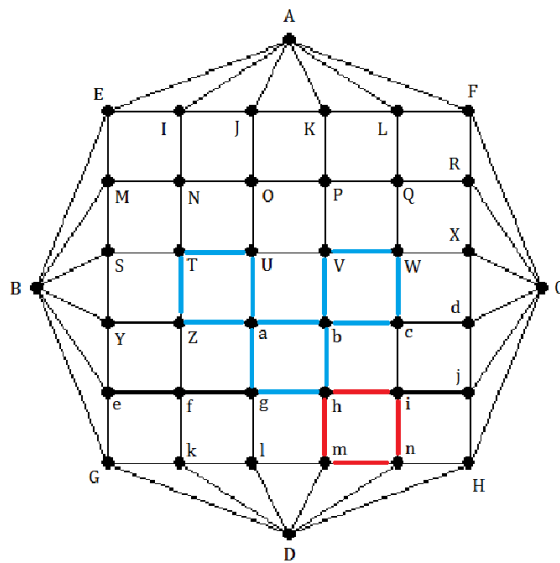
Hình 6.8 Áp dụng thuật toán vào đồ thị gốc 1.1

- Chọn đỉnh a, tạo sub-cycle từ đỉnh a: a, Z, T, U, a và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh a đầu tiên.
 \rightarrow R1 = a, Z, T, U, a, b, h, g, a



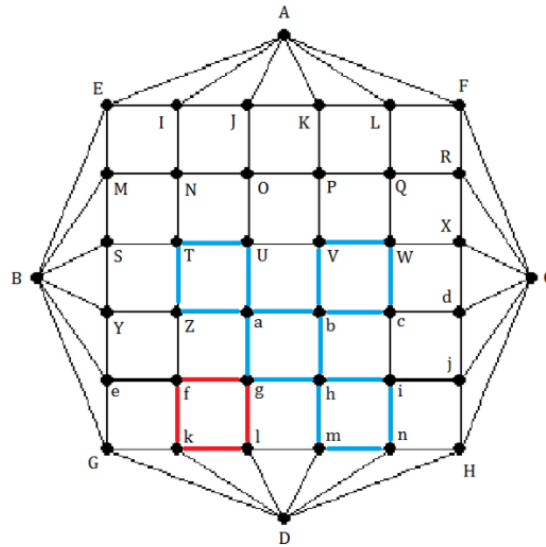
Hình 6.9 – Áp dụng thuật toán vào đồ thị gốc 1.2

- Chọn đỉnh b, tạo sub-cycle từ đỉnh b: a, V, W, c, b và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh b đầu tiên.
 $\rightarrow R1 = a, Z, T, U, a, b, V, W, c, b, h, g, a$



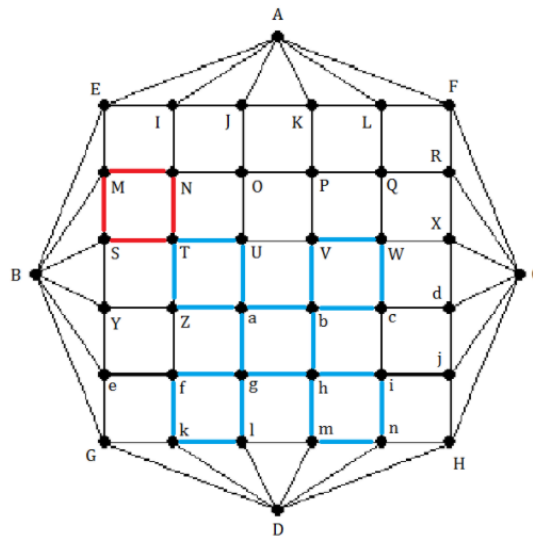
Hình 6.10 – Áp dụng thuật toán vào đồ thị gốc 1.3

- Chọn đỉnh h, tạo sub-cycle từ đỉnh h: h, i, n, m, h và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh h đầu tiên.
 $\rightarrow R1 = a, Z, T, U, a, b, V, W, c, b, h, i, n, m, h, g, a$



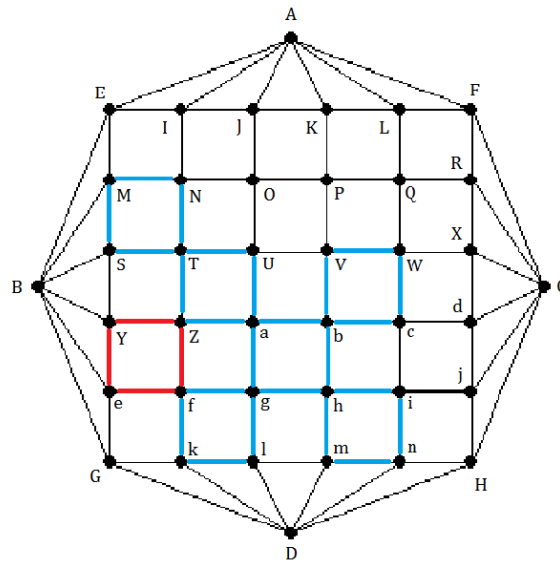
Hình 6.11 – Áp dụng thuật toán vào đồ thị gốc 1.4

- Chọn đỉnh g, tạo sub-cycle từ đỉnh g: g, l, k, f, g và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh g đầu tiên.
 $\rightarrow R1 = a, Z, T, U, a, b, V, W, c, b, h, i, n, m, h, g, l, k, f, g, a$



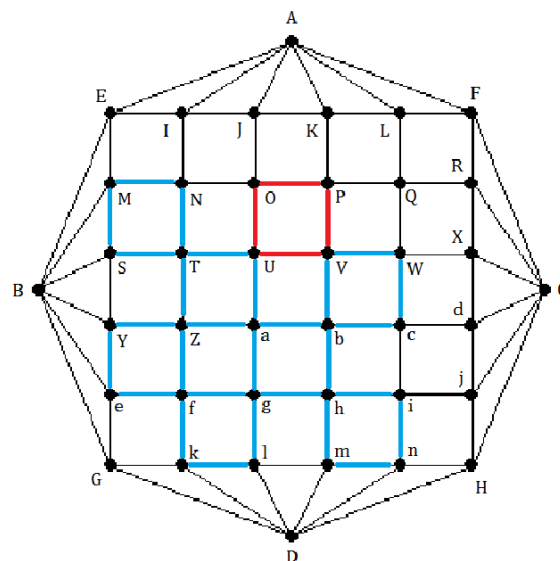
Hình 6.12

- Chọn đỉnh T, tạo sub-cycle từ đỉnh T: T, S, M, N, T và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh T đầu tiên.
 $\rightarrow R1 = a, Z, T, S, M, N, T, U, a, b, V, W, c, b, h, i, n, m, h, g, l, k, f, g, a$



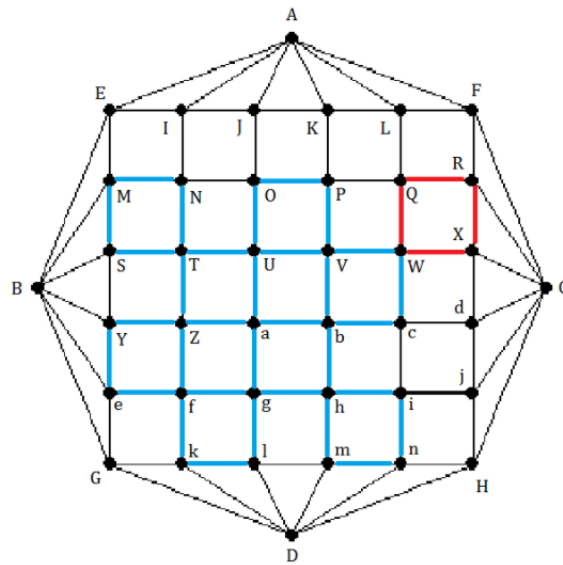
Hình 6.13 – Áp dụng thuật toán vào đồ thị gốc 1.6

- Chọn đỉnh Z, tạo sub-cycle từ đỉnh Z: Z, f, e, Y, Z và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh Z đầu tiên.
- R1 = a, Z, f, e, Y, Z, T, S, M, N, T, U, a, b, V, W, c, b, h, i, n, m, h, g, l, k, f, g, a



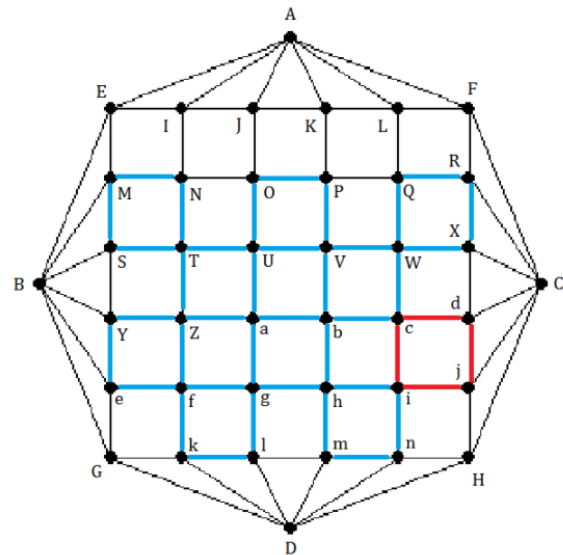
Hình 6.14 – Áp dụng thuật toán vào đồ thị gốc 1.7

- Chọn đỉnh U, tạo sub-cycle từ đỉnh U: U, O, P, V, U và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh U đầu tiên.
- R1 = a, Z, f, e, Y, Z, T, S, M, N, T, U, O, P, V, U, a, b, V, W, c, b, h, i, n, m, h, g, l, k, f, g, a



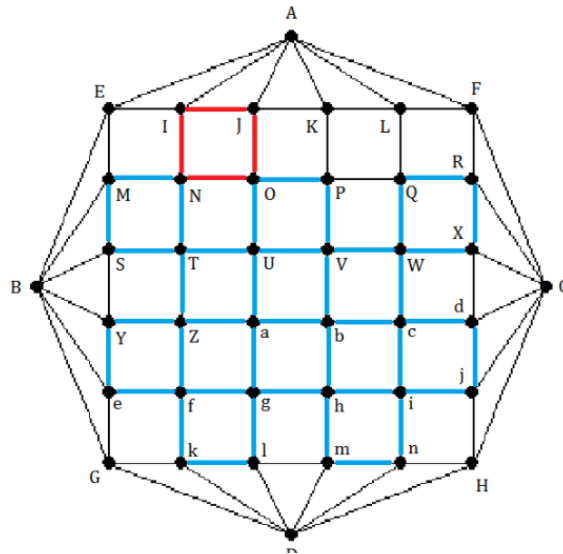
Hình 6.15 8

- Chọn đỉnh W, tạo sub-cycle từ đỉnh W: W, Q, R, X, W và sub-cycle thỏa là một circuit. Sau đó, thêm sub-cycle vào R1 vào vị trí đỉnh W đầu tiên.
 $\rightarrow R1 = a, Z, f, e, Y, Z, T, S, M, N, T, U, O, P, V, U, a, b, V, W, Q, R, X, W, c, b, h, i, n, m, h, g, l, k, f, g, a$



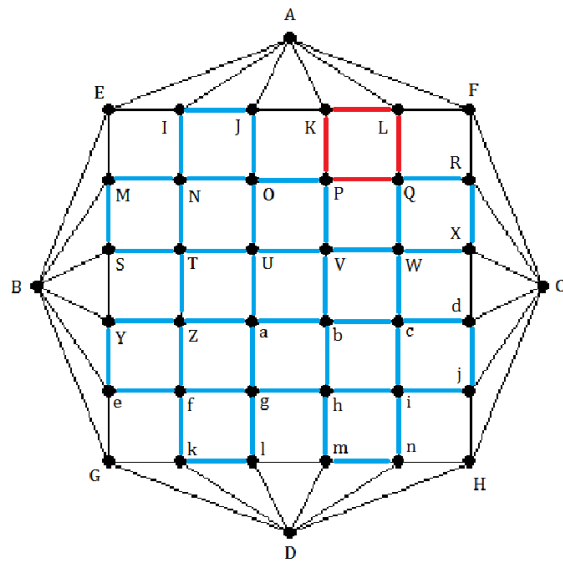
Hình 6.16 9

- Chọn đỉnh c, tạo sub-cycle từ đỉnh c: c, d, j, i, c và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh c đầu tiên.
 $\rightarrow R1 = a, Z, f, e, Y, Z, T, S, M, N, T, U, O, P, V, U, a, b, V, W, Q, R, X, W, c, d, j, i, c, b, h, i, n, m, h, g, l, k, f, g, a$



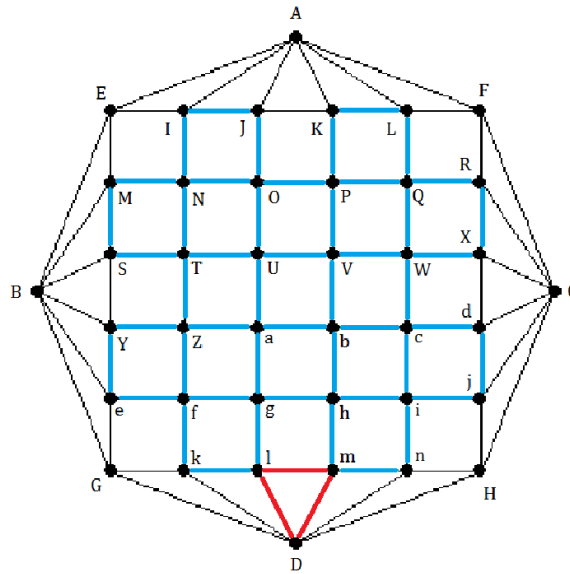
Hình 6.17 10

- Chọn đỉnh O, tạo sub-cycle từ đỉnh O: O, N, I, J, O và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh O đầu tiên.
 $\rightarrow R1 = a, Z, f, e, Y, Z, T, S, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, R, X, W, c, d, j, i, c, b, h, i, n, m, h, g, l, k, f, g, a$



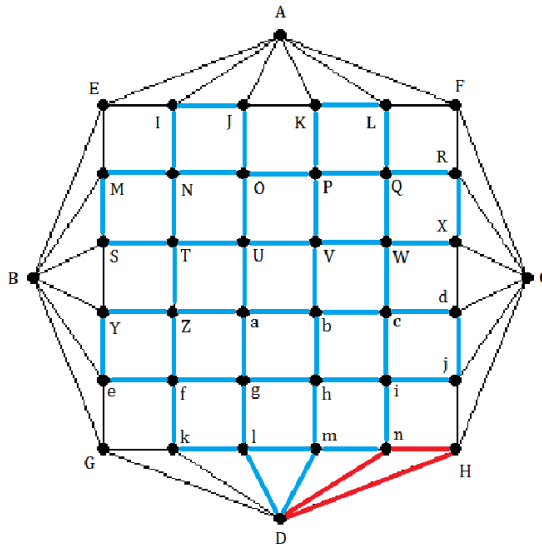
Hình 6.18 11

- Chọn đỉnh Q, tạo sub-cycle từ đỉnh Q: Q, P, K, L, Q và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh Q đầu tiên.
 $\rightarrow R1 = a, Z, f, e, Y, Z, T, S, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, X, W, c, d, j, i, c, b, h, i, n, m, h, g, l, k, f, g, a$



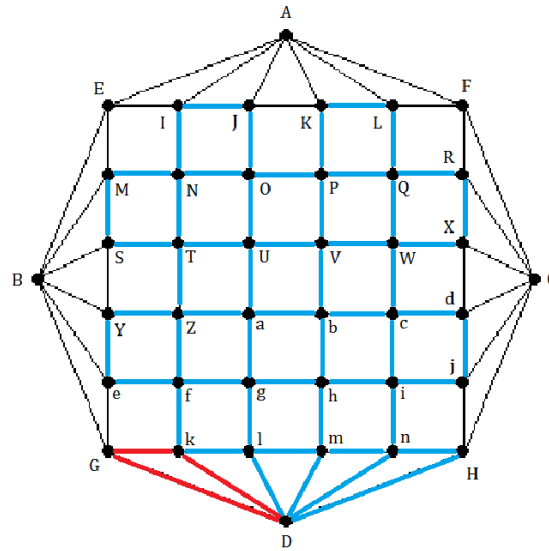
Hình 6.19 12

- Chọn đỉnh m , tạo sub-cycle từ đỉnh m : m, D, l, m và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào $R1$ vào vị trí đỉnh m đầu tiên.
 $\rightarrow R1 = a, Z, f, e, Y, Z, T, S, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, X, W, c, d, j, i, c, b, h, i, n, m, D, l, m, h, g, l, k, f, g, a$



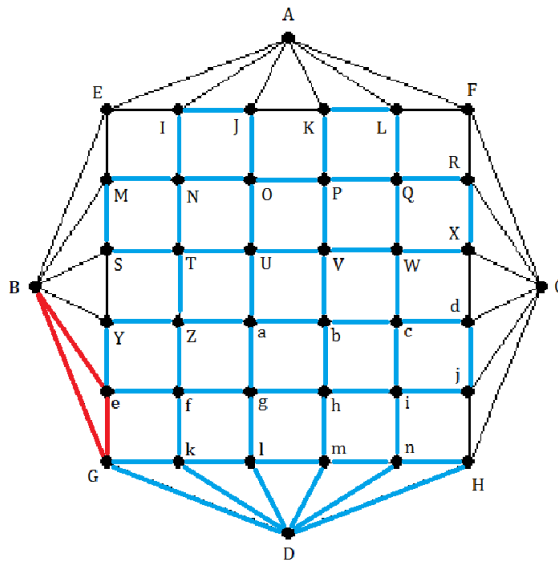
Hình 6.20 13

- Chọn đỉnh n , tạo sub-cycle từ đỉnh n : n, H, D, n và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào $R1$ vào vị trí đỉnh n đầu tiên.
 $\rightarrow R1 = a, Z, f, e, Y, Z, T, S, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, X, W, c, d, j, i, c, b, h, i, n, H, D, n, m, D, l, m, h, g, l, k, f, g, a$



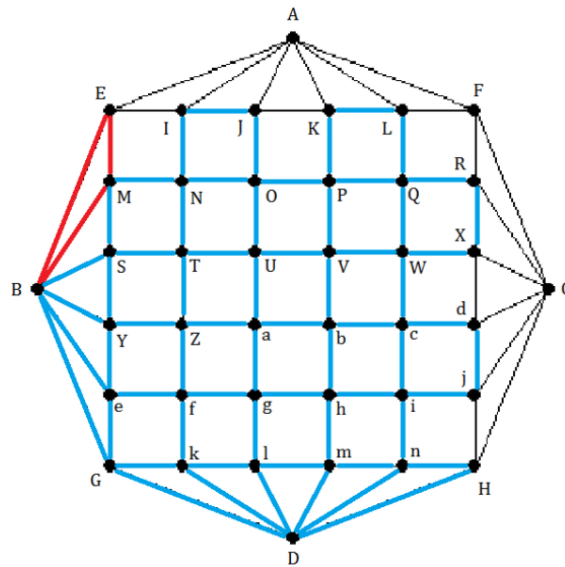
Hình 6.21 14

- Chọn đỉnh k, tạo sub-cycle từ đỉnh k: k, D, G, k và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh k đầu tiên.
 $\rightarrow R1 = a, Z, f, e, Y, Z, T, S, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, X, W, c, d, j, i, c, b, h, i, n, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a$



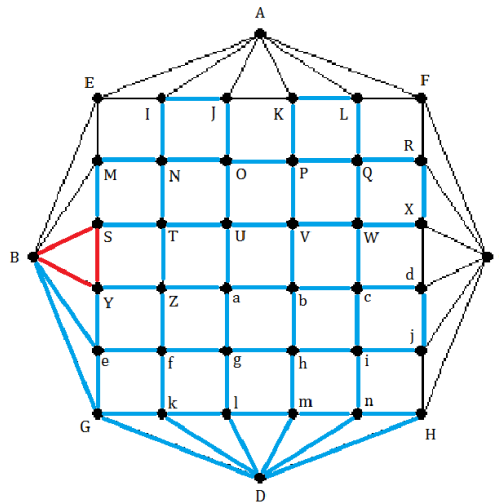
Hình 6.22 15

- Chọn đỉnh e, tạo sub-cycle từ đỉnh e: e, G, B, e và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh e đầu tiên.
 $\rightarrow R1 = a, Z, f, e, G, B, e, Y, Z, T, S, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, X, W, c, d, j, i, c, b, h, i, n, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a$



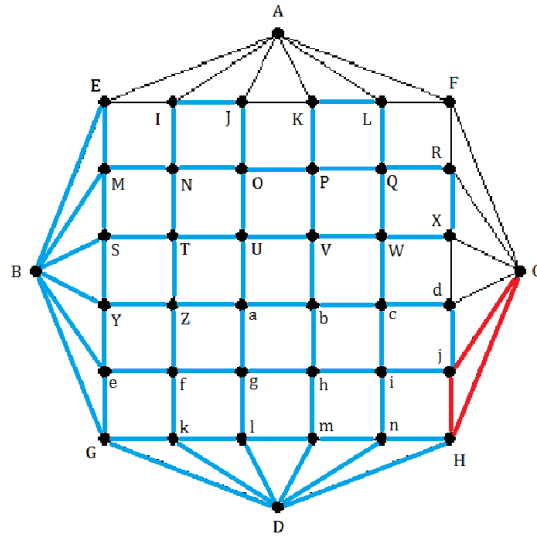
Hình 6.23 16

- Chọn đỉnh Y, tạo sub-cycle từ đỉnh Y: Y, B, S, Y và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh Y đầu tiên.
 $\rightarrow R1 = a, Z, f, e, G, B, e, Y, B, S, Y, Z, T, S, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, X, W, c, d, j, i, c, b, h, i, n, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a$



Hình 6.24 17

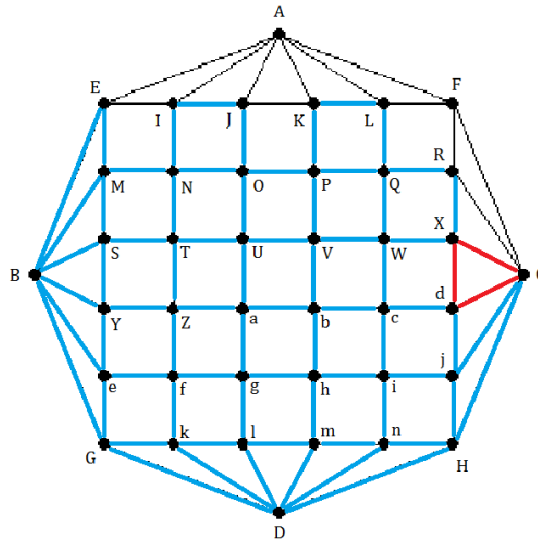
- Chọn đỉnh M, tạo sub-cycle từ đỉnh M: M, B, E, M và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh M đầu tiên.
 $\rightarrow R1 = a, Z, f, e, G, B, e, Y, B, S, Y, Z, T, S, M, B, E, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, X, W, c, d, j, i, c, b, h, i, n, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a$



Hình 6.25 18

- Chọn đỉnh H, tạo sub-cycle từ đỉnh H: H, j, C, H và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh H đầu tiên.

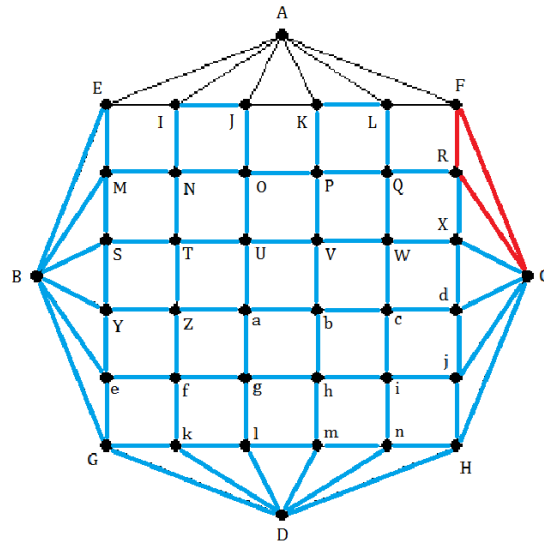
→ R1 = a, Z, f, e, G, B, e, Y, B, S, Y, Z, T, S, M, B, E, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, X, W, c, d, j, i, c, b, h, i, n, H, j, C, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a



Hình 6.26 19

- Chọn đỉnh d, tạo sub-cycle từ đỉnh d: d, X, C, d và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh d đầu tiên.

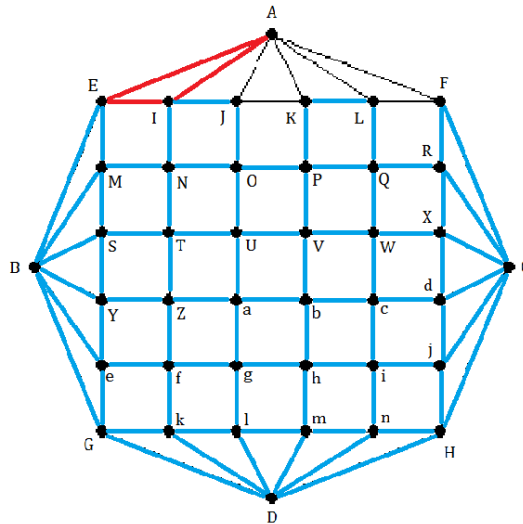
→ R1 = a, Z, f, e, G, B, e, Y, B, S, Y, Z, T, S, M, B, E, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, X, W, c, d, X, C, d, j, i, c, b, h, i, n, H, j, C, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a



Hình 6.27 20

- Chọn đỉnh R, tạo sub-cycle từ đỉnh R: R, F, C, R và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh R đầu tiên.

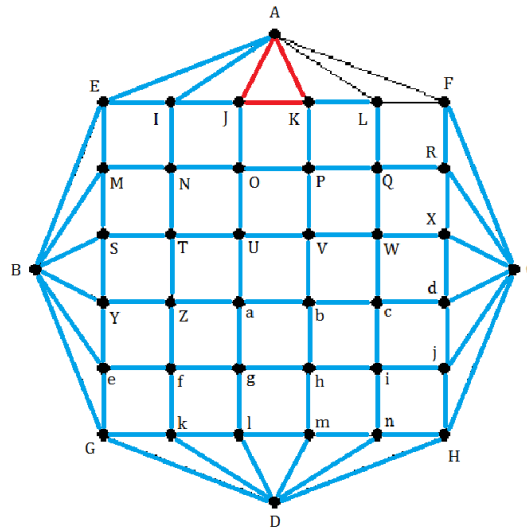
→ R1 = a, Z, f, e, G, B, e, Y, B, S, Y, Z, T, S, M, B, E, M, N, T, U, O, N, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, F, C, R, X, W, c, d, X, C, d, j, i, c, b, h, i, n, H, j, C, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a



Hình 6.28 21

- Chọn đỉnh I, tạo sub-cycle từ đỉnh I: I, A, E, I và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh I đầu tiên.

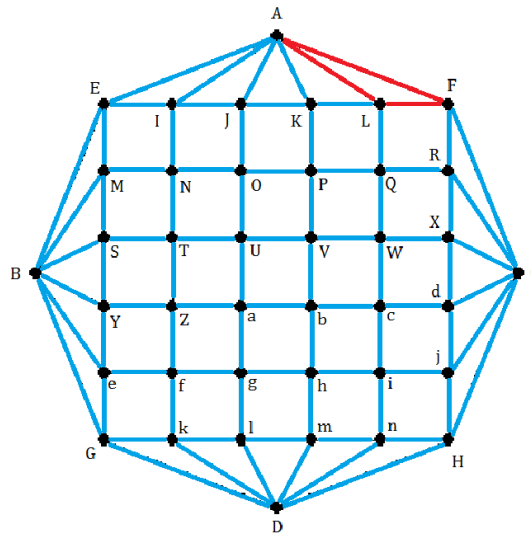
→ R1 = a, Z, f, e, G, B, e, Y, B, S, Y, Z, T, S, M, B, E, M, N, T, U, O, N, I, A, E, I, J, O, P, V, U, a, b, V, W, Q, P, K, L, Q, R, F, C, R, X, W, c, d, X, C, d, j, i, c, b, h, i, n, H, j, C, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a



Hình 6.29 22

- Chọn đỉnh K, tạo sub-cycle từ đỉnh K: K, J, A, K và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh K đầu tiên.

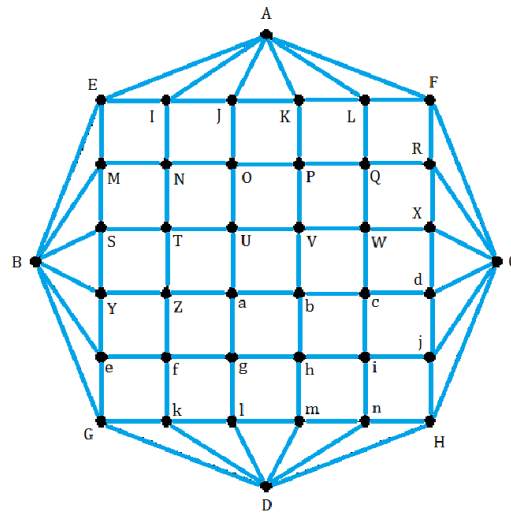
→ R1 = a, Z, f, e, G, B, e, Y, B, S, Y, Z, T, S, M, B, E, M, N, T, U, O, N, I, A, E, I, J, O, P, V, U, a, b, V, W, Q, P, K, J, A, K, L, Q, R, F, C, R, X, W, c, d, X, C, d, j, i, c, b, h, i, n, H, j, C, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a



Hình 6.30 23

- Chọn đỉnh L, tạo sub-cycle từ đỉnh L: L, A, F, L và sub-cycle thỏa là một circuit. Sau đó, tiến hành thêm sub-cycle vào R1 vào vị trí đỉnh L đầu tiên.

→ R1 = a, Z, f, e, G, B, e, Y, B, S, Y, Z, T, S, M, B, E, M, N, T, U, O, N, I, A, E, I, J, O, P, V, U, a, b, V, W, Q, P, K, J, A, K, L, A, F, L, Q, R, F, C, R, X, W, c, d, X, C, d, j, i, c, b, h, i, n, H, j, C, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a



Hình 6.31 24

- Vậy đồ thị đã được duyệt qua tất cả các cạnh.

Kết luận: Eulerian circuit của đồ thị khi R1 là abhga: a, Z, f, e, G, B, e, Y, B, S, Y, Z, T, S, M, B, E, M, N, T, U, O, N, I, A, E, I, J, O, P, V, U, a, b, V, W, Q, P, K, J, A, K, L, A, F, L, Q, R, F, C, R, X, W, c, d, X, C, d, j, i, c, b, h, i, n, H, j, C, H, D, n, m, D, l, m, h, g, l, k, D, G, k, f, g, a

CHƯƠNG 7. MAP COLORING

7.1 Mô hình hoá biểu đồ bằng đồ thị:

- Biểu đồ gốc:



Hình 7.1 Biểu đồ gốc

- Từ biểu đồ gốc, ta có:

- 36 đỉnh:

1	Jammu and Kashmir	2	Uttar Pradesh	3	Chhattisgarh	4	Orissa
5	Himachal Pradesh	6	Gujarat	7	Goa	8	Sikkim
9	Chandigarh	10	Daman and Diu	11	Karnataka	12	West Bengal
13	Punjab	14	Dadra and Nagar Haveli	15	Telangana	16	Arunachal Pradesh
17	Uttarakhand	18	Maharashtra	19	Andhra Pradesh	20	Assam
21	Haryana	22	Madhya Pradesh	23	Pondicherry	24	Meghalaya
25	Delhi	26	Bihar	27	Tamil Nadu	28	Nagaland
29	Rajasthan	30	Jharkhand	31	Kerala	32	Manipur
33	Tripura	34	Mizoram	35	Andaman and Nicobar Islands	36	Lakshadweep

Bảng 7.1 Danh sách các tỉnh có trong biểu đồ

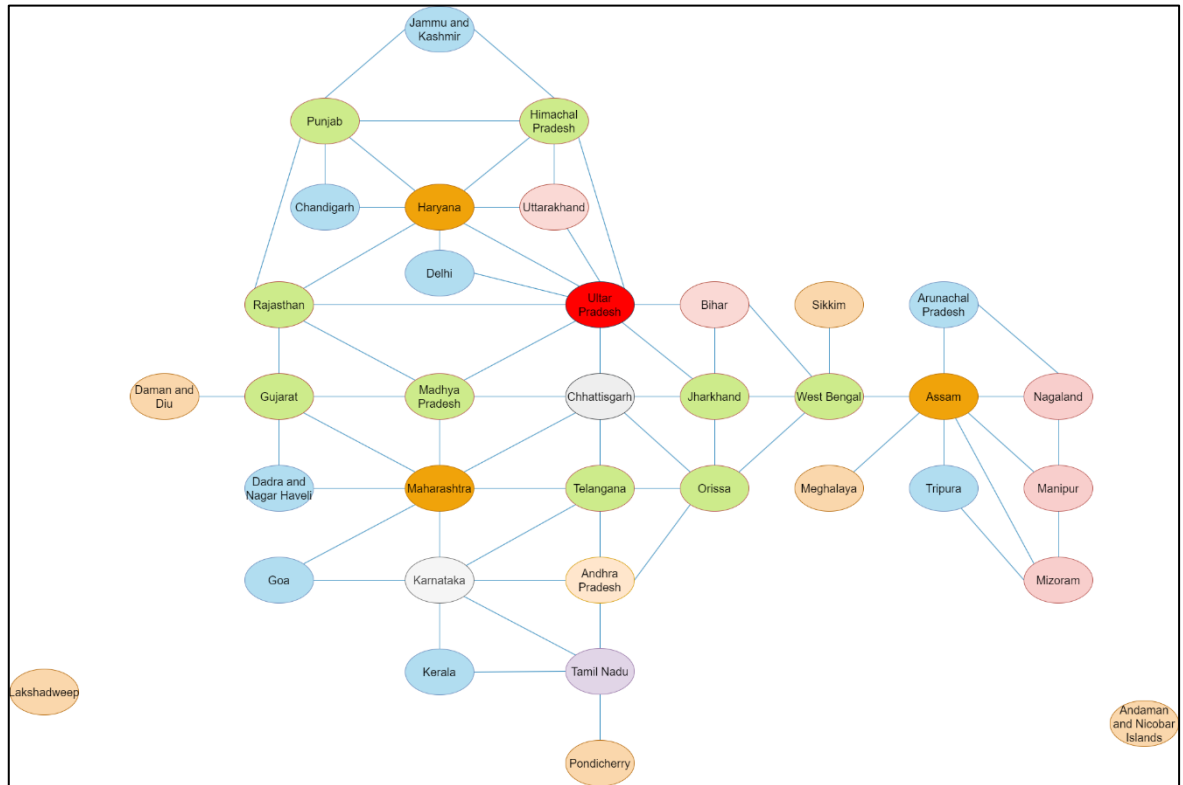
- 65 cạnh:

1. Jammu and Kashmir và Punjab
2. Jammu and Kashmir và Himachal Pradesh
3. Punjab và Himachal Pradesh
4. Punjab và Chandigarh
5. Punjab và Rajasthan
6. Punjab và Haryana
7. Himachal Pradesh và Haryana
8. Himachal Pradesh và Uttarakhand

9. Nagaland và Arunachal Pradesh
10. Himachal Pradesh và Uttar Pradesh
11. Chandigarh và Haryana
12. Haryana và Uttarakhand
13. Haryana và Rajasthan
14. Haryana và Delhi
15. Haryana và Uttar Pradesh
16. Uttarakhand và Uttar Pradesh
17. Delhi và Uttar Pradesh
18. Rajasthan và Uttar Pradesh
19. Rajasthan và Gujarat
20. Rajasthan và Madhya Pradesh
21. Rajasthan và Daman and Diu
22. Gujarat và Madhya Pradesh
23. Gujarat và Dadra and Nagar Haveli
24. Gujarat và Maharashtra
25. Madhya Pradesh và Uttar Pradesh
26. Madhya Pradesh và Chhattisgarh
27. Madhya Pradesh và Maharashtra
28. Maharashtra và Dadra and Nagar Haveli
29. Maharashtra và Chhattisgarh
30. Maharashtra và Telangana
31. Maharashtra và Goa
32. Maharashtra và Karnataka
33. Karnataka và Goa
34. Karnataka và Kerala
35. Karnataka và Telangana
36. Karnataka và Andhra Pradesh
37. Karnataka và Tamil Nadu

- 38. Kerala và Tamil Nadu
- 39. Tamil Nadu và Pondicherry
- 40. Tamil Nadu và Andhra Pradesh
- 41. Andhra Pradesh và Telangana
- 42. Telangana và Chhattisgarh
- 43. Telangana và Orissa
- 44. Orissa và Andhra Pradesh
- 45. Orissa và Chhattisgarh
- 46. Orissa và Jharkhand
- 47. Orissa và West Bengal
- 48. Jharkhand và Uttar Pradesh
- 49. Jharkhand và Bihar
- 50. Jharkhand và West Bengal
- 51. Chhattisgarh và Jharkhand
- 52. Chhattisgarh và Uttar Pradesh
- 53. Bihar và West Bengal
- 54. Bihar và Uttar Pradesh
- 55. West Bengal và Sikkim
- 56. West Bengal và Assam
- 57. Assam và Meghalaya
- 58. Assam và Tripura
- 59. Assam và Arunachal Pradesh
- 60. Assam và Nagaland
- 61. Assam và Manipur
- 62. Assam và Mizoram
- 63. Tripura và Mizoram
- 64. Mizoram và Manipur
- 65. Manipur và Nagaland

- Ta sử dụng dữ liệu 36 đỉnh và 65 cạnh để đồ thị hoá:







Hình 7.2 Đồ thị hoá

- Lưu ý: Màu được sử dụng để tô các đỉnh tuân theo nguyên tắc:
 - Màu cam nhạt: đỉnh bậc 1 và bậc 0
 - Màu xanh biển: đỉnh bậc 2
 - Màu hồng: đỉnh bậc 3
 - Màu tím: đỉnh bậc 4
 - Màu xanh lá: đỉnh bậc 5
 - Màu xám: đỉnh bậc 6
 - Màu cam đậm: đỉnh bậc 7
 - Màu đỏ: đỉnh bậc 9

7.2 Tô màu đồ thị:








- Với mã số sinh viên: 52200033 thì có 4 số cuối là $0033 \% 4 = 1$. Nên bắt đầu tô màu từ tỉnh Orissa.











- Bảng màu em sẽ sử dụng:









Mã màu	Màu	Mã màu	Màu
#F8CECC		#D5E8D4	
#DAE8FC		#FFF2CC	












Bảng 7.2 Danh sách bảng màu

- Sau đây là các bước tô màu đồ thị:

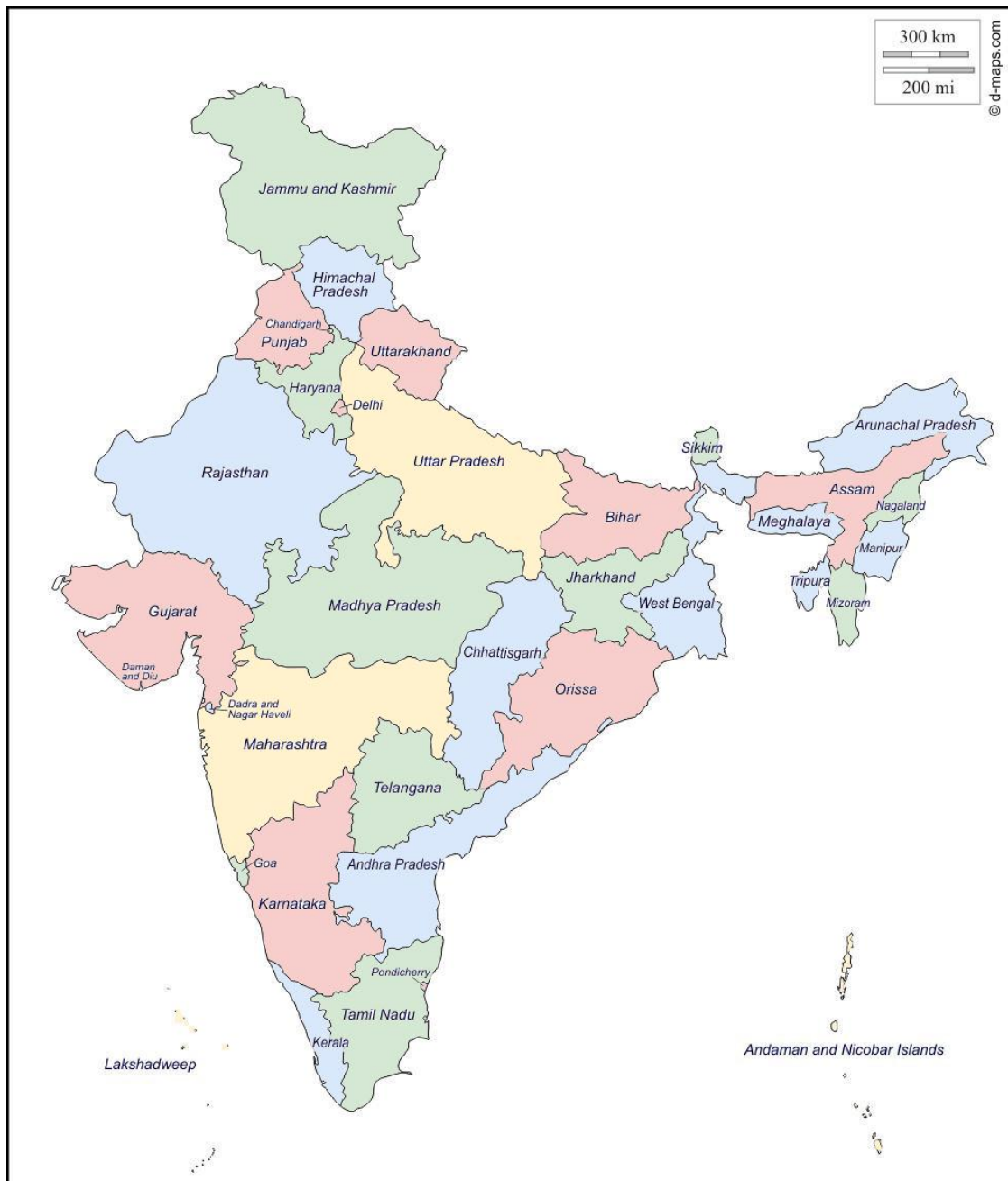
Bước	Tỉnh tô màu	Màu sử dụng	Ghi chú
1	Orissa		Tô Orissa màu hồng
2	West Bengal		West Bengal liền kề với: Orissa (hồng)
3	Jharkhand		Jharhand liền kề với: 1. Orissa(hồng) 2. West Bengal (xanh biển)
4	Chhattisgarh		Chhattisgarh liền kề với: 1. Orissa (hồng) 2. JharkHand (xanh lá)
5	Telangana		Telangana liền kề với: 1. Orissa (hồng) 2. Chhattisgarh (xanh biển)
6	Andhra Pradesh		Andhra Pradesh liền kề với: 1. Orissa (hồng) 2. Telangana (xanh lá)
7	Karnataka		Karnataka liền kề với: 1. Andhra Pradesh (xanh biển) 2. Telangana (xanh lá)

8	Tamil Nadu		Tamil Nadu liền kề với: 1. Andhra Pradesh (xanh biển) 2. Karnataka (hồng)
9	Kerala		Kerala liền kề với: 1. Tamil Nadu (xanh lá) 2. Karnataka (hồng)
10	Pondicherry		Pondicherry liền kề: Tamil Nadu (xanh lá)
11	Goa		Goa liền kề với: Karnataka (hồng)
12	Maharashtra		Maharashtra liền kề với: 1. Goa (xanh lá) 2. Karnataka (hồng) 3. Telangana (xanh lá) 4. Chhattisgarh (xanh biển)
13	Dadra and Nagar Haveli		Dadra and Nagar Haveli liền kề với: Maharashtra (vàng)
14	Gujarat		Gujarat liền kề với: 1. Dadra and Nagar Haveli (xanh biển) 2. Maharashtra (vàng)
15	Daman and Diu		Daman and Diu liền kề với: Gujarat (hồng)
16	Madhya Pradesh		Madhya Pradesh liền kề với: 1. Chhattisgarh (xanh biển) 2. Maharashtra (vàng) 3. Gujarat (hồng)
17	Rajasthan		Rajasthan liền kề với: 1. Gujarat (hồng) 2. Madhya Pradesh (xanh lá)

18	Haryana		Haryana liền kề với Rajasthan (xanh biển)
19	Delhi		Delhi liền kề với Haryana (xanh lá)
20	Punjab		Punjab liền kề với: 1. Haryana (xanh lá) 2. Rajasthan (xanh biển)
21	Chandigarh		Chandigarh liền kề với: 1. Haryana (xanh lá) 2. Punjab (hồng)
22	Jammu and Kashmir		Jammu and Kashmir liền kề: Punjab (hồng)
23	Himachal Pradesh		Himachal Pradesh liền kề với: 1. Punjab (hồng) 2. Jammu and Kashmir (xanh lá) 3. Haryana (xanh lá)
24	Uttarakhand		Uttarakhand liền kề với: 1. Himachal Pradesh (xanh biển) 2. Haryana (xanh lá)
25	Uttar Pradesh		Uttar Pradesh liền kề với: 1. Uttarakhand (hồng) 2. Himachal Pradesh (xanh biển) 3. Delhi (hồng) 4. Haryana (xanh lá) 5. Madhya Pradesh (xanh lá) 6. Rajasthan (xanh biển) 7. Chhattisgarh (xanh biển) 8. Jharkhand (xanh lá)

26	Bihar		Bihar liên kề với: 1. Uttar Pradesh (vàng) 2. Jharkhand (xanh lá) 3. West Bengal (xanh biển)
27	Sikkim		Sikkim liên kề với: West Bengal (xanh biển)
28	Assam		Assam liên kề với: West Bengal (xanh biển)
29	Meghalaya		Meghalaya liên kề với: Assam (hồng)
30	Tripura		Tripura liên kề với: Assam (hồng)
31	Mizoram		Mizoram liên kề với: 1. Assam (hồng) 2. Tripura (xanh biển)
32	Manipur		Manipur liên kề với: 1. Assam (hồng) 2. Mizoram (xanh lá)
33	Nagaland		Nagaland liên kề với: 1. Assam (hồng) 2. Manipur (xanh biển)
34	Arunachal Pradesh		Arunachal Pradesh liên kề với: 1. Assam (hồng) 2. Nagaland (xanh lá)
35	Andaman and Nicobar Islands		Đỉnh Andaman and Nicobar Islands không liên kề với đỉnh nào
36	Lakshadweep		Đỉnh Lakshadweep không liên kề với đỉnh nào

Bảng 7.3 Các bước tô màu đồ thị



Hình 7.3 Đồ thị đã được tô màu

Như vậy, Đồ thị đã được tô màu bằng 4 màu mà các tỉnh liền kề phân biệt màu với nhau

CHƯƠNG 8. FINDING AN INVERSE MODULO

8.1 Cơ sở lý thuyết

8.1.1 Phép tính đồng dư modulo

- Ta có a, b là các số nguyên thuộc \mathbb{Z} , khi lấy a / b thì ta được thương số là t và r là số dư của phép chia. Khi đó ta nói: a đồng dư với r trong modulo cho b .

Ký hiệu toán học:

$$a \equiv r \pmod{b} \quad (8.1)$$

- Ví dụ: Ta có 2 số nguyên là 7 và 2. Khi đó $7/2 = 3$ dư 1. Khi đó ta nói:

$$7 \equiv 1 \pmod{2}$$

- Một cách tổng quát: Nếu $a \equiv b \pmod{m} \Rightarrow a = km + b$
- Phép tính đồng dư modulo m được gọi là phép tính trong $\mathbb{Z}/m\mathbb{Z}$ [2]

8.1.2 Số nguyên tố cùng nhau và phần tử nghịch đảo

- Trong phần 1.1, em đã có nhắc đến thuật toán Euclid sử dụng để tìm ƯCLN, khi 2 số nguyên tố a và b có ƯCLN bằng 1 thì a và b được gọi là hai số nguyên tố cùng nhau.
- Trong tập số thực \mathbb{R} , a^{-1} là số nghịch đảo của a nếu $a^{-1} \times a = 1$. Nhưng trong modulo $\mathbb{Z}/m\mathbb{Z}$, a^{-1} là số nghịch đảo của a nếu $a^{-1} \times a = 1 \pmod{m}$. [2]
- Ví dụ: $1 = 5 \times 3 - 2 \times 7 \Rightarrow 5 \times 3 \equiv 1 \pmod{7}$
- Suy ra: 5 là phần tử nghịch đảo của 3 trong $\mathbb{Z}/7\mathbb{Z}$
- Ngoài ra, ví dụ trên còn được viết như sau:
 $1 = 5 \times 3 - 2 \times 7 \Rightarrow -2 \times 7 \equiv 1 \pmod{3}$, -2 là phần tử nghịch đảo của 7 trong $\mathbb{Z}/3\mathbb{Z}$

- Từ đó, ta có kết luận: Chỉ những số nguyên tố cùng nhau mới có phần tử nghịch đảo.

8.1.3 Áp dụng thuật toán Euclid mở rộng vào bài toán tìm phần tử nghịch đảo

- Bổ đề Bézout (Thuyết 4.5.3 [1]) khẳng định rằng: Với hai số nguyên không âm a, m có ƯCLN là d , tồn tại hai số nguyên x và y sao cho: $ax + my = d$. Phương trình có thể được viết lại: $\frac{a}{d}x + \frac{m}{d}y = 1$.
- Theo bổ đề Bezout và Phương trình Diophantos, nếu:
 $d = \gcd(a, m) = 1$ thì ta luôn luôn tìm được 2 số nguyên x và y thoả mãn: $a \times x + m \times y = 1$. Vì ta đang làm việc trên modulo m , ta có thể bỏ $m \times y$ và viết lại đẳng thức trên như sau: $a \times x \equiv 1 \pmod{m}$

Do đó, x chính là a^{-1} hay còn gọi là phần tử nghịch đảo của a .

8.1.4 Ví dụ áp dụng

Cho 2 số nguyên tố cùng nhau là 97 và 13. Tìm $13^{-1} \pmod{97}$

Giải:

- **Áp dụng thuật toán Euclid để tìm ƯCLN của 13 và 97:**

1. $97 = 13 \times 7 + 6$
2. $13 = 6 \times 2 + 1$
3. $6 = 1 \times 6 + 0$
4. Vậy $\gcd(13, 97) = 1$

- **Áp dụng Thuật toán Euclid Mở rộng để Tìm Hệ số Bézout:**

- Ở bước này, em sẽ áp dụng thuật toán Euclid mở rộng để tìm hệ số x và m sao cho: $13x + 97m = 1$
- Các bước ngược lại của thuật toán Euclid:
 1. $1 = 13 - 6 \times 2$ (1)
 2. $6 = 97 - 13 \times 7$
- Thay thế giá trị của 6 từ phương trình thứ hai vào phương trình (1):

$$1 = 13 - (97 - 13 \times 7) \times 2 = 13 - 2 \times 97 + 13 \times 14 = 13 \times 15 - 97 \times 2$$

- Như vậy, ta có: $13 \times 15 - 97 \times 2 = 1$.

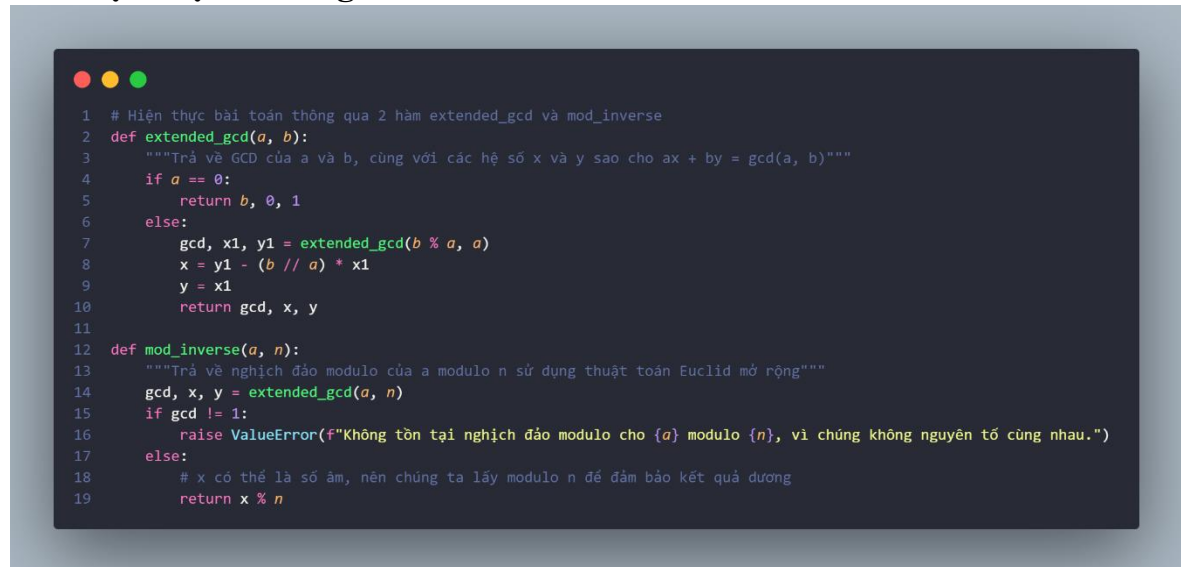
Do đó, hệ số x là 15 và hệ số m là -2.

- *Tìm nghịch đảo modulo:*

Hệ số x mà em tìm được là 15. Vì 15 thỏa mãn phương trình $13x + 97m = 1$, ta có: $13 \times 15 \equiv 1 \pmod{97}$

Vậy $13^{-1} \pmod{97} = 15$

8.2 Hiện thực chương trình



Hình 8.1 Code hiện thực tìm modulo nghịch đảo n

- *Hàm extended_gcd:*

- Hàm này hiện thực thuật toán Euclid mở rộng.
- Nó đệ quy tìm UCLN của a và b , cũng như các hệ số x và y sao cho $ax + by = 1$

- *Hàm mod_inverse:*

- Hàm này sử dụng *extended_gcd* để tìm nghịch đảo modulo.
- Nếu UCLN của a và n không phải là 1, nó sẽ ném ra lỗi *ValueError* vì nghịch đảo modulo không tồn tại.
- Ngược lại, nó trả về x modulo n , đảm bảo kết quả là một số nguyên dương.

8.3 Kiểm thử & xác thực

8.3.1 Kiểm tra chương trình bằng dữ liệu mẫu

```

1 # Kiểm tra chương trình bằng dữ liệu mẫu
2 sample_a = 13
3 sample_n = 97
4 inverse = mod_inverse(sample_a, sample_n)
5 print(f"Nghịch đảo modulo của {sample_a} modulo {sample_n} là {inverse}")

```

Hình 8.2 Code kiểm tra chương trình bằng dữ liệu mẫu

- Biến *sample_a* là giá trị cần tìm nghịch đảo modulo của nó modulo biến *sample_b*
- Biến *inverse* là giá trị nghịch đảo của *sample_a* modulo *sample_b*
- Ở đây dữ liệu em test là 2 số nguyên tố đã được tính trong phần 8.1.4 để dễ kiểm tra lại kết quả.

8.3.2 Kết quả chạy

```

PS C:\Study\HK2 - 2023\CTRR\Final> python 52200205_52200051_52200033_8.py
Nghịch đảo modulo của 13 modulo 97 là 15

```

Hình 8.3 Kết quả chạy của bài toán $13^{-1} \bmod 97$

Kết quả chạy trùng khớp với làm tay ở phần 8.1.4.

8.3.3 Hậu kiểm kết quả

- Code hậu kiểm:

```

1 # Kiểm tra kết quả
2 if((sample_a * inverse) % sample_n == 1):
3     print(f"Hậu kiểm kết quả, phép tính ((sample_a) * {inverse}) % {sample_n} = {(sample_a * inverse) % sample_n} cho thấy phép tính nghịch đảo này là ĐÚNG")
4 else:
5     print(f"Hậu kiểm kết quả, phép tính ((sample_a) * {inverse}) % {sample_n} != {(sample_a * inverse) % sample_n} cho thấy phép tính nghịch đảo này là SAI")

```

Hình 8.4 Code hậu kiểm kết quả thực thi

- Để kiểm tra xem kết quả tìm được có đúng hay không, ta thực hiện phép tính: $(a \times inverse) \% n$. Nếu kết quả của phép tính này bằng 1, nghĩa là $inverse$ thực sự là nghịch đảo modulo của $a \bmod n$. Theo định nghĩa, nghịch đảo modulo của $a \bmod n$ là một số x sao cho $a \times x \equiv 1 \pmod{m}$. Nếu kiểm tra này đúng, điều đó chứng tỏ kết quả của em đáp ứng đúng định nghĩa này.
- Các trường hợp chạy của cả đoạn code, có cả hợp lệ lẫn không hợp lệ:

```
PS C:\Study\HK2 - 2023\CTRR\Final> python 52200205_52200051_52200033_8.py
Nghịch đảo modulo của 13 modulo 97 là 15
Hậu kiểm kết quả, phép tính (13 * 15) % 97 = 1 cho thấy phép tính nghịch đảo này là ĐÚNG
PS C:\Study\HK2 - 2023\CTRR\Final>
```

Hình 8.7 Kết quả chạy hậu kiểm của bài toán $13^{-1} \bmod 97$

```
PS C:\Study\HK2 - 2023\CTRR\Final> python 52200205_52200051_52200033_8.py
Traceback (most recent call last):
  File "C:\Study\HK2 - 2023\CTRR\Final\52200205_52200051_52200033_8.py", line 24, in <module>
    inverse = mod_inverse(sample_a, sample_n)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Study\HK2 - 2023\CTRR\Final\52200205_52200051_52200033_8.py", line 16, in mod_inverse
    raise ValueError(f"Không tồn tại nghịch đảo modulo cho {a} modulo {n}, vì chúng không nguyên tố cùng nhau.")
ValueError: Không tồn tại nghịch đảo modulo cho 8 modulo 12, vì chúng không nguyên tố cùng nhau.
PS C:\Study\HK2 - 2023\CTRR\Final>
```

Hình 8.6 Kết quả chạy hậu kiểm của bài toán $8^{-1} \bmod 12$

```
PS C:\Study\HK2 - 2023\CTRR\Final> python 52200205_52200051_52200033_8.py
Traceback (most recent call last):
  File "C:\Study\HK2 - 2023\CTRR\Final\52200205_52200051_52200033_8.py", line 24, in <module>
    inverse = mod_inverse(sample_a, sample_n)
              ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "C:\Study\HK2 - 2023\CTRR\Final\52200205_52200051_52200033_8.py", line 16, in mod_inverse
    raise ValueError(f"Không tồn tại nghịch đảo modulo cho {a} modulo {n}, vì chúng không nguyên tố cùng nhau.")
ValueError: Không tồn tại nghịch đảo modulo cho 13 modulo 13, vì chúng không nguyên tố cùng nhau.
PS C:\Study\HK2 - 2023\CTRR\Final>
```

Hình 8.5 Kết quả chạy hậu kiểm của bài toán $13^{-1} \bmod 13$

- Ở Hình 8.6 Kết quả chạy hậu kiểm của bài toán $8^{-1} \bmod 12$ và Hình 8.5 Kết quả chạy hậu kiểm của bài toán $13^{-1} \bmod 13$, vì dữ liệu đầu vào (2 số truyền vào) không nguyên tố cùng nhau nên chúng không có nghịch đảo modulo, chương trình sẽ văng ra lỗi để người dùng hậu kiểm.

CHƯƠNG 9. RSA CRYPTOSYSTEM

9.1 Hệ thống mã hóa RSA

- RSA (Rivest–Shamir–Adleman) là một hệ thống mật mã khóa công khai, một trong những hệ thống lâu đời nhất được sử dụng rộng rãi để truyền dữ liệu an toàn. [3]
- Thuật toán RSA hoạt động bằng cách sử dụng hai loại khóa: khóa công khai và khóa bí mật. Khóa công khai được công bố rộng rãi và dùng để mã hóa thông tin, trong khi khóa bí mật được giữ bí mật và chỉ dùng để giải mã thông tin đã được mã hóa bằng khóa công khai tương ứng.

9.1.1 Các khái niệm toán học cơ bản

- Số nguyên tố: Các số nguyên tố lớn là cơ bản trong RSA. Chúng được sử dụng để tạo ra các khóa.
- Số học modulo: Liên quan đến các phép toán với số dư và là trung tâm của quá trình mã hóa và giải mã.
- Thuật toán Euclid mở rộng: Được sử dụng để tìm ước chung lớn nhất (GCD) của hai số và cần thiết để tính toán nghịch đảo modulo.
- Phân tích số nguyên tố: Bảo mật của RSA dựa trên độ khó của việc phân tích tích của hai số nguyên tố lớn.

9.1.2 Cơ bản về mã RSA

Giả sử A và B đang muốn trao đổi thông tin bí mật thông qua Internet. Với thuật toán RSA, A và B đều tạo ra cặp khóa gồm khóa công khai và khóa bí mật thông qua các trình tự sau:

1. Chọn 2 số nguyên tố p và q , biết rằng p, q rất lớn và $p \neq q$
2. Tính $n = p \times q$
3. Tính giá trị của hàm số Euler: $\Phi(n) = (p - 1) \times (q - 1)$
4. Chọn $e \in \mathbb{N}$ sao cho $1 < e < \Phi(n)$ và $\text{UCLN}(\Phi(n), e) = 1$

5. Tính d với $e \times d = 1 \pmod{\Phi(n)}$

Trong đó: (n, e) là khóa công khai và d là khóa bí mật

A gửi thông tin x cho B, trước khi gửi, thông tin x sẽ được **biến đổi thành y** thông qua thuật toán RSA **bằng khóa công khai của B**:

- Mã hóa: $x^e \pmod{n} = y$

B nhận được thông tin y từ A, sử dụng khóa bí mật của mình để tìm ra thông tin gốc là x :

- Giải mã: $y^d \pmod{n} = x$

9.1.3 Vận hành RSA

Giả sử A và B đang muốn trao đổi thông tin bí mật thông qua Internet.

A chọn:

1. Chọn $p = 7, q = 13$
2. $n = p \times q = 7 \times 13 = 91$
3. $\Phi(n) = (p - 1) \times (q - 1) = (7 - 1) \times (13 - 1) = 72$
4. Chọn $e = 7$
5. $e \times d = 1 \pmod{\Phi(n)} \Rightarrow d = 31 \pmod{72}$ (Cách tính tương tự như ở CHƯƠNG 8.)

Khóa công khai của A là: (91, 7)

B chọn:

1. Chọn $p = 17, q = 19$
2. $n = p \times q = 17 \times 19 = 323$
3. $\Phi(n) = (p - 1) \times (q - 1) = (17 - 1) \times (19 - 1) = 288$
4. Chọn $e = 5$
5. $e \times d = 1 \pmod{\Phi(n)} \Rightarrow d = 173 \pmod{288}$ (Cách tính tương tự như ở CHƯƠNG 8.)

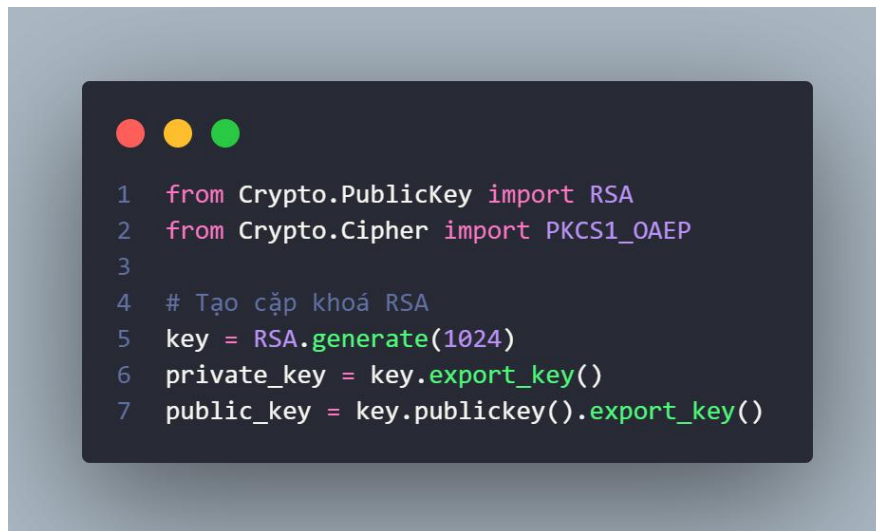
Khóa công khai của B là: (323, 5)

A muốn gửi số 27 cho B, số 27 sẽ được mã hóa như sau: $27^5 \bmod 323 = 278$.
A tiến hành gửi số 278 cho B.

B nhận được số 278 và giải mã như sau: $278^{173} \bmod 55 = 27$.

9.2 Hiện thực

Để hiện thực bài toán, em sẽ sử dụng thư viện *pycryptodome*. *pycryptodome* là một thư viện mã nguồn mở cung cấp các công cụ mật mã mạnh mẽ và hiệu quả cho Python. Nó là một nhánh của thư viện *PyCrypto* với nhiều cải tiến về bảo mật và hiệu suất.



```

1  from Crypto.PublicKey import RSA
2  from Crypto.Cipher import PKCS1_OAEP
3
4  # Tạo cặp khóa RSA
5  key = RSA.generate(1024)
6  private_key = key.export_key()
7  public_key = key.publickey().export_key()

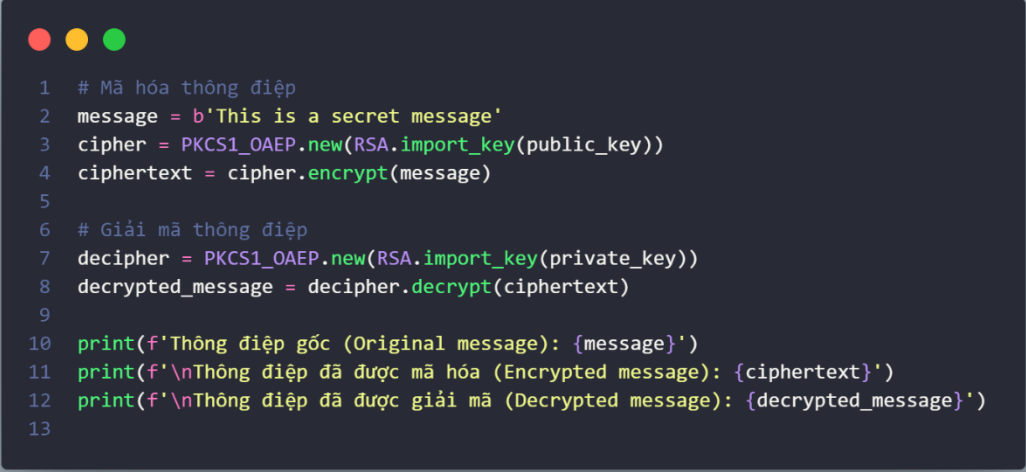
```

Hình 9.1 Code hiện thực quy trình tạo khóa công khai và khóa bí mật

- *Crypto.PublicKey.RSA*: Cung cấp các phương pháp để tạo và quản lý các khóa RSA.

- *RSA.generate(bits)*: Tạo một cặp khóa RSA mới với độ dài là bits bit. Trong đoạn code, độ dài này là 1024 bit.
- *key.export_key()*: Xuất khóa dưới dạng chuỗi byte. Có thể xuất khóa ở dạng PEM hoặc DER.
- *key.publickey()*: Truy xuất khóa công khai từ cặp khóa RSA.

9.3 Kiểm thử và xác minh



```

1  # Mã hóa thông điệp
2  message = b'This is a secret message'
3  cipher = PKCS1_OAEP.new(RSA.import_key(public_key))
4  ciphertext = cipher.encrypt(message)
5
6  # Giải mã thông điệp
7  decipher = PKCS1_OAEP.new(RSA.import_key(private_key))
8  decrypted_message = decipher.decrypt(ciphertext)
9
10 print(f'Thông điệp gốc (Original message): {message}')
11 print(f'\nThông điệp đã được mã hóa (Encrypted message): {ciphertext}')
12 print(f'\nThông điệp đã được giải mã (Decrypted message): {decrypted_message}')
13

```

Hình 9.2 Kiểm thử & mã hóa và giải mã thông điệp

- *Crypto.Cipher.PKCS1_OAEP*: Cung cấp một lớp để mã hóa và giải mã sử dụng RSA với padding OAEP (Optimal Asymmetric Encryption Padding).
 - *PKCS1_OAEP.new(key)*: Tạo một đối tượng mã hóa/giải mã OAEP với khóa RSA được cung cấp.
 - *cipher.encrypt(message)*: Mã hóa thông điệp bằng OAEP và khóa RSA công khai.
 - *cipher.decrypt(ciphertext)*: Giải mã thông điệp đã mã hóa bằng OAEP và khóa RSA riêng.
- Quy trình mã hóa và giải mã:
 - Mã hóa thông điệp: *PKCS1_OAEP.new(RSA.import_key(public_key))* tạo ra một đối tượng mã hóa sử dụng khóa công khai.
cipher.encrypt(message) mã hóa thông điệp.
 - Giải mã thông điệp:
PKCS1_OAEP.new(RSA.import_key(private_key)) tạo ra một đối tượng giải mã sử dụng khóa riêng. *decipher.decrypt(ciphertext)* giải mã thông điệp đã mã hóa.

- Kết quả:

```
PS C:\Study\HK2 - 2023\CTRR\Final> python 52200205_52200051_52200033_9.py
Thông điệp gốc (Original message): b'This is a secret message'

Thông điệp đã được mã hóa (Encrypted message): b'\x84\x80\xef\x0f"sn.B@F\x04\x05\xba\x00\x97\x00+\x95\x90\x01D\xba\x04\x10\x1d\x02T\x0e\x1b\xf7\x89c\xbdU\x
e8\xaf\x1e\x09\x9e\xcd\xdb\xba\x89!\xf8#:\xa2\x9d\xa25#>\xa7W\xb46\x91*\#\x9b8\heck\x8f\x8d\xa4*?\x0b\x04`\xf2\x17:\x94`S\xfb;\xf4\xcb\x82\xeb\x9ac\xbf\xa5\x1a
\xa5\x80\xa6;\x04\xdb\x04\x04\xa77\x0b\x16\x9f]g4\x03\xdc\x85\xab\xef\xa7v\x920$^\x1c\x0e\xdd'

Thông điệp đã được giải mã (Decrypted message): b'This is a secret message'
```

Hình 9.3 Kết quả quy trình mã hóa & giải mã

9.3.2 Xác minh

```
1 # Xác minh
2 if message == decrypted_message:
3     print("\nMã hóa thành công: Thông điệp được giải mã khớp với thông điệp gốc.")
4 else:
5     print("\nMã hóa thất bại: Thông điệp được giải mã không khớp với thông điệp gốc.")
6
```

Hình 9.4 Code xác minh lại tính đúng đắn của thuật toán

- So sánh thông điệp gốc với thông điệp đã giải mã để đảm bảo rằng quá trình giải mã đã thành công và trả về thông điệp gốc bằng câu lệnh:

if message == decrypted_message.

- Kết quả:

Mã hóa thành công: Thông điệp được giải mã khớp với thông điệp gốc.

Hình 9.5 Kết quả xác minh quy trình giải mã

9.4 Phân tích hiệu quả và bảo mật

9.4.1 Hiệu quả

- **Sinh khóa:** Quá trình sinh khóa RSA yêu cầu việc sinh hai số nguyên tố lớn và tính toán các giá trị n , $\Phi(n)$, e , d . Điều này có thể tốn nhiều thời gian và tài nguyên tính toán, đặc biệt khi sử dụng các khóa lớn (1024 bit hoặc lớn hơn). Việc sinh số nguyên tố lớn có thể mất nhiều thời gian do việc kiểm tra tính nguyên tố.
- **Mã hóa:** Mã hóa sử dụng phép toán mũ modulo, cụ thể là

$c \equiv m^e \pmod n$. Phép toán này có độ phức tạp thời gian là $O((\log n)^3)$ với n là độ dài của khóa. Quá trình mã hóa tương đối nhanh chóng so với quá trình giải mã.

- **Giải mã:** Giải mã sử dụng phép toán mũ modulo với khóa bí mật d , cụ thể là $m \equiv c^d \pmod n$. Do d thường rất lớn, quá trình giải mã có thể tốn nhiều thời gian hơn so với mã hóa. Độ phức tạp thời gian là tương tự như mã hóa, nhưng do giá trị d lớn hơn e , giải mã thường chậm hơn mã hóa.

9.4.2 Bảo mật

- **Dựa trên khó khăn của việc phân tích số:** Bảo mật của RSA dựa vào độ khó của việc phân tích số n thành các thừa số nguyên tố p và q . Nếu các số nguyên tố p và q được chọn đủ lớn, việc phân tích n trở nên cực kỳ khó khăn và tốn thời gian.
- **Khóa công khai và khóa bí mật:** Khóa công khai e và n có thể được chia sẻ rộng rãi mà không ảnh hưởng đến bảo mật của hệ thống. Khóa bí mật d phải được giữ kín. Nếu d bị lộ, toàn bộ hệ thống bảo mật sẽ bị phá vỡ.
- **Phụ thuộc vào việc chọn khóa:** Việc chọn các giá trị e và d đúng cách rất quan trọng. Ví dụ, nếu e quá nhỏ hoặc không được chọn một cách cẩn thận, nó có thể dẫn đến các lỗ hổng bảo mật.

9.5 Thảo luận về các mối đe dọa bảo mật và hạn chế của hệ mật RSA

9.5.1 Các mối đe dọa bảo mật

- **Exponent công khai nhỏ:** Chọn e quá nhỏ (ví dụ, $e = 3$) có thể dẫn đến các cuộc tấn công văn bản được chọn (chosen plaintext attack). Các lược đồ padding như OAEP có thể giúp ngăn chặn vấn đề này.
- **Tấn công thời gian:** Các cuộc tấn công thời gian (timing attacks) có thể khai thác sự thay đổi thời gian tính toán để lấy thông tin về khóa bí mật d . Điều này có thể xảy ra nếu việc tính toán các phép toán mũ modulo không được thực hiện trong thời gian hằng.

- **Tấn công kênh bên:** Các cuộc tấn công kênh bên (side-channel attacks) sử dụng thông tin từ việc tiêu thụ điện năng, bức xạ điện từ, hoặc các đặc điểm khác của hệ thống để lấy thông tin về khóa bí mật.
- **Máy tính lượng tử:** Các thuật toán lượng tử như thuật toán Shor có thể phân tích số lớn một cách hiệu quả, đe dọa bảo mật của RSA. Khi máy tính lượng tử đủ mạnh được phát triển, RSA có thể trở nên không an toàn.

9.5.2 Hạn chế

- **Kích thước khóa:** Để đảm bảo an toàn, RSA cần sử dụng các khóa có kích thước lớn (1024 bit hoặc lớn hơn), làm tăng độ phức tạp và tài nguyên tính toán cần thiết.
- **Hiệu suất:** Mã hóa và đặc biệt là giải mã RSA với các khóa lớn có thể rất tốn tài nguyên và không phù hợp cho các hệ thống yêu cầu hiệu suất cao hoặc thời gian thực.

9.6 Khuyến nghị cải thiện

- **Tăng kích thước khóa:** Sử dụng các khóa lớn hơn (tối thiểu 1024 bit, tốt hơn là 3072 bit hoặc lớn hơn) để tăng cường bảo mật. Điều này giúp đảm bảo an toàn trước các cuộc tấn công brute-force và phân tích số.
- **Sử dụng lược đồ padding:** Triển khai các lược đồ padding như OAEP (Optimal Asymmetric Encryption Padding) để ngăn chặn các cuộc tấn công văn bản được chọn và các lỗ hổng bảo mật liên quan đến việc chọn e nhỏ.
- **Thuật toán thời gian hằng:** Sử dụng và triển khai các thuật toán với thời gian hằng để giảm nguy cơ tấn công thời gian và các cuộc tấn công kênh bên.
- **Bảo vệ kênh bên:** Áp dụng các biện pháp bảo vệ chống lại các cuộc tấn công kênh bên như che giấu tín hiệu (signal masking), cân bằng năng lượng (power balancing), và bảo vệ vật lý.
- **Nghiên cứu mật mã hậu lượng tử:** Bắt đầu nghiên cứu và triển khai các hệ mật hậu lượng tử (post-quantum cryptography) để đảm bảo bảo mật trong tương lai khi máy tính lượng tử đủ mạnh được phát triển.

TÀI LIỆU THAM KHẢO

[1] S. S. Epp, Discrete Mathematics with Applications, Fifth Edition ed., Boston:

Cengage, Boston, MA, 2020.

[2] Khoa Học & Chúng Ta, Director, *Mật Mã RSA: Khi Toán học bảo vệ túi tiền của bạn.*

[Film]. 2022.

[3] I. Wikimedia Foundation, "Wikipedia," 3 5 2024. [Online]. Available:

[https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](https://en.wikipedia.org/wiki/RSA_(cryptosystem)).

[4] I. Wikimedia Foundation, "Wikipedia," 4 5 2024. [Online]. Available:

https://en.wikipedia.org/wiki/Eulerian_path#Hierholzer.27s_algorithm.

[5] TS. Nguyễn Thị Huỳnh Trâm, "DHTDT_CTRR_Lecture_02.pdf," HCHC, 2024.

[6] TS. Nguyễn Thị Huỳnh Trâm, "DHTDT_CTRR_Lecture_01.pdf," HCHC, 2024.

[7] TS. Nguyễn Thị Huỳnh Trâm, "DHTDT_CTRR_Lecture_03.pdf," HCHC, 2024.

[8] TS. Nguyễn Thị Huỳnh Trâm, "DHTDT_CTRR_Lecture_03.pdf," HCHC, 2024.