TRƯỜNG ĐH KHOA HỌC TỰ NHIÊN – ĐH QUỐC GIA HÀ NỘI

KHOA TOÁN – CƠ – TIN HỌC

**----- -----**



BÁO CÁO

ĐỀ TÀI: TRUY TÌM KHO BÁU

*Giảng viên:* Nguyễn Thị Hồng Minh

Trần Bá Tuấn

*Sinh viên thực hiên:*

22001604 – Đỗ Chí Kiên

22001623 – Đỗ Quốc Minh Nghĩa

22001665 – Cao Sỹ Nguyên Vũ

22001666 – Lê Minh Vương

HÀ NỘI – 2023

**Mục lục**

[I - Giới thiệu chủ đề 4](#_Toc153100900)

[**1. Đặt vấn đề** 4](#_Toc153100901)

[**2. Mục Tiêu Đề tài** 4](#_Toc153100902)

[**3. Ý Nghĩa Của Đề tài** 4](#_Toc153100903)

[**4. Cấu Trúc Báo Cáo** 4](#_Toc153100904)

[II - Giới thiệu về các khái niệm cơ bản về cấu trúc dữ liệu và thuật toán 4](#_Toc153100905)

[**1. Mảng hai chiều** 4](#_Toc153100906)

[**2. Priority Queue** 5](#_Toc153100907)

[**3. Stack** 6](#_Toc153100908)

[**4. Hash table** 7](#_Toc153100909)

[**5. Union Find** 8](#_Toc153100910)

[**6. Thuật toán A\*** 10](#_Toc153100911)

[III - Code ứng dụng 18](#_Toc153100912)

[IV - Đưa ra kết luận 24](#_Toc153100913)

[V - Giới thiệu về game Truy tìm bảo vật 25](#_Toc153100914)

[**1. Giới thiệu chung** 25](#_Toc153100915)

[**2. Bối cảnh và quy tắc chơi** 25](#_Toc153100916)

[**2.1. Bối cảnh:** 25](#_Toc153100917)

[**2.2 Quy tắc trò chơi** 29](#_Toc153100918)

**Mục lục ảnh**

[Hình 2.1: Ảnh minh hoạ mảng 2 chiều 5](#_Toc154154296)

[HÌnh 2.2: Ảnh minh hoạ Priority Queue 6](#_Toc154154297)

[Hình 2.3: Ảnh minh hoạ Stack 7](#_Toc154154298)

[Hình 2.4: Ảnh minh hoạ HashTable 8](#_Toc154154299)

[Hình 2.5: API của Union Find 9](#_Toc154154300)

[Hình 2.6: Ví dụ về Union Find 9](#_Toc154154301)

[Hình 2.7: Các cách để khai triển Union Find 10](#_Toc154154302)

[Hình 2.9: Ảnh minh hoạ Robot tìm đường đi bằng thuật toán A\* 11](#_Toc154154303)

[Hình 2.10: Mô tả cách hoạt động của thuật toán A\* 12](#_Toc154154304)

[Hình 2.11: Mô tả cách hoạt động của hàm Heuristic 13](#_Toc154154305)

[Hình 2.12: Sử dụng A\* tìm đường đi 15](#_Toc154154306)

[Hình 2.13: Ví dụ cách tính của A\* 15](#_Toc154154307)

[Video 2.15: So sánh thời gian chạy các thuật toán tìm đường đi 18](#_Toc154154308)

[Hình 3.1: Mô tả code của A\* 19](#_Toc154154309)

[Hình 3.2: Mô tả code của A\* 20](#_Toc154154310)

[Hình 3.3: Mô tả code của A\* 21](#_Toc154154311)

[Hình 3.4: Mô tả code của Union Find 22](#_Toc154154312)

[Hình 3.5: Mô tả code của Union Find 23](#_Toc154154313)

[Hình 3.6: Mô tả code của Union Find 23](#_Toc154154314)

[Hình 3.7: Mô tả code của Union Find 24](#_Toc154154315)

[Hình 5.1: Khu vực 1 26](#_Toc154154316)

[Hình 5.2: Khu vực 2 27](#_Toc154154317)

[Hình 5.3: Khu vực 3 28](#_Toc154154318)

[Hình 5.4: Khu vực mở đầu 29](#_Toc154154319)

[Hình 5.5: Game kết thúc 30](#_Toc154154320)

[Hình 5.6: Đạt được vật phẩm 31](#_Toc154154321)

[Hình 5.7: Con đường khai sáng 32](#_Toc154154322)

[Hình 5.8: Kho báu cuối cùng 33](#_Toc154154323)

# **I - Giới thiệu chủ đề**

## **1. Đặt vấn đề**

Trong thế giới ngày nay, với sự phát triển mạnh mẽ của công nghệ thông tin, nhu cầu trong việc tối ưu hóa các quy trình là mối quan tâm hàng đầu. Một trong những thách thức quan trọng nhất là vấn đề về tìm đường đi ngắn nhất giữa hai điểm trên một ma trận. Điều này không chỉ quan trọng trong lĩnh vực địa lý, nơi chúng ta cần tìm đường đi ngắn nhất từ một địa điểm này đến địa điểm khác, mà còn trong nhiều lĩnh vực khác như quản lý chuỗi cung ứng, mạng lưới giao thông, và nhiều ứng dụng khác.

## **2. Mục Tiêu Đề tài**

Mục tiêu của nghiên cứu này là tìm hiểu, phân tích, và phát triển các thuật toán tìm đường đi ngắn nhất trên ma trận. Chúng ta sẽ xem xét các phương pháp hiện đại nhất để giải quyết vấn đề này, đồng thời đề xuất và thử nghiệm các cải tiến để cải thiện hiệu suất và độ chính xác của các thuật toán.

## **3. Ý Nghĩa Của Đề tài**

Việc nghiên cứu về tìm đường đi ngắn nhất trên ma trận không chỉ mang lại những kết quả quan trọng trong việc hiểu sâu về cơ sở lý thuyết mà còn có ứng dụng thực tế mạnh mẽ. Các giải pháp hiệu quả có thể giúp tối ưu hóa quá trình lập kế hoạch và điều phối trong nhiều lĩnh vực, từ giao thông vận tải đến quản lý nguồn lực.

## **4. Cấu Trúc Báo Cáo**

Báo cáo này sẽ được tổ chức thành các chương theo trình tự logic để trình bày quá trình báo cáo và kết quả đạt được. Phần II sẽ giới thiệu về các khái niệm cơ bản liên quan đến vấn đề tìm đường đi ngắn nhất và các thuật toán hiện đại, các cấu trúc dữ liệu. Phần III sẽ trình bày phương pháp nghiên cứu và thiết kế thực nghiệm. Phần IV sẽ trình bày ý nghĩa của đề tài

# **II - Giới thiệu về các khái niệm cơ bản về cấu trúc dữ liệu và thuật toán**

## **1. Mảng hai chiều**

- Cấu trúc mảng 2 chiều, hay còn được gọi là ma trận, là một trong những khái niệm cơ bản và quan trọng trong lập trình và toán học. Nó cho phép tổ chức dữ liệu dưới dạng một bảng hai chiều, với hàng và cột, giúp mô phỏng hiệu quả các tình huống thực tế đa chiều.

- Mảng 2 chiều có các đặc điểm chính sau:

* **Hàng và Cột:** Mảng 2 chiều có thể được hình dung như một bảng có hàng và cột, nơi mỗi phần tử được xác định bằng một cặp chỉ số (i, j), với i là chỉ số của hàng và j là chỉ số của cột.
* **Kích Thước:** Mảng 2 chiều có kích thước được xác định bởi số lượng hàng và số lượng cột. Được biểu diễn dưới dạng m x n, với m là số hàng và n là số cột.

A table with numbers and symbols

Description automatically generated

**Hình 2.1: Ảnh minh hoạ mảng 2 chiều**

## **2. Priority Queue**

- Priority Queue là một cấu trúc dữ liệu mà các phần tử được quản lý sẽ có “độ ưu tiên” khác nhau gắn với từng phần tử. Phần tử có thứ tự ưu tiên cao hơn trong Priority Queue sẽ được xếp lên trước và truy vấn trước. Đơn giản hơn, Priority Queue là một cấu trúc cho phép nó tự động sắp xếp các phần tử của nó.

- Ưu Điểm:

* Hiệu Quả: Giúp ưu tiên xử lý các công việc quan trọng hơn trước, tăng hiệu suất của các thuật toán và hệ thống.
* Ứng Dụng Đa Dạng: Có thể được áp dụng trong nhiều lĩnh vực khác nhau, từ lập trình cơ bản đến các ứng dụng phức tạp.

A diagram of a diagram

Description automatically generated

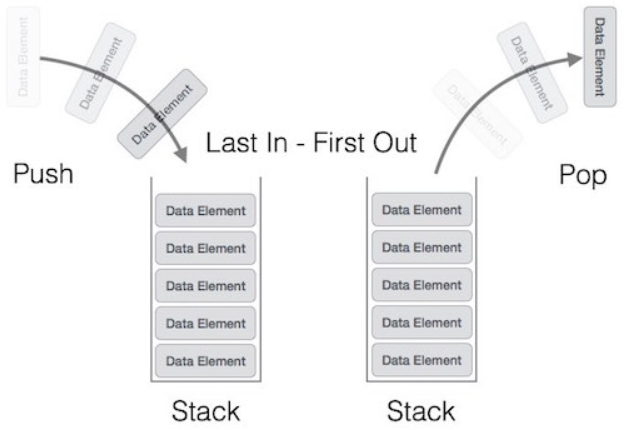
**HÌnh 2.2: Ảnh minh hoạ Priority Queue**

## **3. Stack**

- Stack (ngăn xếp) hoạt động theo nguyên tắc "Last In, First Out" (LIFO), có nghĩa là phần tử cuối cùng được thêm vào là phần tử đầu tiên được lấy ra. Stack được sử dụng rộng rãi trong nhiều ứng dụng và thuật toán với mục đích lưu trữ và quản lý dữ liệu một cách có tổ chức.

- Ưu Điểm:

* Đơn Giản và Hiệu Quả: Thực hiện các thao tác Push và Pop nhanh chóng, làm cho stack trở thành một cấu trúc dữ liệu đơn giản nhưng mạnh mẽ.
* Dễ Hiểu và Triển Khai: Thích hợp cho nhiều bài toán và dễ triển khai trong mã nguồn.



**Hình 2.3: Ảnh minh hoạ Stack**

## **4. Hash table**

- Hashtable là một mảng của danh sách. Mỗi danh sách được gọi là một nhóm chứa các cặp khóa/giá trị trong đó. Nó sử dụng phương thức hashCode() để xác định nhóm nào sẽ được gán cho tổ hợp khóa/giá trị. Nói chung, mã băm sử dụng số nguyên không âm bằng với các Đối tượng bằng nhau nhưng có thể bằng hoặc không bằng đối với các Đối tượng không bằng nhau. Hàm equals() được bảng băm sử dụng để phát hiện xem hai mục có bằng nhau hay không.

A diagram of a company

Description automatically generated

**Hình 2.4: Ảnh minh hoạ HashTable**

## **5. Union Find**

- Cấu trúc dữ liệu lưu trữ tập hợp con không chồng chéo hoặc rời rạc của các phần tử được gọi là cấu trúc dữ liệu tập hợp rời rạc. Cấu trúc dữ liệu tập hợp rời rạc hỗ trợ các chức năng sau:

* Thêm tập hợp mới vào tập hợp rời rạc.
* Hợp nhất các tập hợp rời rạc thành một tập hợp rời rạc duy nhất bằng cách sử dụng hàm Union .
* Tìm đại diện của một tập hợp rời rạc bằng hàm Find.
* Kiểm tra xem hai tập hợp có rời nhau hay không.

- Union-find (API)

A screenshot of a computer code

Description automatically generated

**Hình 2.5: API của Union Find**

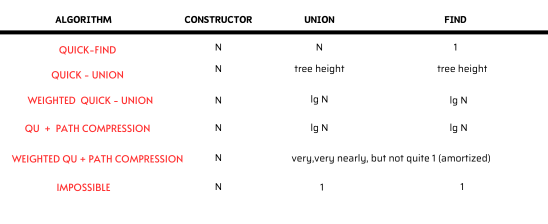
- Dưới đây là ví dụ về các API của Union Find

A screenshot of a computer

Description automatically generated

**Hình 2.6: Ví dụ về Union Find**

- Các cách triển khai cấu trúc Union Find



**Hình 2.7: Các cách để khai triển Union Find**

## **6. Thuật toán A\***

- A\* là một thuật toán máy tính được sử dụng rộng rãi trong việc tìm đường và duyệt đồ thị. Thuật toán tạo một đường đi khả thi một cách hiệu quả nhất giữa nhiều nút hoặc điểm trên đồ thị . Trên bản đồ có nhiều chướng ngại vật, việc tìm đường đi từ điểm A đến B có thể khó khăn. Ví dụ: một robot nếu không khả thi đi những hướng khác sẽ tiếp tục đi cho đến khi gặp chướng ngại vật, như trong ví dụ tìm đường ở bên dưới

A diagram of a path

Description automatically generated

**Hình 2.8: Ảnh minh hoạ Robot tìm đường đi bằng các thuật toán không phải A\***

- Tuy nhiên, thuật toán A\* giới thiệu phương pháp phỏng đoán vào thuật toán tìm kiếm biểu đồ thông thường, về cơ bản là lập kế hoạch trước ở mỗi bước để đưa ra quyết định tối ưu hơn. Với A\* robot sẽ thay vào đó tìm đường đi theo cách tương tự như sơ đồ bên dưới.

A virtual obstacle diagram with text

Description automatically generated with medium confidence

**Hình 2.9: Ảnh minh hoạ Robot tìm đường đi bằng thuật toán A\***

- A\* là phần mở rộng của thuật toán Dijkstra với một số đặc điểm của tìm kiếm theo chiều rộng (BFS).

- Giống như Dijkstra, A\* hoạt động bằng cách tạo cây đường dẫn có chi phí thấp nhất từ ​​nút bắt đầu đến nút đích. Điều làm cho A\* khác biệt và tốt hơn đối với nhiều tìm kiếm là đối với mỗi nút, A\* sử dụng hàm f(n) đưa ra ước tính về tổng chi phí của một đường dẫn sử dụng nút đó. Do đó, A\* là một hàm heuristic, khác với một thuật toán ở chỗ heuristic thiên về ước tính hơn và không nhất thiết phải được chứng minh là chính xác. A\* mở rộng các đường dẫn vốn đã ít tốn kém hơn bằng cách sử dụng hàm này:

  f(n) = g(n) + h(n)

- Trong đó:

* f(n): Tổng chi phí ước chừng để đến n
* g(n): Tổng chi phí ước chừng từ n đến node bắt đầu
* h(n): Tổng chi phí ước chừng từ n đến đích

A graph with a rectangle and a line

Description automatically generated with medium confidence

**Hình 2.10: Mô tả cách hoạt động của thuật toán A\***

- Trong lưới ở trên, thuật toán A\* bắt đầu ở đầu (nút màu đỏ) và xem xét tất cả các ô liền kề. Khi danh sách các ô liền kề đã được điền, nó sẽ lọc ra những ô không thể tiếp cận được (tường, chướng ngại vật, ngoài giới hạn). Sau đó, nó chọn ô có chi phí thấp nhất, đó là f(n) ước tính. Quá trình này được lặp lại đệ quy cho đến khi tìm được đường đi ngắn nhất tới đích (nút màu xanh). Việc tính toán f(n) được thực hiện thông qua phương pháp heuristic thường cho kết quả tốt.

- Việc tính h(n) có thể được thực hiện theo nhiều cách khác nhau:

* Khoảng cách Manhattan (được giải thích bên dưới) từ nút n đến mục tiêu thường được sử dụng. Đây là một heuristic tiêu chuẩn cho một lưới.
* Nếu h(n) = 0, A\* trở thành thuật toán Dijkstra, đảm bảo tìm được đường đi ngắn nhất.

- Hàm heuristic phải được chấp nhận, nghĩa là nó không bao giờ có thể đánh giá quá cao chi phí để đạt được mục tiêu. Cả khoảng cách Manhattan và h(n) = 0 đều được chấp nhận.

- Heuristic:

* Việc sử dụng phương pháp phỏng đoán tốt là điều quan trọng trong việc xác định hiệu suất của A\*. Giá trị của h(n) lý tưởng nhất sẽ bằng chi phí chính xác để đến đích. Tuy nhiên, điều này là không thể bởi vì chúng ta thậm chí còn không biết con đường chính xác. Tuy nhiên, chúng ta có thể chọn một phương pháp sẽ cho chúng ta giá trị chính xác trong một số trường hợp, chẳng hạn như khi di chuyển theo đường thẳng không có chướng ngại vật. Điều này sẽ dẫn đến hiệu suất hoàn hảo của A\* trong trường hợp như vậy.
* Ta muốn có thể chọn một hàm h(n) nhỏ hơn chi phí để đạt được mục tiêu. Điều này sẽ cho phép h hoạt động chính xác, nếu chúng ta chọn giá trị h lớn hơn sẽ dẫn đến hiệu suất nhanh hơn nhưng kém chính xác hơn. Vì vậy, thông thường chúng ta chọn h(n) nhỏ hơn chi phí thực.

A diagram of a grid with a line drawn on it

Description automatically generated with medium confidence

**Hình 2.11: Mô tả cách hoạt động của hàm Heuristic**

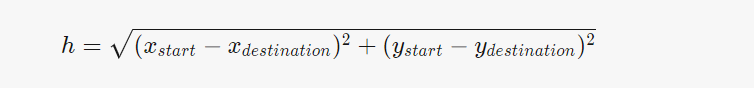
* Phương pháp tính h(n) này được gọi là phương pháp Manhattan vì nó được tính bằng cách tính tổng số ô vuông được di chuyển theo chiều ngang và chiều dọc để đến ô vuông mục tiêu từ ô vuông hiện tại. Chúng tôi bỏ qua chuyển động chéo và bất kỳ chướng ngại vật nào có thể cản trở.



* Phương pháp phỏng đoán này chính xác bất cứ khi nào đường đi của chúng ta đi theo một đường thẳng. Nghĩa là, A\* sẽ tìm các đường đi là sự kết hợp của các chuyển động thẳng. Đôi khi chúng ta có thể thích một con đường có xu hướng đi thẳng đến đích.

- The Euclidean Distance Heuristic

* Phương pháp phỏng đoán này chính xác hơn một chút so với phương pháp phỏng đoán ở Manhattan. Nếu chúng ta thử chạy cả hai cùng một lúc trên cùng một mê cung, công cụ tìm đường Euclide sẽ thiên về đường đi dọc theo một đường thẳng. Điều này chính xác hơn nhưng cũng chậm hơn vì phải khám phá một khu vực rộng hơn để tìm đường đi.



* Hạn chế chính của thuật toán  A ∗ và thực tế của bất kỳ tìm kiếm tốt nhất đầu tiên nào là yêu cầu về bộ nhớ của nó. Vì ít nhất toàn bộ danh sách mở phải được lưu nên thuật toán A\* bị giới hạn nghiêm trọng về không gian trong thực tế và không thực tế hơn thuật toán tìm kiếm đầu tiên tốt nhất trên các máy hiện tại.
* Sử dụng A\* để tìm đường đi ngắn nhất từ ô xanh đến ô vàng ở hình dưới

A grid of squares with different colored squares

Description automatically generated

**Hình 2.12: Sử dụng A\* tìm đường đi**

* Chúng ta hãy bắt đầu bằng cách chọn một heuristic có thể chấp nhận được.
* Trong trường hợp này, chúng ta có thể sử dụng phương pháp phỏng đoán Manhattan. Sau đó chúng ta tiến tới ô bắt đầu. Chúng tôi gọi nó là ô hiện tại và sau đó ta tiến hành xem xét tất cả các ô lân cận của nó và tính toán f,g,h cho từng f,g,h. Sau đó chúng ta chọn hàng xóm có chi phí thấp nhất. Đây là ô hiện tại mới và sau đó lặp lại quy trình trên. (điền vào hàng xóm và tính f,g và h rồi chọn mức thấp nhất). Làm điều này cho đến khi ô mục tiêu.

A white square with yellow and green squares

Description automatically generated

**Hình 2.13: Ví dụ cách tính của A\***

- Ứng dụng vào thực tể của thuật toán A\*

[](https://www.youtube.com/embed/CgW0HPHqFE8?feature=oembed)

**Video 2.14: Ứng dụng thực tế của A\***

- Dưới đây là một đoạn so sánh về các thời gian chạy giữa các thuật toán tìm đường đi ngắn nhất trong ma trận

[](https://www.youtube.com/embed/TqnlsbXbTUg?feature=oembed)

**Video 2.15: So sánh thời gian chạy các thuật toán tìm đường đi**

# **III - Code ứng dụng**

- Khởi tạo các đối tượng cơ bản cho thuật toán A\*

A screen shot of a computer

Description automatically generated

**Hình 3.1: Mô tả code của A\***

A screen shot of a computer program

Description automatically generated

**Hình 3.2: Mô tả code của A\***

* Kiểm tra điều kiện thỏa mãn và đánh dấu đường đi từ điểm bắt đầu đến điểm kết thúc và thêm các tọa độ đường đi vào một ArrayList.
* Dưới đây là cách thức hoạt động của thuật toán A\*:

A screen shot of a computer program

Description automatically generated

**Hình 3.3: Mô tả code của A\***

- Các API của Union Find trong ứng dụng game truy tìm báu vật

* Khởi tạo tất cả các phần tử của ma trận (là đường đi) trong HashMap parent và union các phần tử đường đi đó

A computer screen shot of a program code

Description automatically generated

**Hình 3.4: Mô tả code của Union Find**

* Hàm find() trả về cha của một phần tử (địa chỉ trong mảng)

A computer screen shot of code

Description automatically generated

**Hình 3.5: Mô tả code của Union Find**

* Hàm union() thay đổi cha của phần tử thứ 2 bằng cha của phần tử thứ nhất

A computer screen shot of numbers and symbols

Description automatically generated

**Hình 3.6: Mô tả code của Union Find**

* Kiểm tra 2 phần tử có liên kết hay không

A screen shot of a computer code

Description automatically generated

**Hình 3.7: Mô tả code của Union Find**

# **IV – Kết luận**

- Hiệu suất cao và độ chính xác: Thuật toán A\* kết hợp giữa chi phí thực tế đã đi và hàm ước lượng (heuristic) để đảm bảo hiệu suất cao trong việc tìm kiếm đường đi ngắn nhất. Sự cân nhắc giữa chi phí thực tế và ước lượng giúp giảm bớt độ phức tạp của bài toán tìm kiếm.

- Tính toán chi phí tối ưu và độ tối ưu hóa: A\* đảm bảo rằng nó sẽ tìm ra đường đi có chi phí thấp nhất từ điểm bắt đầu đến điểm đích. Đồng thời, nó không bỏ qua bất kỳ đường đi nào có chi phí thấp hơn, giúp đạt được độ tối ưu hóa cao.

- Áp dụng rộng rãi và linh hoạt: Khả năng điều chỉnh hàm ước lượng và chi phí thực tế làm cho A\* rất linh hoạt và có thể được áp dụng trong nhiều lĩnh vực khác nhau. Từ các ứng dụng trong trò chơi video đến quản lý tuyến đường giao thông, thuật toán này thích ứng với nhiều ngữ cảnh khác nhau.

- Chống được trường hợp "ùn tắc" (dead ends): A\* giảm đáng kể số lượng nút mở và tìm kiếm thông qua sự chọn lọc thông minh dựa trên hàm ước lượng. Điều này làm cho thuật toán khá hiệu quả và thích hợp cho các bài toán với không gian trạng thái lớn.

- Ứng dụng thực tế:

* Robot tự hành và dịch vụ giao hàng: A\* thường được tích hợp trong hệ thống điều khiển của robot tự hành và dịch vụ giao hàng để lập kế hoạch đường đi an toàn và hiệu quả trong môi trường thực tế.
* Quản lý tuyến đường giao thông thông minh: Trong lĩnh vực quản lý giao thông, A\* có thể được sử dụng để tối ưu hóa lịch trình và giảm ùn tắc giao thông trong các hệ thống đô thị.
* Mô phỏng và tư duy nhân tạo phức tạp: A\* được áp dụng rộng rãi trong mô phỏng hành vi và tư duy nhân tạo phức tạp, giúp mô phỏng và dự đoán đường đi của các thực thể trong môi trường ảo.
* Hệ thống hỗ trợ quyết định trong y tế: Trong y tế, A\* có thể được sử dụng để tối ưu hóa lộ trình của xe cứu thương hoặc tìm đường đi nhanh nhất đến các khu vực cần cấp cứu.
* Thuật toán A\* không chỉ là một công cụ mạnh mẽ cho tìm kiếm đường đi mà còn là một phương tiện linh hoạt và có thể điều chỉnh để đáp ứng các yêu cầu cụ thể của ứng dụng trong thế giới thực.

# **V - Giới thiệu về game Truy tìm bảo vật**

## **1. Giới thiệu chung**

Nhân dịp Noel 24/12 tựa game truy tìm báu vật với nhân vật chính là ông già noel vượt qua các chướng ngại vật thử thách nhặt các vật phẩm khác nhau để đi tới kho báu cuối cùng và phát người yêu cho các cháu.

## **2. Bối cảnh và quy tắc chơi**

### **2.1. Bối cảnh:**

**2.1.1. Khu vực 1:**

- Một khu rừng lạnh lẽo, quanh co, hiu quạnh được bao phủ bởi tuyết và vô vàn cây thông cùng với những hồ băng lạnh giá.

A video game screen with a cartoon character and a pond

Description automatically generated

**Hình 5.1: Khu vực 1**

**2.1.2. Khu vực 2:**

- Sau khi thoát khỏi khu rừng, chúng ta đến với căn phòng phúc lợi chứa hòm vật phẩm bí ẩn có thể giúp đỡ cho ông già noel trong khu vực tiếp theo nhưng chúng ta phải hoàn thành thử thách do căn phòng đặt ra

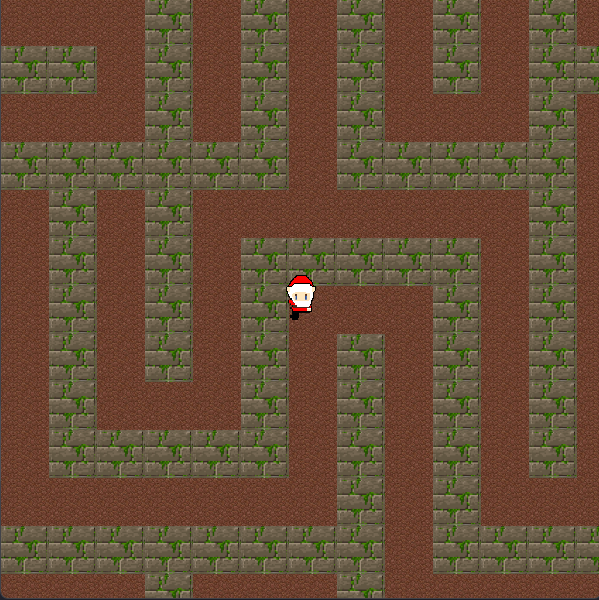
A video game screen with a red carpet and a door

Description automatically generated

**Hình 5.2: Khu vực 2**

**2.1.3 Khu vực 3:**

- Khu vực 3 sau khi thoát căn phòng phúc lợi bước đến cảnh cửa mở ra mê cung tối tăm, vô định chứa đựng vật phẩm dành cho người chiến thắng



**Hình 5.3: Khu vực 3**

### **2.2 Quy tắc trò chơi**

- Ông già noel bước vào hành trình tìm kho báu, khu vực mở đầu gồm 1 vật phẩm là chiếc rìu và 3 con đường chính và bị chặn bởi những chiếc cây.

- Ông già noel phải đưa ra quyết định của mình để chặt chiếc cây nào có thể mở ra con đường an toàn nhất.

A video game screen with trees and plants

Description automatically generated

**Hình 5.4: Khu vực mở đầu**

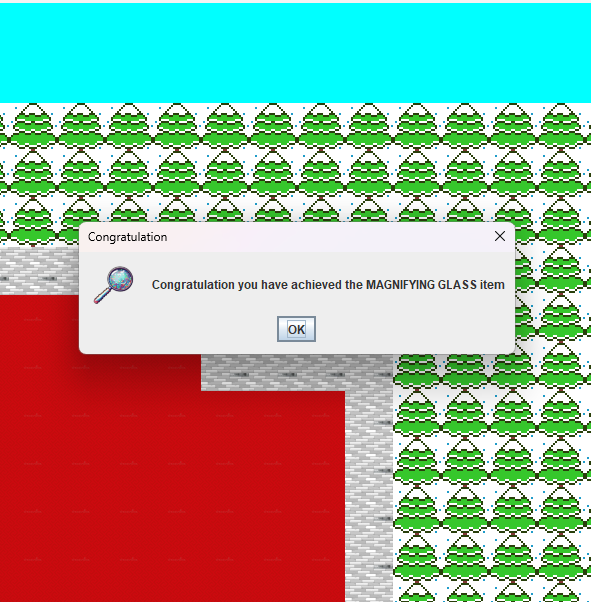
- Nếu chặt nhầm cây mà không mở ra con đường khả thi thì trò chơi sẽ kết thúc

A black background with red text

Description automatically generated

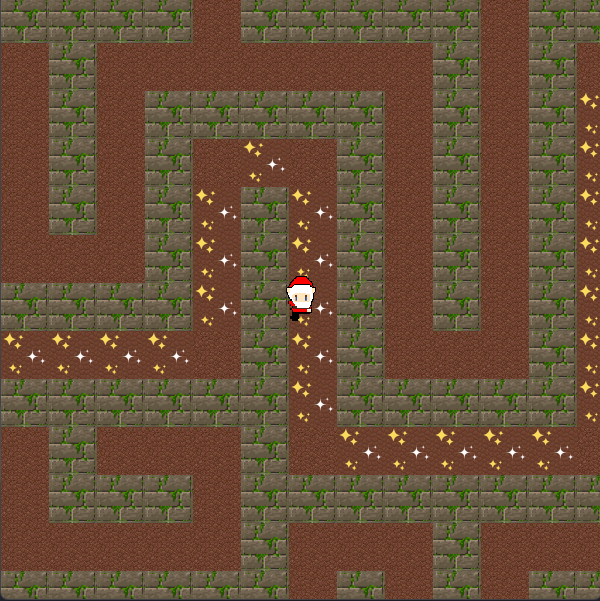
**Hình 5.5: Game kết thúc**

- Sau khi trả lời đúng câu hỏi ta sẽ nhận được vật phẩm kính lúp để mở ra con đường vượt qua mê cung tăm tối



**Hình 5.6: Đạt được vật phẩm**

- Vật phẩm kính lúp sẽ khai sáng con đường để nhận được vật phẩm dẫn đến kho báu cuối cùng



**Hình 5.7: Con đường khai sáng**

- Sau hành trình gian nan vượt qua nhiều thử thách khó khăn, ông già noel đã đến kho báu cuối cùng. Đó chính là những điểm A+ ngọt ngào, hạnh phúc



**Hình 5.8: Kho báu cuối cùng**

**Tài liệu tham khảo:**

<https://www.iostream.co/article/thuat-giai-a-DVnHj>

<https://viblo.asia/p/a-pathfinding-nwmkyEjlkoW>

<https://bloglaptrinh2016.wordpress.com/2016/06/09/a-thuat-toan-tim-kiem-a/>

<https://brilliant.org/wiki/a-star-search/>

<https://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html>

<https://d3c33hcgiwev3.cloudfront.net/_b65e7611894ba175de27bd14793f894a_15UnionFind.pdf?Expires=1702425600&Signature=eXc6VEsnY54aq66mJE9kRbSISr2xBjKheaR-TObT6c9BtLsfIHQMUPHr7RQROPVnCZ8Ss-eW4Oi1bIqafrKqPecL8ZO8YBHiolCA-vHxh~Wi4K15271YBo5U65pXtQUjdUwc9heD1TOCydcrF6bxofjF2p1SU7FHA~syglsZI4Y_&Key-Pair-Id=APKAJLTNE6QMUY6HBC5A>