**HCMC UNIVERSITY OF TECHNOLOGY AND EDUCATION**

**FACULTY OF INFORMATION TECHNOLOGY**

Logo

Description automatically generated

**FINAL TERM PROJECT**

**Course name: DISCRETE MATHEMATICS & GRAPH THEORY**

**BUIDING A FAMILY TREE USING TREE STRUCTURES AND DETERMINING WHETHER TWO BLOOD RELATIVES CAN MARRY**

**Lecturer name**: Assoc. Prof. Hoang Van Dung

**List of members**

|  |  |  |
| --- | --- | --- |
| **Student ID** | **Student name** | **Contribution (%)** |
| 24110088 | Ngô Trần Ngân Đô | 100% |
| 24110093 | Doãn Huy | 100% |
| 24110121 | Lương Đình Thư Phong | 100% |

*Ho Chi Minh City, 9/2025*

*The content of project report includes some major parts as follows:*

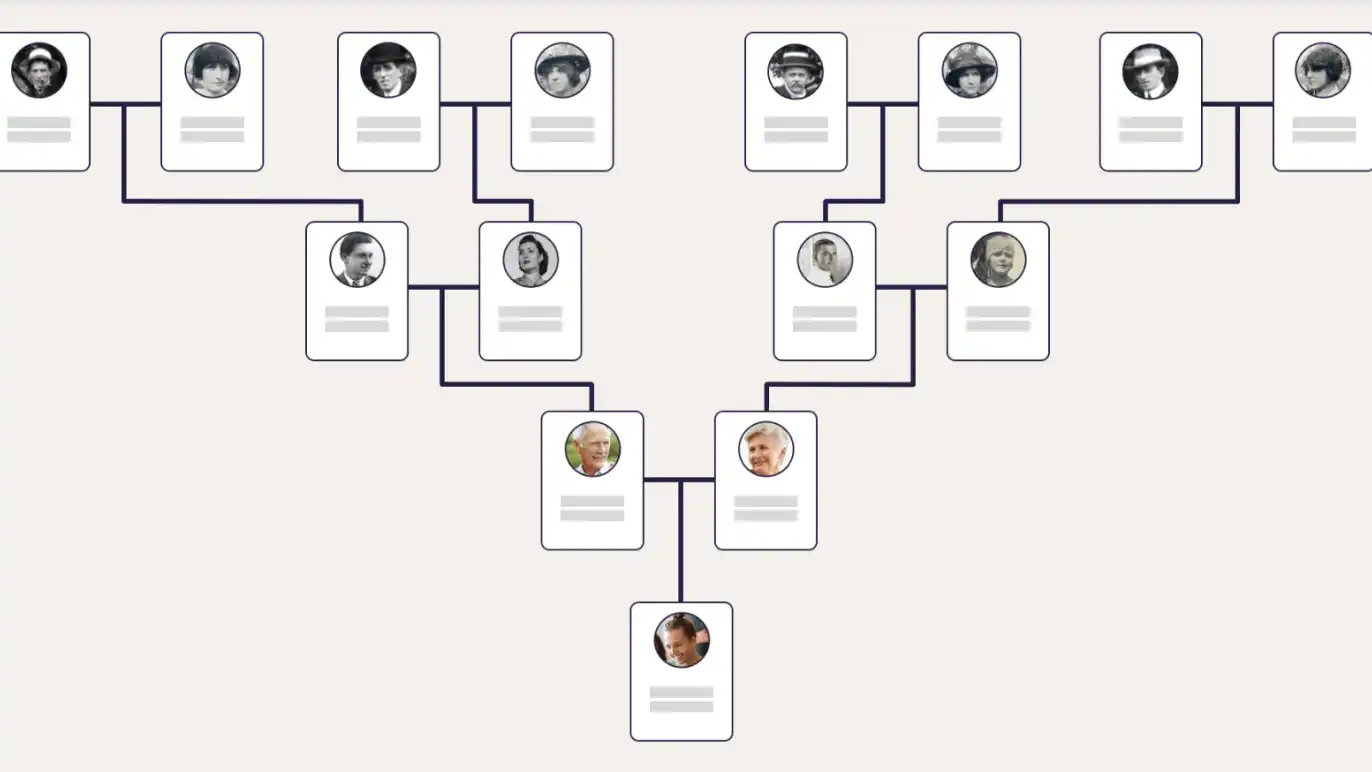
**Index**

1. Abbreviation List
2. Figure List
3. Table List
4. Part 1: Project Description
5. Part 2: Background Knowledge
6. Part 3: System Design
7. Part 4: Implementation, Test Cases, Results, and Discussion
8. Part 5: Conclusions
9. Part 6 : References

**Abbereviation List**

* DFS – Depth-First Search
* BFS – Breath-First Search
* LCA – Lowest Common Ancestor
* STL – Standard Template Library

**Figure List**



This image is an example of how a **tree structure** can be used to represent a **family tree**. In computer science, a tree is a hierarchical data structure where each node represents an entity (in this case, a person), and edges represent relationships (such as parent-child).

Image about Input

Image about Output

Image about problem while setting algorithm

**Table List**

**Part 1 : Project Description**

In this project, we develop a system that builds and manages a **family tree** using a tree data structure. The program applies **depth-first search (DFS)** and **breadth-first search (BFS)** algorithms to traverse and analyze family relationships. By storing information about individuals and their connections, the system can determine the degree of kinship between two people.

Based on this analysis, the program provides notifications to the user about whether two individuals are allowed to marry, according to their blood relationship. The project demonstrates the practical application of fundamental concepts in data structures**,** tree traversal algorithms**,…**

**Part 2 : Background Knowledge**

* **DFS**  
  DFS is used to traverse the family tree by exploring as far as possible along each branch before backtracking. In this project, DFS helps list descendants, find all ancestors of a person, or check if two nodes are connected.
* **BFS**  
  BFS explores the tree level by level, making it useful for identifying generational distances. For example, BFS can quickly determine how many generations separate two individuals.
* **LCA**  
  The LCA of two nodes is the deepest ancestor they share. In a family tree, the LCA is crucial for determining the blood relationship between two people. For instance, if the LCA is a parent, the two nodes are siblings; if it is a grandparent, they are cousins. This information is applied to marriage eligibility rules.

**Part 3 : implementation algorithm**Overall architecture :

* Family represented as a tree structure.
* Each person is a node with atrributes (name, gender, parents, children).

**Part 4 : Implementation, Testing, Results, and Discussions**

Pseudo code for using tree structure with BFS, DFS algorithm, C++ language :

// function buildTree():

// Khởi tạo: ai không có cha/mẹ → là gốc, depth = 0

tạo queue Q

với mỗi người trong family:

nếu p.father\_id == -1 và p.mother\_id == -1:

thêm id vào Q

depth[id] = 0

p.parent = -1

// BFS để gán depth và parent cho con cháu

trong khi Q không rỗng:

lấy u ra khỏi Q

với mỗi người p trong family:

nếu p.father\_id == u HOẶC p.mother\_id == u:

nếu p chưa có depth:

depth[p.id] = depth[u] + 1

p.parent = u // chọn u (cha hoặc mẹ) làm parent

thêm p.id vào Q

// ========== LẤY ĐƯỜNG ĐI TỪ NÚT ĐẾN GỐC ==========

function getPath(id):

tạo mảng path = []

trong khi id != -1:

thêm id vào path

id = family[id].parent // đi lên cha

đảo ngược path (để gốc ở đầu)

trả về path

// ========== TÍNH KHOẢNG CÁCH HUYẾT THỐNG ==========

function getBloodDistance(id1, id2):

nếu id1 hoặc id2 không tồn tại → trả về -1

path1 = getPath(id1) // [gốc, ..., ông, cha, id1]

path2 = getPath(id2) // [gốc, ..., ông, mẹ, id2]

lca = -1

// Duyệt từ gốc xuống, tìm điểm khác nhau đầu tiên

i = 0

trong khi i < min(len(path1), len(path2)) và path1[i] == path2[i]:

lca = path1[i]

i = i + 1

nếu lca == -1 → không có tổ tiên chung → trả về -1

depth1 = độ dài path1 - 1 // vì path[0] là gốc, depth=0

depth2 = độ dài path2 - 1

depthLCA = vị trí của lca trong path1 (hoặc path2)

distance = depth1 + depth2 - 2 \* depthLCA

trả về distance

// ========== HÀM MAIN ==========

main():

readCSV("family.csv")

buildTree()

in "Nhập ID hai người để kiểm tra:"

lặp:

nhập id1, id2

distance = getBloodDistance(id1, id2)

nếu distance == -1:

in "Không cùng huyết thống → ĐƯỢC PHÉP KẾT HÔN"

ngược lại:

in "Khoảng cách huyết thống: ", distance

nếu distance <= 4:

in "KHÔNG ĐƯỢC PHÉP KẾT HÔN (trong vòng 3 đời)"//

**Part 5 : Conclusions :**

This project highlights the practical use of **tree structures in C++** to represent and analyze family relationships. By modeling individuals as nodes and connections as parent–child links, the system can trace ancestry and determine blood relations efficiently. The main feature—checking whether two people are eligible to marry—demonstrates how computational methods can be applied to real-world cultural and social rules.

Although the current design is simple and relies on basic C++ structures, it successfully shows the strength of algorithmic approaches in handling hierarchical data. The project also emphasizes how family trees, which are naturally complex, can be simplified through data structure representation.

In the future, the system could be extended with larger datasets, user-friendly interfaces, or database integration to make it more practical. Overall, this work proves that **fundamental data structures** provide a strong foundation for solving problems beyond computer science, bridging technology with everyday social contexts.

**Part 6 : Conferences :**

1. Skiena, S. (2008). *The Algorithm Design Manual*. Springer.
2. Morgan, L. H. (1871). *Systems of Consanguinity and Affinity of the Human Family*. Smithsonian Institution.
3. Stone, L. (2001). *Kinship and Gender: An Introduction*. Westview Press.
4. Parish, S. (2011). *Tree Data Structures and Applications in Computing*. Cambridge Scholars Publishing.

Tóm tắt bằng bản Việt

**1. Mở đầu**

Đề tài tập trung xây dựng hệ thống quản lý gia phả từ dữ liệu Excel (xuất ra CSV) và kiểm tra quan hệ huyết thống giữa hai cá nhân. Bài toán đặt ra là xác định xem hai người có chung tổ tiên trong vòng 3 đời (cha, ông bà, cụ) hay không. Nếu có → không được kết hôn; nếu không → được phép kết hôn.

**Mục tiêu:**

Quản lý thông tin cá nhân bằng mô hình dữ liệu rõ ràng.

Áp dụng thuật toán BFS giới hạn 3 tầng để kiểm tra tổ tiên chung.

Xuất kết quả chi tiết, có giải thích.

**2. Cơ sở lý thuyết**

Cấu trúc cây/đồ thị có hướng không chu trình (DAG): biểu diễn quan hệ cha–mẹ–con.

Thuật toán BFS: duyệt theo tầng, trong phạm vi 3 tầng để tìm tổ tiên.

Định dạng CSV: người dùng nhập dữ liệu trong Excel, sau đó “Save As → CSV (UTF-8)” để chương trình đọc.

Luật Hôn nhân và Gia đình (2014): cấm kết hôn trong phạm vi 3 đời.

**3. Phân tích, thiết kế giải pháp**

Mô hình dữ liệu: mỗi cá nhân có ID, họ tên, giới tính, ngày sinh, cha, mẹ, quê quán, ghi chú.

Quy trình xử lý:

Người dùng nhập dữ liệu vào Excel.

Xuất ra file CSV.

Chương trình đọc CSV → xây dựng quan hệ cha mẹ.

Với hai cá nhân A và B: chạy BFS từ mỗi người lên 3 đời.

Nếu có tổ tiên chung → không được kết hôn.

Nếu không trùng → được kết hôn.

Sơ đồ khối:

Excel → CSV → Chương trình đọc → BFS tổ tiên ≤ 3 đời → So sánh → Kết luận

**4. Thực nghiệm và kết quả dự kiến**

Dữ liệu thử nghiệm: 15–20 cá nhân có nhiều quan hệ khác nhau (anh chị em, cô cháu, họ xa).

Kết quả dự kiến:

A và B có cùng cha/mẹ → không được kết hôn.

A và B có chung ông bà/cụ → không được kết hôn.

A và B khác cây hoặc tổ tiên chung > 3 đời → được kết hôn.

Ví dụ đầu ra:

So sánh: Con D (1990) và Con E (1992)

Kết quả: Không được kết hôn

Lý do: Có chung cha mẹ → anh chị em ruột (1 đời).