**TRƯỜNG ĐẠI HỌC NGOẠI THƯƠNG**

**KHOA CÔNG NGHỆ VÀ KHOA HỌC DỮ LIỆU**

**---------\*\*\*\*\*\*---------**

A red circle with white text

AI-generated content may be incorrect.

**BÁO CÁO PROJECT**

**MÔN CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT**

**------------------------------**

**ĐỀ TÀI:**

**QUẢN LÝ HÀNG ĐỢI BỆNH VIỆN**

|  |  |
| --- | --- |
|  | **Các thành viên Nhóm 4:**  **Nguyễn Sỹ Bách**  **Vũ Đức Học**  **Nguyễn Ngọc Phương Linh**  **Trần Trung Nhân** |

# **MỤC LỤC**

[**MỤC LỤC** 2](#_Toc199235958)

[**1. Bài toán** 3](#_Toc199235959)

[**1.1. Đặt vấn đề** 3](#_Toc199235960)

[**1.2. Giới thiệu chung:** 3](#_Toc199235961)

[**1.3. Định nghĩa bài toán:** 3](#_Toc199235962)

[**2. Cấu trúc dữ liệu** 4](#_Toc199235963)

[**2.1. Danh sách (List)** 4](#_Toc199235964)

[**2.2. Hàng đợi ưu tiên (Priority Queue)** 4](#_Toc199235965)

[**2.3. Từ điển (Dictionary)** 4](#_Toc199235966)

[**3. Giải thuật** 5](#_Toc199235967)

[**3.1. Sắp xếp (Sort)** 5](#_Toc199235968)

[**3.2. Tìm kiếm tuyến tính (Linear Search)** 5](#_Toc199235969)

[**4. Cải tiến (cấu trúc dữ liệu hoặc thuật toán)** 5](#_Toc199235970)

[**4.1. Thay thế sử dụng List bằng Priority Queue** 5](#_Toc199235971)

[**4.2. Sử dụng Dictionary** 6](#_Toc199235972)

[**5. Kết quả (thực nghiệm, so sánh và triển khai)** 6](#_Toc199235973)

[**5.1. Thực nghiệm** 6](#_Toc199235974)

[**5.2. So sánh** 6](#_Toc199235975)

[**5.3. Triển khai** 6](#_Toc199235976)

[**6. Kết luận và future work** 7](#_Toc199235977)

[**6.1. Kết luận** 7](#_Toc199235978)

[**6.2. Future work** 7](#_Toc199235979)

[**7. Đánh giá thành viên** 8](#_Toc199235980)

[**8. Tài liệu tham khảo** 8](#_Toc199235981)

[**9. Phụ lục** 8](#_Toc199235982)

# **1. Bài toán**

## **1.1. Đặt vấn đề**

Hiện nay, ở một số bệnh viện vẫn tồn tại tình trạng ùn tắc hoặc bệnh nhân chen lấn hàng mặc dù họ không trong trường hợp không thật sự cần thiết. Tình trạng này xảy ra là do bệnh viện chưa có một hệ thống quản lý hàng đợi thật sự hiệu quả. Với hệ thống quản lý hàng đợi hợp lý, bệnh viện có thể đảm bảo công bằng cho bệnh nhân, cũng như ưu tiên các ca bệnh cần cấp cứu khẩn cấp, giúp giảm thiểu rủi ro về tính mạng. Đồng thời, việc này cũng tránh tình trạng ùn tắc, giảm căng thẳng cho cả nhân viên y tế và người bệnh, giúp tiết kiệm thời gian, tăng năng suất làm việc của bệnh viện. Cuối cùng, việc sắp xếp hàng đợi một cách khoa học còn thể hiện sự chuyên nghiệp của bệnh viện, giúp thu hút nhiều bệnh nhân hơn.

=> Đặt ra yêu cầu về một giải pháp giúp tối ưu hóa việc sắp xếp hàng đợi bệnh viện.

## **1.2. Giới thiệu chung:**

- Hệ thống được xây dựng nhằm hỗ trợ việc tổ chức và quản lý hàng chờ trong bệnh viện một cách khoa học, hiệu quả và công bằng cho người bệnh.

- Mục tiêu chính là giúp ưu tiên các bệnh nhân cần cấp cứu, giảm thời gian chờ đợi và nâng cao hiệu quả khám chữa bệnh.

- Hệ thống mô phỏng quy trình tiếp nhận và xếp hàng bệnh nhân theo mức độ ưu tiên, thời gian đến và trạng thái khám bệnh.

- Các chức năng chính:

+ Hiển thị danh sách bệnh nhân đang chờ khám theo thứ tự ưu tiên.

+ Sắp xếp bệnh nhân tiếp theo cần khám dựa trên:

1) Mức độ ưu tiên y tế: Emergency hoặc Normal

2) Thời gian đến bệnh viện

+ Cho phép người dùng thêm bệnh nhân mới vào hàng chờ, cập nhật hoặc xóa bệnh nhân đã khám khỏi hàng chờ.

+ Xem chi tiết thông tin từng bệnh nhân.

## **1.3. Định nghĩa bài toán:**

- Cần xây dựng một lớp Patient để lưu trữ thông tin và trạng thái của từng người trong hàng đợi.

- Các thuộc tính của lớp Patient bao gồm:

+ ID bệnh nhân

+ Họ và tên

+ Mức độ ưu tiên: "Emergency" hoặc "Normal"

+ Thời gian đến bệnh viện

+ Trạng thái: "chờ khám", "đang khám" hoặc "đã khám"

- Cần xây dựng một danh sách (Queue) để lưu trữ và quản lý danh sách bệnh nhân chờ khám.

- Xây dựng các hàm hỗ trợ chức năng hệ thống:

+ Hàm in ra thông tin toàn bộ hàng chờ bệnh nhân.

+ Hàm sắp xếp danh sách bệnh nhân dựa vào mức độ ưu tiên và thời gian đến.

+ Hàm cập nhật trạng thái bệnh nhân sau khi được khám xong.

+ Hàm thêm bệnh nhân mới vào hàng chờ theo định dạng chuẩn.

+ Hàm xóa bệnh nhân khỏi hàng đợi.

# **2. Cấu trúc dữ liệu**

## **2.1. Danh sách (List)**

- List là một kiểu cấu trúc dữ liệu tuyến tính, nơi các phần tử được sắp xếp theo một thứ tự cụ thể. Các phần tử trong danh sách có thể được truy cập theo thứ tự hoặc được truy cập theo vị trí của chúng trong danh sách.

- Sử dụng List để cập nhật và sắp xếp danh sách các bệnh nhân trong hàng chờ.

## **2.2. Hàng đợi ưu tiên (Priority Queue)**

- Hàng đợi ưu tiên là loại hàng đợi sắp xếp các phần tử dựa trên giá trị ưu tiên của chúng.

+ Mỗi phần tử có một mức độ ưu tiên liên quan. Khi thêm một phần tử mới, phần tử đó sẽ được chèn vào vị trí dựa trên mức độ ưu tiên của nó.

+ Các phần tử có mức độ ưu tiên cao hơn thường được truy xuất hoặc xóa trước các phần tử có mức độ ưu tiên thấp hơn.

**-** Độ ưu tiên gồm 2 mức: Emergency và Normal

=> Các bệnh nhân trong tình trạng Emergency sẽ được ưu tiên khám trước. Nếu 2 bệnh nhân có mức độ ưu tiên ngang nhau thì sẽ sử dụng quy tắc FIFO để xác định thứ tự bệnh nhân được khám bệnh.

## **2.3. Từ điển (Dictionary)**

- Kiểu dữ liệu từ điển (Dictionary) là một cấu trúc dữ liệu lưu trữ giá trị theo cặp khóa: giá trị, trong đó khóa phải là duy nhất.

- Mỗi bệnh nhân sẽ có 1 ID định danh của mình, và ID đó là duy nhất. Nhờ có ID riêng mà bệnh nhân không cần phải khai báo lại thông tin khi đến khám bệnh những lần tiếp theo, đồng thời bác sĩ có thể nắm được thời gian khám bệnh và các bệnh mà bệnh nhân đã mắc phải trước đó.

# **3. Giải thuật**

## **3.1. Sắp xếp (Sort)**

- Là sự kết hợp của:

+ Sắp xếp trộn (Merge Sort): Thuật toán này hoạt động bằng cách đệ quy chia mảng đầu vào thành hai nửa, đệ quy sắp xếp hai nửa và cuối cùng hợp nhất chúng lại với nhau để thu được mảng đã sắp xếp. Ở đây, các dữ liệu được chia thành 2 mảng chính dựa theo mức độ ưu tiên (Emergency và Normal).

+ Sắp xếp chèn (Insertion Sort): Đây là thuật toán sắp xếp đơn giản hoạt động bằng cách lặp lại việc chèn từng phần tử của một danh sách chưa được sắp xếp vào đúng vị trí của nó trong một phần đã được sắp xếp của danh sách. Khi thêm một bệnh nhân mới vào hàng chờ, thuật toán này sẽ giúp sắp xếp đúng thứ tự của bệnh nhân theo thứ tự ưu tiên đã quy định.

## **3.2. Tìm kiếm tuyến tính (Linear Search)**

- Thuật toán này sẽ giúp tìm kiếm phần tử thông qua việc duyệt qua tất cả các phần tử của mảng và kiểm tra xem phần tử hiện tại có bằng với phần tử mục tiêu hay không. Nhờ sử dụng thuật toán này mà người dùng có thể dễ dàng tìm kiếm, cập nhật hay xóa thông tin bệnh nhân theo ID.

# **4. Cải tiến (cấu trúc dữ liệu hoặc thuật toán)**

## **4.1. Thay thế sử dụng List bằng Priority Queue**

Nhờ sử dụng Priority Queue thay thế cho List mà có thể đem lại các ưu điểm sau:

1) Tối ưu hóa hiệu suất

- Với Priority Queue (dựa trên heap), bạn không cần sắp xếp toàn bộ danh sách sau mỗi lần thêm hay gọi bệnh nhân. Điều này cực kỳ hữu ích khi số lượng bệnh nhân lớn (ví dụ vài trăm hoặc hàng ngàn người chờ).

- Việc lấy bệnh nhân ưu tiên cao nhất (cấp cứu) luôn diễn ra tức thì (O(1)).

2) Giảm lỗi thao tác

- Khi dùng List, lập trình viên phải tự nhớ gọi hàm sort mỗi lần thêm hoặc cập nhật. Nếu quên, thứ tự hàng đợi sẽ sai.

- Với Priority Queue, cơ chế sắp xếp được đảm bảo tự động – giảm nguy cơ lỗi logic.

3) Sẵn sàng cho triển khai thực tế

- Trong môi trường bệnh viện thật, nơi thời gian và độ chính xác rất quan trọng, việc sử dụng Priority Queue giúp đảm bảo phục vụ đúng người, kịp thời, đặc biệt đối với các trường hợp khẩn cấp.

4) Linh hoạt hơn trong xử lý nhiều mức ưu tiên

- Bạn có thể mở rộng dễ dàng với nhiều cấp độ ưu tiên hơn chỉ bằng cách thay đổi hệ thống đánh trọng số.

## **4.2. Sử dụng Dictionary**

- Giúp lưu trữ đầy đủ thông tin bệnh nhân, phục vụ cho các lần khám bệnh trong tương lai.

- Tìm kiếm, truy xuất thông tin nhanh hơn, giúp tiết kiệm thời gian và tài nguyên.

# **5. Kết quả (thực nghiệm, so sánh và triển khai)**

## **5.1. Thực nghiệm**

- Triển khai hệ thống: Đã xây dựng thành công ứng dụng web Flask quản lý hàng đợi bệnh nhân tại bệnh viện, cho phép thêm, xóa, cập nhật và gọi bệnh nhân tiếp theo.

- Kiểm thử thực tế:

+ Nhập dữ liệu với nhiều loại bệnh nhân: Emergency và Normal, với thời gian đến khác nhau.

+ Quan sát thứ tự sắp xếp bệnh nhân khi hiển thị danh sách: hệ thống luôn ưu tiên bệnh nhân cấp cứu (Emergency) lên đầu, sau đó mới đến Normal.

+ Thực hiện chức năng "gọi bệnh nhân tiếp theo": luôn gọi đúng bệnh nhân cấp cứu trước, nếu không còn thì gọi bệnh nhân đến sớm nhất trong nhóm còn lại.

## **5.2. So sánh**

a) So sánh với sắp xếp hàng đợi thông thường

- Hàng đợi thông thường chỉ sắp xếp thứ tự bệnh nhân dựa trên quy tắc FIFO, bệnh nhân nào đến trước thì sẽ được khám trước.

- Hệ thống quản lý hàng đợi: kết hợp thêm sắp xếp theo mức độ ưu tiên, do đó đảm bảo được an toàn về tính mạng cho các trường hợp bệnh nhân cần cấp cứu khẩn cấp.

b) So sánh với sắp xếp hàng đợi thủ công

- Hệ thống quản lý hàng đợi sẽ sắp xếp tự động thứ tự bệnh nhân, từ đó sẽ giúp tiết kiệm nguồn nhân lực, giảm áp lực quá tải trong khoảng thời gian cao điểm, và giảm các lỗi thủ công có thể mắc phải.

## **5.3. Triển khai**

- Môi trường thử nghiệm: Chạy trên máy tính cá nhân, dùng Flask với host "0.0.0.0", debug=True để dễ kiểm thử.

- Khả năng mở rộng: Ứng dụng dễ dàng triển khai lên các server nội bộ của bệnh viện, có thể tích hợp với hệ thống quản lý bệnh viện lớn hơn.

- Khả năng tích hợp: Có thể bổ sung thêm chức năng như thống kê lượt khám, lưu trữ hồ sơ bệnh án của những lần khám trước, …

# **6. Kết luận và future work**

## **6.1. Kết luận**

- Hệ thống quản lý hàng đợi bệnh nhân giúp nâng cao hiệu quả trong việc tiếp nhận, phục vụ bệnh nhân trong bệnh viện.

- Việc sử dụng cấu trúc dữ liệu kết hợp danh sách và thuật toán sắp xếp theo ưu tiên giúp đảm bảo sự công bằng và đặt an toàn tính mạng của bệnh nhân lên hàng đầu.

- Hệ thống đã kiểm thử hoạt động ổn định, dễ triển khai, phù hợp với quy mô nhỏ đến vừa của các cơ sở y tế.

## **6.2. Future work**

- Nâng cấp hiệu suất với cấu trúc Priority Queue (heap): Khi dữ liệu lớn, sử dụng heap để tối ưu thao tác thêm, gọi, xóa bệnh nhân.

- Tính năng tự động chuyển sang gọi bệnh nhân tiếp theo sau khi bệnh trước trước đã khám xong.

- Tính năng xóa bệnh nhân đã khám xong.

- Kết nối database: Chuyển từ file JSON sang database như SQLite, MySQL để tăng khả năng lưu trữ, mở rộng và bảo mật.

- Tích hợp hệ thống thông báo: Thông báo cho bệnh nhân khi đến lượt, hoặc nhắc nhở nhân viên y tế.

- Quản lý đa hàng đợi: Hỗ trợ phân chia bệnh nhân về từng phòng khám cụ thể theo độ tuổi, khoa khám, …

# **7. Đánh giá thành viên**

- Nguyễn Sỹ Bách: Làm bản báo cáo + cải thiện code (25%)

- Vũ Đức Học: Làm slide, nội dung thuyết trình + cải thiện code (25%)

- Nguyễn Ngọc Phương Linh: Làm slide, nội dung thuyết trình + cải thiện code (25%)

- Trần Trung Nhân: Làm nội dung code cơ bản + cải thiện code (25%)

# **8. Tài liệu tham khảo**

**1.** [**https://www.geeksforgeeks.org/python-dictionary/**](https://www.geeksforgeeks.org/python-dictionary/)

**2.** [**https://www.geeksforgeeks.org/priority-queue-set-1-introduction/**](https://www.geeksforgeeks.org/priority-queue-set-1-introduction/)

**3.** [**https://www.geeksforgeeks.org/merge-sort/**](https://www.geeksforgeeks.org/merge-sort/)

**4.** [**https://www.geeksforgeeks.org/insertion-sort-algorithm/**](https://www.geeksforgeeks.org/insertion-sort-algorithm/)

**5.** [**https://www.geeksforgeeks.org/linear-search/**](https://www.geeksforgeeks.org/linear-search/)

# **9. Phụ lục**

Link code: [Commits · nhansp/hq · GitHub](https://github.com/nhansp/hq/commits/main/) https://github.com/nhansp/hq

Link web: 13.229.205.139:5000