**TẠO CHƯƠNG TRÌNH MÔ PHỎNG DIJKSTRA SỬ DỤNG TKINTER**

1. **Yêu cầu bài toán**

Sử dụng thư viện Tkinter của Python tạo ra một GUI mô phỏng kết quả của các thuật toán tìm đường đi như Dijkstra

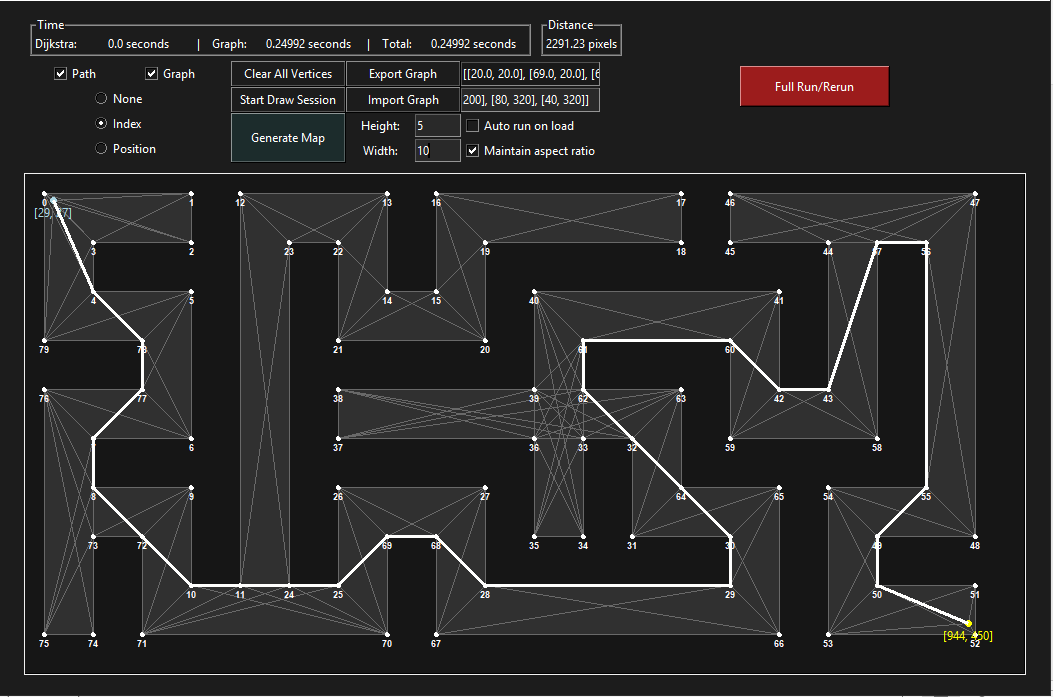
Yêu cầu đầy đủ bao gồm:

* + Biểu hiện một bản đồ thể hiện được khu vực đi được
  + Biểu hiện một bản đồ gồm những đường đi trong khu vực đi được
  + Biểu hiện đường đi ngắn nhất từ 2 điểm cho trước nằm trong khu vực đi được
  + Biểu hiện thời gian chạy của thuật toán và độ dài quãng đường
  + Có khả năng đổi giữa nhiều bản đồ khác nhau

Chương trình hiện tại có khả năng thực hiện tất cả những yêu cầu trên, và:

* + Khả năng tạo ra vô tận loại bản đồ khác nhau một cách ngẫu nhiên
  + Khả năng lưu và tải lại bản đồ đã lưu
  + Khả năng vẽ bản đồ sử dụng chuột

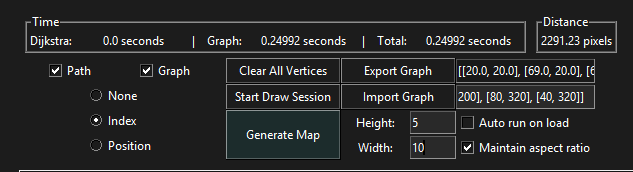
1. **Nhìn chung**



*Hình ảnh chương trình hiện tại*

1. Màn hình chính

Chương trình được viết theo hường OOP (dù không cần thiết), sử dụng Frame và LabelFrame để chứa các Widget(Button, Checkbutton, Entry,...), và sử dụng Grid để tải các Widget lên màn hình chính ngay khi chương trình bắt đầu



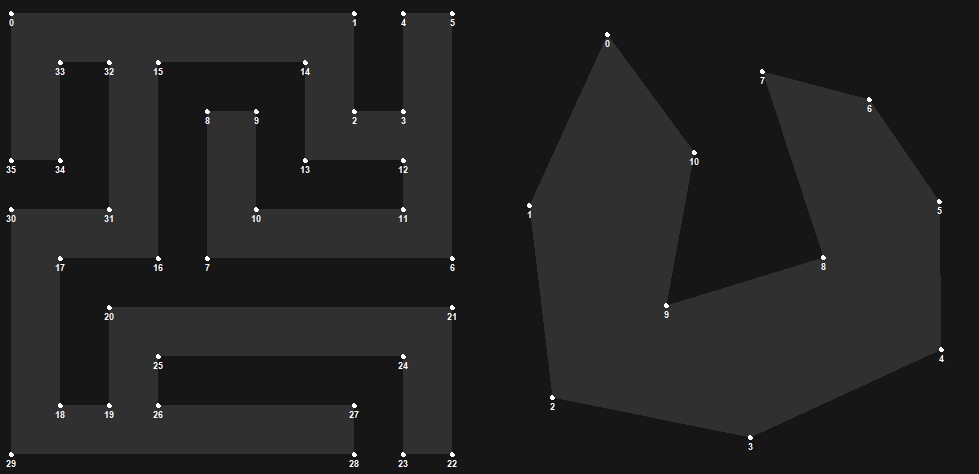
Chương trình sử dụng Canvas để vẽ ra bản đồ, đường đi được và đường đi ngắn nhất bằng cách xóa đi vẽ lại theo sự kiện cho vào bởi người dùng.

* Để tải lại Canvas, lưu ý cần sử dụng Canvas.delete(“all”) để xóa toàn bộ hình vẽ hiện tại trên Canvas. Vì Canvas có lưu lại từng phương pháp vẽ thành từng Object trong chương trình, việc không chạy Canvas.delete(“all”) mỗi loop có thể tạo Memory Leak khi chương trình chạy quá lâu

1. Bản đồ (trong Canvas)

Bản đồ đường đi là phần đầu tiên được vẽ lên Canvas vì nó cho những thông tin quan trọng nhất để mô phỏng Dijkstra như: vùng đi được, tường,...

Bản đồ hiện tại được thể hiện qua một List những đỉnh (một list chứa 2 tọa độ x, y) tạo thành một đa giác. Hiện tại bản đồ chỉ có thể có đa giác không có khoảng trống bên trong, và chỉ có tối đa một đa giác



*Một số hình ví dụ của bản đồ với các đỉnh được đánh thứ tự*

Các đỉnh của đa giác được sắp xếp theo thứ tự vẽ nên bản đồ có thể dễ dàng được vẽ lên Canvas sử dụng Canvas.create\_polygon()

1. Điểm đầu/cuối (trong Canvas)

Điểm đến và đi trong bản đồ có thể được đặt bằng chuột trái/phải bằng cách sử dụng bind() hay cụ thể hơn là Canvas.bind() trong trường hợp hiện tại để gán một function nhất định và thực hiện nó khi sự kiện cụ thể xảy ra trong Widget được gán





1. Đường đi (trong Canvas)

Để có thể thực hiện thuật toán Dijkstra, việc tính ra các đoạn đường đi được trong bản đồ là cần thiết. Do có thể dễ dàng thấy được rằng đường đi ngắn nhất sẽ được tạo bởi các đường đi qua từng đỉnh, ta có thể sử dụng List những đỉnh từ trước để thực hiện việc tìm tất cả những liên kết giữa các đỉnh trong bản đồ, từ đó sử dụng nó để tìm đường đi.

Thuật toán nói chung của việc tìm tất cả những liên kết phù hợp trong bản đồ là như sau:

* Kiểm tra tất cả đôi điểm khác nhau trong đa giác, với mỗi đôi:
  + Kiểm tra đôi điểm có phải 1 cạnh của đa giác bản đồ hay không

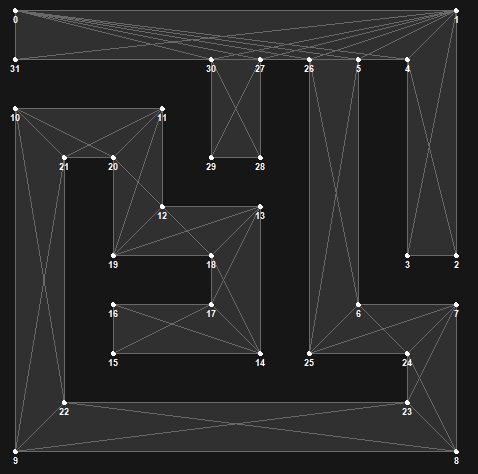
+ Nếu có: đường đi phù hợp và dừng kiểm tra

+ Nếu không:

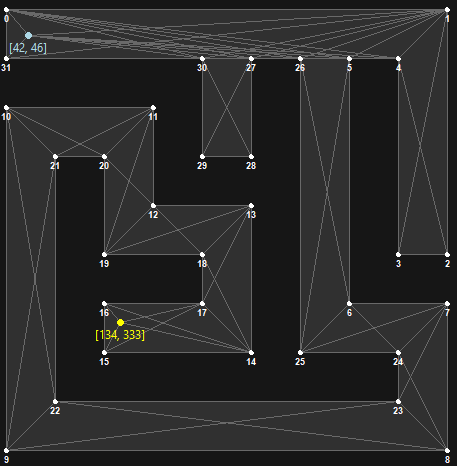
* + Kiểm tra đoạn thẳng tạo bởi đôi điểm có cắt/đè trên 1 cạnh của bản đồ hay không.
  + Kiểm tra đoạn thẳng có nằm trong bản đồ hay không.

+ Nếu cả 2 đều có: đường đi phù hợp

Sau khi chạy thuật toán, sử dụng Canvas.create\_line() để vẽ những đường đi được sẽ ra kết quả như hình dưới:

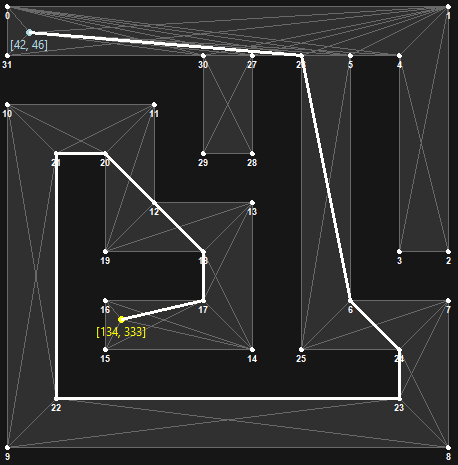


Từ đây, ta có thể tìm tất cả liên kêt từ điểm đầu/cuối tới tất cả các đỉnh của đa giác với phương pháp gần tương tự:



Chia việc tìm liên kết từng đỉnh trong đa giác với nhau và tìm liên kết giữa điểm đầu/cuối với đa giác là để có thể giảm thời gian chạy thuật toán trong trường hợp điểm đầu/cuối có sự thay đổi về vị trí.

Sau khi có được tất cả liên kết cần thiết, ta có thể sử dụng thuật toán Dijkstra để tìm đường đi ngắn nhất:



1. Thời gian chạy và quãng đường

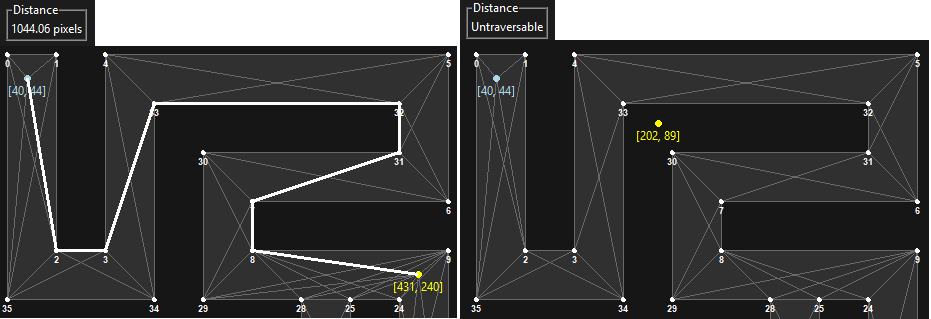
Cả thời gian chạy và quãng đường đều được hiện lên màn hình sử dụng tkinter.Label() và gán tkinter.StringVar() vào argument textvariable của Label. Từ đây, ta có thể sử dụng phương pháp tkinter.StringVar.set() để update giá trị của các Label khi có tương tác của người dùng.

Để tính thời gian chạy của các thuật toán vẽ/tìm đường đi, ta sử dụng thư viện **time** của Python, lưu thời gian bắt đầu và kết thúc của thuật toán bằng cách sử dụng phương pháp time.time() ở đầu và cuối các thuật toán, sau đó lấy hiệu của hai giá trị làm thời gian chạy.



Quãng đường của đường đi ngắn nhất có thể dễ dàng được tính bằng cách lưu lại các đỉnh nối với nhau thành đường đi, sau đó tính quãng đường từng đoạn một nhờ vào vị trí các điểm và lấy tổng các giá trị đó.

Ngoài ra, nếu không thể tìm thấy được đường đi, ta có thể thay bằng một giá trị đặc biệt cho thấy rằng không tìm thấy được đường đi phù hợp



1. Nhập xuất bản đồ

Sử dụng Tkinter.Entry, chương trình có thể nhận và xuất bản đồ hiện tại dưới dạng một List các đỉnh đa giác. Để có thể chuyển một List dưới dạng String sang một List trong Python, ta sử dụng thư viện **ast** và dùng hàm ast.literal\_eval().

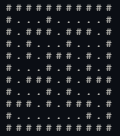
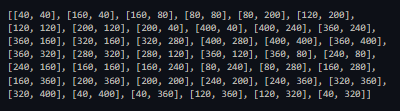




1. **Thuật toán**
2. Tạo bản đồ ngẫu nhiên

Bản đồ trong chương trình có thể được tạo một cách ngẫu nhiên bằng cách sử dụng nút “Generate Random Map”, cho ra một bản đồ là một đa giác với một số lượng hàng và cột được cho từ người dùng.

Quá trình tạo bản đồ được chia ra làm hai bước: Tạo một bản đồ ban đầu bằng kí tự, sau đó chuyển các đỉnh của bản đồ đó sang tọa độ đỉnh cho chương trình chính

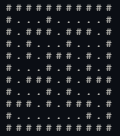
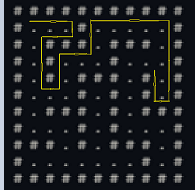
 

Bản đồ kí tự ban đầu được tạo bằng thuật toán DFS ngẫu nhiên gồm các bước:

* + Tạo một List hoạt động như một Stack chứa các điểm đã đi qua
  + Chọn một điểm ngẫu nhiên để bắt đầu, cho điểm đó vào Stack

1. Mỗi lần lặp, chọn một điểm ngẫu nhiên chưa được đi qua và bên cạnh ô đầu Stack để cho vào đầu Stack
2. Tạo đường đi giữa điểm đầu Stack cũ và mới
3. Nếu không kiếm được điểm, bỏ phần tử đầu khỏi Stack
4. Lặp từ bước 1 cho đến khi Stack hết phần tử

Từ đó, ta chuyển bản đồ được tạo thành các tọa độ đỉnh như hình vẽ dùng phương pháp ôm bên trái

1. Kiểm tra đường đi phù hợp

Đường đi phù hợp được kiểm tra bằng hai bước được liệt kê trước. Để kiểm tra xem đường đi hiện tại có xuyên qua tường hay không, ta có thể sử dụng phương pháp kiểm tra bằng chiều quay tam giác để kiểm tra hai đoạn thẳng cắt nhau.

Sau đó, ta dùng phương pháp từ đường dẫn này để kiểm tra xem đường đi có trong đa giác hay không bằng cách kiểm tra xem trung điểm của đường đi có nằm trong đa giác hay không.