Chương trình đào tạo Việt Nhật - HEDSPI

Đề thi Lập trình C nâng cao – K54

(Thời gian làm bài 120 phút - Được phép sử dụng các bài tập và cấu trúc dữ liệu đã thực hành – Tuyệt đối cấm sao chép)

Sử dụng lý thuyết đồ thị giải bài toán tìm đường trong MÊ CUNG

Mê cung được mô hình bằng một ma trận vuông kích thước nxn ví dụ như sau (dữ liệu chuẩn sẽ công bố trước khi hết giờ 40 phút).

```
0 1 1 1 0 0
0 0 1 1 1 1
1 0 0 0 1 0
0 0 0 0 1 0
1 1 1 0 0 0
1 1 1 1 1 0
```

Trong đó các vị trí có giá trị 0 là đường có thể đi qua. Giá trị 1 đại diện cho vị trí tường chắn. ***Cửa vào và ra của mê cung ứng với các vị trí có giá trị 0 nằm ở biên mê cung (hàng 0, cột 0, hàng n-1, cột n-1)***. Để tìm đường đi trong mê cung người ta biểu diễn mê cung bằng đồ thị vô hướng theo cách sau. Mỗi một phần tử trong ma trận có giá trị 0 ứng với một đỉnh có nhãn là vị trí của nó trong đồ thị, ví dụ phần tử [0,0] - ứng với đỉnh nhãn 00. phần tử [4,5] - ứng với đỉnh nhãn 45. Các phần tử có giá trị 1 bị loại. Các cạnh của đồ thị dùng để nối 2 đỉnh liền kề. Một đỉnh có tối đa 4 đỉnh liền kề ở các vị trí trên, dưới, trái, phải.

Viết chương trình **MAZE SEARCH** tổ chức dưới dạng menu (2 điểm) để thực hiện các công việc sau

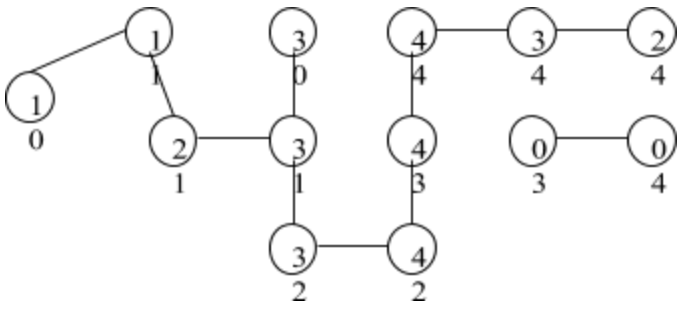
1. Đọc ma trận mê cung từ tệp mecung.txt để tạo ra đồ thị. Sau khi đọc in lại mê cung ra màn hình. **(1.5 điểm)**
2. In ra màn hình tổng số đỉnh và tổng số cạnh có trong đồ thị. Đồ thị của mê cung trong ví dụ trên có 18 đỉnh và 16 cạnh **(1.5 điểm).**
3. Nhập vào từ bàn phím một đỉnh. In ra các đỉnh kề với nó. Nếu đỉnh không tồn tại in ra thông báo. Ví dụ nhập vào 11 sẽ in ra 2 đỉnh liền kề là 10 và 21. **(1điểm)**
4. Nhập vào 2 đỉnh và in ra đường đi ngắn nhất giữa 2 điểm đó. Khoảng cách giữa 2 đỉnh kề nhau luôn là 1. Ví dụ in ra đường đi từ đỉnh 10 đến đỉnh 43 như sau: 00->10->11->21->31->32->33->43. Một lời giải khác là 00->10->11->21->22->23->33->43. Gợi ý: Dùng BFS. Chú ý kiểm tra tính hợp lệ của đỉnh nhập vào. **(2 điểm)**
5. Nhập vào một đỉnh là cửa vào, in ra đường đi ngắn nhất giúp thoát khỏi mê cung (tức là đường đi nối tới một cửa ra bất kỳ). *Cửa ra không thể trùng cửa vào*. Chú ý kiểm tra bắt lỗi việc nhập cửa vào không hợp lệ (đỉnh không tồn tại, tồn tại nhưng không phải cửa vào) . Ví dụ cửa vào là 00 thì đáp án là 00->10, cửa vào là 30 thì đáp án là 30->31->21->11->10 **(1 điểm)**
6. In ra thành phần liên thông lớn nhất trong đồ thị - chính là tập lớn nhất các đỉnh có giá trị 0 kết nối được với nhau bằng đường đi. Mê cung ví dụ có 2 thành phần liên thông. Thành phần nhỏ hơn nằm ở góc trên phải: 04. 05 **(1 điểm)**

Ghi chú: Nếu bài làm không sử dụng thư viện đồ thị cài đặt dùng JRB thì bị trừ 2 điểm

Dữ liệu chấm thi

1) Nội dung mecung.txt

1 1 1 0 0
0 0 1 1 1
1 0 1 1 0
0 0 0 1 0
1 1 0 0 0



2) Tổng số đỉnh: 13 bao gồm:
10,11,21,31,30,32,42,43,44,34,24 và 03, 04
Tổng số cạnh: 11

3) Nhập 33 (hoặc 02, 12, 22, 23 ,41) – Nếu không in ra đỉnh không tồn tại (trừ 0.5 điểm)
Nhập 43: In ra 42, 44 – Nhập 31 in ra 21 30 32

4) Nhập đỉnh không tồn tại – kiểm tra bắt lỗi (Cả 2 đầu) – 0,5

Nhập 21 và 24: 21 – 31 -32 -42 -43 - 44 – 34 -24

Nhập 31 và 10: 31 – 21 – 11 – 10

5) Nhập 02 – báo ko là đỉnh
Nhập 31 – báo không phải là cửa vào (Không bắt lỗi) trừ 0,5

Nhập vào 10 → đường thoát ngắn nhất : 10 11 21 31 30
Nhập vào 03 : 03 04
Nhập vào: 30: 30 31 32 33.

6) 10 11 21 31 30 32 42 43 44 34 24
Cách làm: Thuật toán BFS – (xét từ trên xuống dưới – từ trái sang phải hoặc theo cách nào đó đảm bảo duyệt hết các đỉnh)

Bắt đầu từ 1 đỉnh → Cho vào Queue → Khi lấy ra đánh dấu đã thăm và cho Các đỉnh kề với nó vào Queue. Mỗi lần thăm 1 đỉnh mới tăng biến đếm - đưa vào mảng kết quả. Chạy khi không thăm được nữa.
Nhảy sang đỉnh khác để thăm thành phần liên thông còn lại.