CS 330 Dự án 2: Công cụ nhân tố LU

1. Giới thiệu

Đối với dự án này, bạn sẽ tạo một thư viện C99 để giải các phương trình tuyến tính có dạng Ax = b bằng cách sử dụng hệ số LU với phép xoay một phần, như đã thảo luận trong lớp. Mục tiêu là tạo ra một thư viện phần lớn khép kín mà phần mềm khác có thể sử dụng để giải quyết vấn đề, do đó bạn không cần phải lo lắng về việc đọc dữ liệu đầu vào.

Tôi đã cung cấp một số tệp khung trong Canvas. Chúng bao gồm một tệp tiêu đề cho thư viện của bạn (bạn có thể thay đổi cấu trúc như đã lưu ý trong Phần 2), phần bắt đầu tại chính thư viện và một tệp để kiểm tra thư viện của bạn. Những gì cần nộp được mô tả trong Phần 4.

2 mô-đun phân tích hệ số LU

Bạn sẽ tạo mô-đun hệ số LU có thể tái sử dụng trong C99 với giao diện sau:

```
typedef struct { ... } LUfact; LUfact
*LUfactor(int N, const double **A); void
LUdestroy(LUfact*); void
LUsolve(LUfact *fact, const double *b, double *x);
```

LUfact : cấu trúc dữ liệu mờ chứa thông tin hệ số LU cần thiết.

LUfactor() : quy trình này thực hiện phân tích nhân tử (xem ghi chú khóa học để biết chi tiết) và phân bổ, điền và trả về một đối tượng LUfact. Nếu ma trận A là số ít trả về NULL. Lưu ý rằng ma trận A được lưu trữ dưới dạng vectơ của con trỏ hàng.

LUdestroy() : giải phóng dữ liệu được phân bổ trong LUfactor().

LUsolve() : giải hệ Ax = b cho x.

2.1 Cấu trúc dữ liệu hệ số LU

Hàm LUfactor() phân bổ và điền vào một đối tượng LUfact chứa thông tin phân tích hệ số. Bạn có thể triển khai cấu trúc này theo cách bạn muốn (xét cho cùng thì nó có nghĩa là không rõ ràng), mặc dù biểu mẫu sau đây có lẽ là điểm khởi đầu tốt:

```
cấu trúc typedef {
     */ int              /* Kích thước của ma trận NxN đầu vào A
     N; gấp đôi **LU; /* Ma trận NxN chứa ma trận L và U kết hợp */ ngắn *biến đổi; /*
     Hoán vị hàng của A */
} LUfact;
```

Lưu ý rằng xoay vòng một phần có nghĩa là bạn phải theo dõi cách các hàng được hoán đổi. Ở đây, người đột biến mảng nhằm mục đích giữ ánh xạ từ vị trí ban đầu của hàng sang vị trí mới.

3 Một trường hợp thử nghiệm đơn giản

- Ví dụ về ma trận 4 × 4:

$$A = \begin{matrix} 1 & & & 2 \\ 3 & & 11 & 2 \\ & 1 & 4 & 1 \\ 2 & 3 & 1 3 & 3 \end{matrix}$$

- Phân tách tương ứng (không xoay vòng một phần) là

$$LU \approx \begin{matrix} 1.000 & 0,000 & 0,000 & 0,000 \\ 1.000 & 1.000 & 0,000 & 0,000 \\ 3.000 & 1.600 & 1.000 & 0.000 \\ 3.000 & 1.200 & 0.857 & 1.000 \end{matrix} \begin{matrix} 1.000 & 3.000 & 1.000 & 2.000 \\ 0,000 & 5,000 & 2,000 & 4.000 \\ 0,000 & 0,000 & 4,200 & 1.400 \\ 0,000 & 0,000 & 0,000 & 3,000 \end{matrix}$$

- Sự phân tách sau đây là kết quả của việc xoay vòng một phần và tương ứng với các hàng {2, 0, 3, 1} của MỘT :

$$LU \approx \begin{matrix} 1.000 & 0.000 & 0.000 & 0.000 \\ 0,333 & 1,000 & 0,000 & 0,000 \\ 1.000 & 0.750 & 1.000 & 0.000 \\ 0,333 & 0,875 & 0,389 & 1,000 \end{matrix} \begin{matrix} 3.000 & 1.000 & 4.000 & 1.000 \\ 0,000 & 2,667 & 0,333 & 1,667 \\ 0,000 & 0,000 & 6,750 & 5,250 \\ 0,000 & 0,000 & 0,000 & 1,167 \end{matrix}$$

Xin lưu ý rằng bạn phải triển khai xoay vòng một phần trong thư viện của mình. Giải pháp không cần xoay vòng, ở trên, chỉ được trình bày để bạn tham khảo trong trường hợp bạn muốn triển khai mọi thứ mà không cần xoay vòng trước.

4 Những gì cần nộp

Gửi tệp lưu trữ bao gồm LUfact.c và LUfact.h, đảm bảo mã của bạn được định dạng gọn gàng, được nhận xét tốt và nhận dạng được bạn, khóa học và dự án. Bạn không cần phải bao gồm LUtest.c Nếu thư viện bao gồm các tệp nguồn ngoài cặp có tên ở trên, bao gồm hướng dẫn xây dựng thư viện của bạn.