Create an orthogonal polygon with n arbitrary vertices

Mục lục

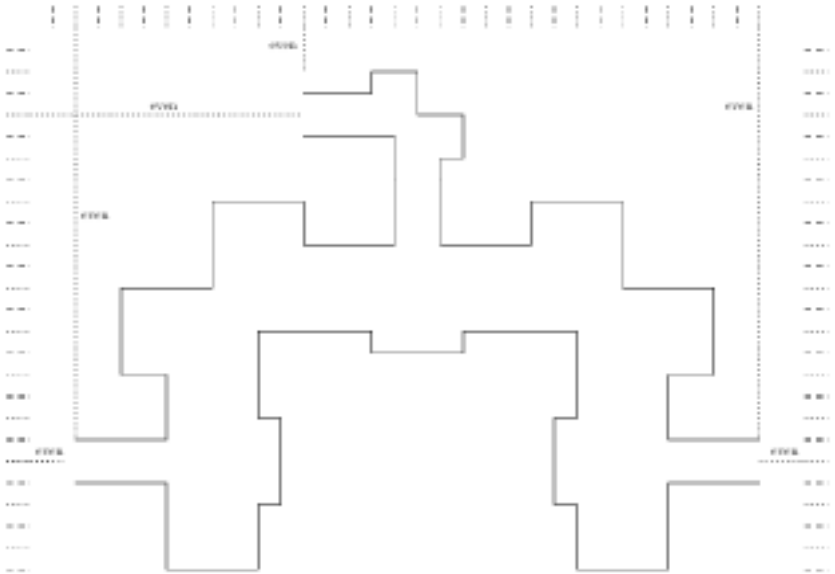
1. Kiến thức cần chuẩn bị
2. Mô tả và trình bày thuật toán
3. Ví dụ và hình ảnh minh họa
4. Đề xuất hướng cải thiện bài toán
5. Kết luận

I. Kiến thức cần chuẩn bị

- Khái niệm đa giác trực giao
- Hệ tọa độ Descartes
- Cấu trúc dữ liệu và thuật toán cơ bản
- Ngôn ngữ lập trình Julia
- Thư viện Plots.jl
- Hình học tính toán
- Kiến thức về số ngẫu nhiên

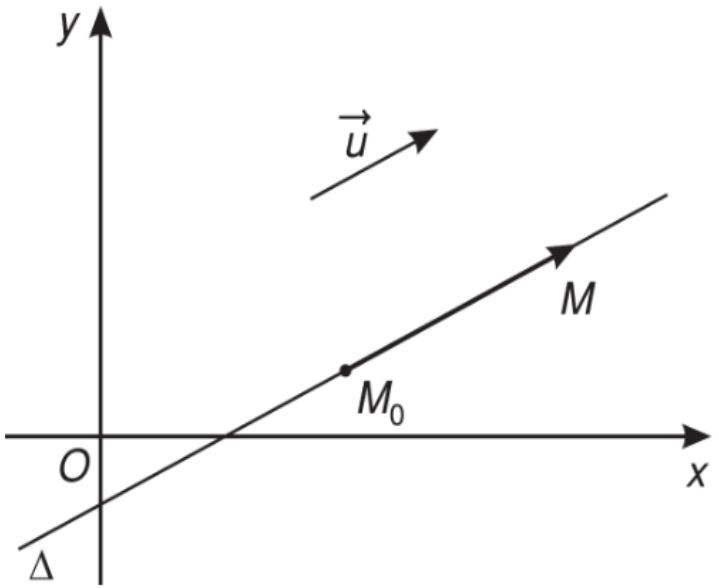
1. Khái niệm đa giác trực giao

- Đa giác đơn, các cạnh song song với trục hoành hoặc trục tung.
- Góc trong 90° (vuông) hoặc 270° (lồi).
- Chu trình đóng, không giao nhau.



2. Hệ tọa độ Descartes

- Biểu diễn các điểm trong mặt phẳng 2 chiều dưới dạng tọa độ (x,y).
- Cách tính toán khoảng cách và xác định hướng giữa các điểm.



Tích vô hướng:

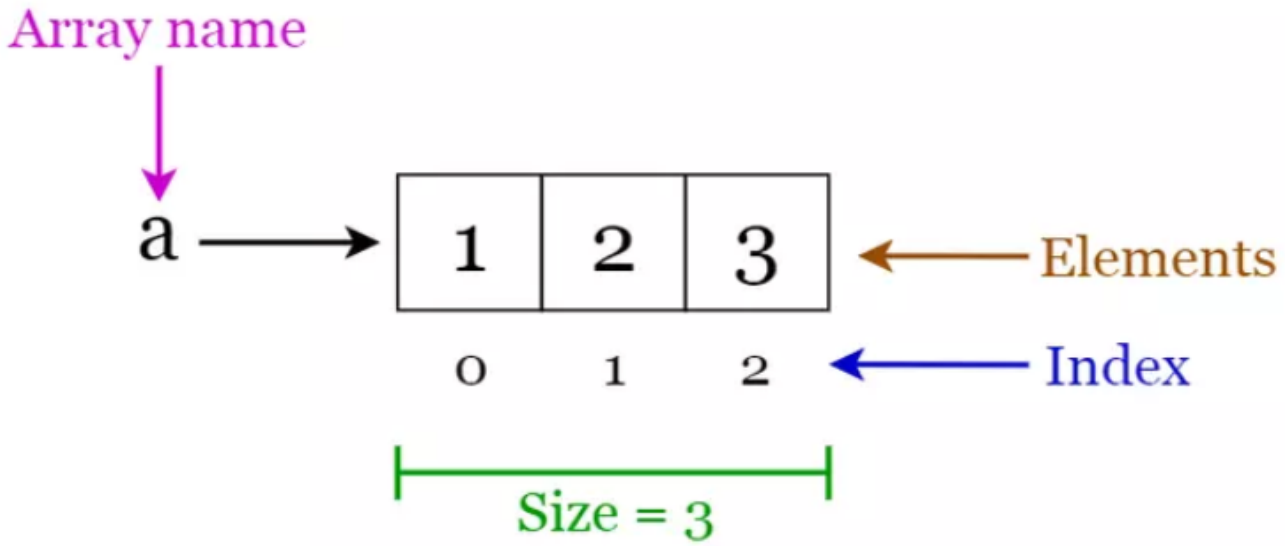
- Với hai vectơ $\vec{u} = (x_1, y_1)$, $\vec{v} = (x_2, y_2)$, ta có:

$$\vec{u} \cdot \vec{v} = x_1 x_2 + y_1 y_2 (= \|\vec{u}\| \cdot \|\vec{v}\| \cdot \cos(\theta))$$

- Ứng dụng: Tính góc giữa hai vectơ, kiểm tra vuông góc ($\vec{u} \cdot \vec{v} = 0$)

3. Cấu trúc dữ liệu và thuật toán cơ bản

- Tập hợp (Set): Dùng để kiểm tra điểm đã xuất hiện hay chưa.
- Danh sách (List / Array): Dùng để lưu trữ đường đi, các đỉnh của đa giác.
- Vòng lặp và rẽ nhánh: Được sử dụng để điều hướng và xây dựng cấu trúc fractal.
- Hiểu về khái niệm và cách xây dựng chu trình trên đồ thị.



4 & 5. Ngôn ngữ Julia và Thư viện Plots.jl

Ngôn ngữ Julia

- Khai báo hàm, biến, package
- Xử lý ngoại lệ (try-catch)
- Thao tác với Tuple, Array, Set

Thư viện Plots.jl

- Trực quan hóa dữ liệu với backend GR
- Lệnh: plot, scatter, fill, savefig
- Thiết lập tỷ lệ, màu sắc, tiêu đề

6. Hình học tính toán

- Hiểu biết cơ bản về hình học rời rạc: xác định hướng quay (trái/phải/lên/xuống), kiểm tra giao nhau giữa các đoạn thẳng.
- Ứng dụng trong việc xây dựng và xác minh tính chất của đa giác trực giao.

7. Kiến thức về số ngẫu nhiên

- Sử dụng hàm rand trong Julia để chọn vị trí, sinh nhánh một cách ngẫu nhiên.
- Hiểu được tác động của tính ngẫu nhiên đến kết quả hình học.

II.1. Mô tả thuật toán

- Tạo đa giác trực giao fractal khép kín với số đỉnh do người dùng chỉ định.
- Sử dụng Julia và thư viện Plots.
- Năm hàm chính:
  1. `is_right_angle`
  2. `generate_fractal_like_path`
  3. `save_path_to_csv`
  4. `draw_path`
  5. `main`

1. Hàm `is_right_angle`

Hàm `is_right_angle` kiểm tra xem góc tại điểm p_2 (giữa hai đoạn thẳng p_1p_2 và p_2p_3) có phải là góc vuông không.

Các bước thực hiện:

- Tính vector $\vec{v}_1 = (p_2[1] - p_1[1],\ p_2[2] - p_1[2])$
- Tính vector $\vec{v}_2 = (p_3[1] - p_2[1],\ p_3[2] - p_2[2])$
- Tính tích vô hướng: $\vec{v}_1 \cdot \vec{v}_2 = v_1[1] \cdot v_2[1] + v_1[2] \cdot v_2[2]$
- Nếu tích vô hướng bằng 0 $\Rightarrow$ góc tại p_2 là 90° $\Rightarrow$ trả về `true`, ngược lại trả về `false`.

2. Hàm `generate_fractal_like_path`

- **Mục tiêu**: Tạo danh sách tọa độ cho đa giác fractal(Path)
- **Các bước**:
  - Khởi tạo hình vuông khép kín:
    $(0,0) \to (1,0) \to (1,1) \to (0,1) \to (0,0)$
  - Lưu vào visited, 4 góc 90°
  - Thêm đỉnh ngẫu nhiên (2 đỉnh) cho đến khi đủ `num_points`
  - Giới hạn di chuyển (4 hướng): $(1,0), (-1,0), (0,1), (0,-1)$
- **Độ phức tạp**: $O(n^2)$ trung bình, tối đa $O(T * n^2)$

3. Hàm `save_path_to_csv`

- Lưu path vào file csv:
  - Ghi tiêu đề "x, y", rồi mỗi tọa độ (x, y) thành "x, y"

4. Hàm `draw_path`

- Trực quan hóa đa giác bằng Plots.jl
- Tách tọa độ x, y, vẽ nét mỏng, tỷ lệ 1:1
- Tô màu vùng trong (độ trong suốt 0.15, nếu yêu cầu)
- Đánh dấu đỉnh bằng hàm `is_right_angle`, vẽ chấm đỏ
- Điểm đầu/cuối (ngôi sao đỏ)

5. Hàm `main`

- Điều phối quy trình:
  - Nhập số đỉnh, kiểm tra hợp lệ (chẵn, ≥ 4)
  - Gọi `generate_fractal_like_path` và `draw_path`
  - Lưu ảnh PNG (`fractal_<num_points>_dinh.png`)
  - Xử lý lỗi, in thời gian thực thi
- Đảm bảo đa giác khép kín, trực giao, không trùng đỉnh.

II.2. Trình bày thuật toán

Thuật toán is_right_angle

Input: Tọa độ ba điểm $p_1(x_1, y_1), p_2(x_2, y_2), p_3(x_3, y_3)$

Output: Trả về `True` nếu góc tại p_2 là 90 độ, ngược lại trả về `False`

Bước 1: Xác định vecto từ p_1 đến p_2:

$v_1 = (x_2 - x_1, y_2 - y_1)$

Bước 2: Xác định vecto từ p_2 đến p_3:

$v_2 = (x_3 - x_2, y_3 - y_2)$

Bước 3: Tính tích vô hướng của hai vecto v_1 và v_2:

$v_1[1] \times v_2[1] + v_1[2] \times v_2[2]$

Bước 4: Kiểm tra nếu tích vô hướng bằng 0:

if $v_1[1] \times v_2[1] + v_1[2] \times v_2[2] == 0$ **then**

Return: `True` $\rightarrow$ Góc tại p_2 là 90 độ

else

Return: `False` $\rightarrow$ Góc tại p_2 không phải là 90 độ

end if

II.2. Trình bày thuật toán

Algorithm 1 generate_fractal_like_path(num_points, step)

Require: `num_points` là số chẵn và ≥ 4

1: **if** *num_points* lẻ **hoặc** *num_points* < 4 **then**
2: **throw** lỗi
3: **Khởi tạo:**
4: `path` $\leftarrow \{(0, 0), (1, 0), (1, 1), (0, 1), (0, 0)\}$
5: `visited` $\leftarrow$ các điểm trong `path`
6: `directions` $\leftarrow \{(1,0), (0,1), (-1,0), (0,-1)\}$
7: `right_angle_count` $\leftarrow 4$
8: **while** `right_angle_count` $<$ `num_points` **do**
9: Chọn ngẫu nhiên đoạn (p, q) trong `path`
10: $(dx, dy) \leftarrow q - p$
11: `perp_dirs` $\leftarrow$ hoán vị của $\{(dy, -dx), (-dy, dx)\}$
12: **for all** hướng $(test\_dx, test\_dy)$ trong `perp_dirs` $\cup$ `directions` **do**
13: $r \leftarrow p + (test\_dx, test\_dy) \times step$
14: $s \leftarrow r + (dx, dy) \times step$
15: **if** $r, s \notin visited$ **then**
16: Chèn r, s vào `path` tại vị trí sau p
17: `new_count` $\leftarrow$ `right_angle_count`
18: **if** `is_right_angle`(p, r, s) **then**
19: `new_count++`
20: **if** `is_right_angle`(r, s, q) **then**
21: `new_count++`
22: **if** góc tại p hoặc q thay đổi **then**
23: cập nhật `new_count`
24: **if** `new_count` $\leq$ `num_points` **then**
25: Thêm r, s vào `visited`
26: `right_angle_count` $\leftarrow$ `new_count`
27: **if** `right_angle_count` $=$ `num_points` **then**
28: **return** `path`
29: **else**
30: Loại bỏ r, s khỏi `path`
31: **return** `path`

II.2. Trình bày thuật toán

Hàm `save_path_to_csv(path, filename)`

- Mở file `filename` để ghi.
- Ghi tiêu đề `"x,y"`.
- Với mỗi điểm (x, y) trong `path`:
  - Ghi `"$x,$y"` vào file.
- Đóng file.

Hàm `draw_path(path, num_points, title, fill_polygon)`

- Tách `path` thành danh sách x, y.
- Vẽ đường đi với x, y (màu đen, độ dày 0.5).
- Nếu `fill_polygon` = `true`:
  - Tô đa giác với màu tím nhạt.
- **Tìm các đỉnh góc 90 độ:**
  - Với mỗi điểm i từ 2 đến $n - 1$:
    - Nếu góc tại `path[i]` là 90 độ,
      thêm `path[i]` vào danh sách (right_angle_x, right_angle_y).
  - Kiểm tra góc tại điểm đầu/cuối (`path[1]` và `path[n-1]`).
- **Vẽ:**
  - Các đỉnh góc 90 độ (chấm đỏ).
  - Điểm đầu/cuối (ngôi sao đỏ).
- Trả về đối tượng plot.

Hàm `main()`

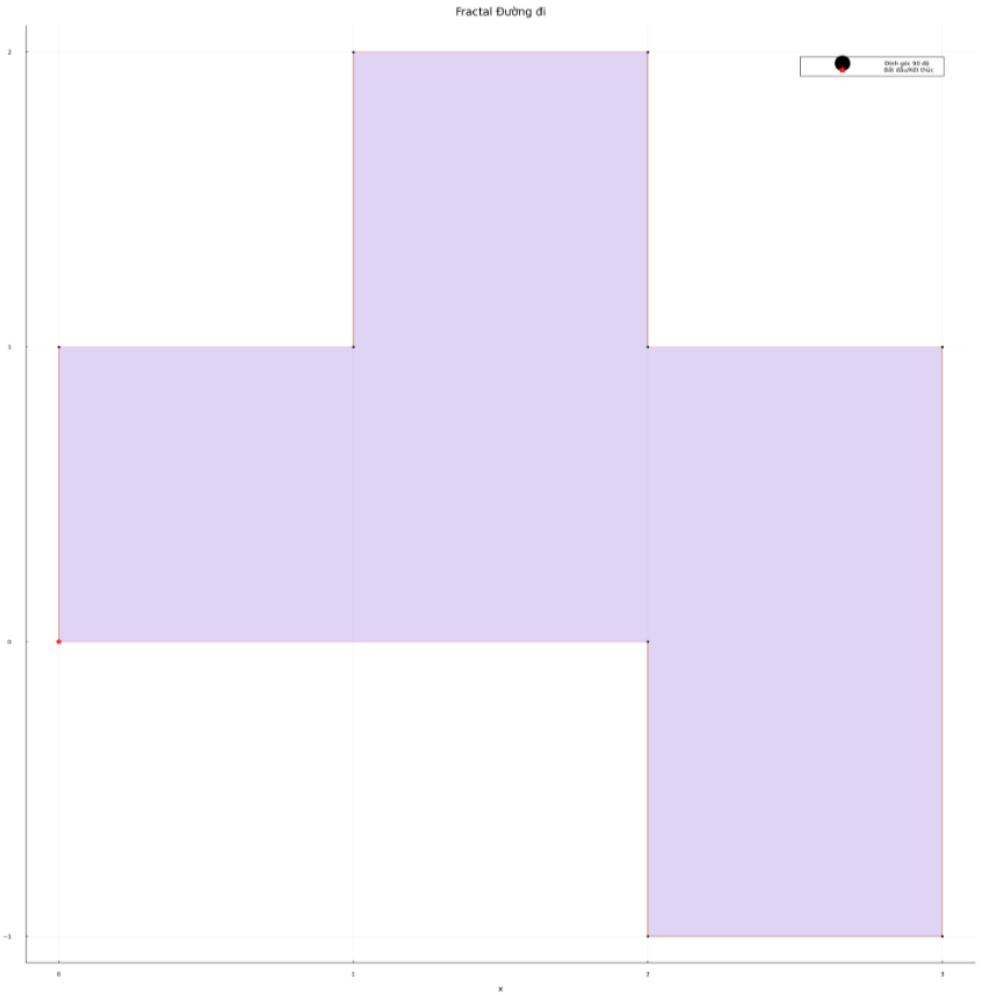
- Nhập số đỉnh góc 90 độ (`num_points`) từ người dùng.
- Nếu input không phải số nguyên, báo lỗi.
- Đo thời gian tổng.
- Gọi `generate_fractal_like_path(num_points)`.
- Đo thời gian tạo đường đi.
- Lưu đường đi vào file CSV `"fractal_$(num_points)_dinh.csv"`.
- Gọi `draw_path` với `fill_polygon = true`.
- Lưu hình vào `"fractal_$(num_points)_dinh.png"`.
- Đo thời gian vẽ.
- In các thông tin:
  - Thời gian tạo đường đi.
  - Tổng số điểm.
  - Thời gian vẽ.
  - Tổng thời gian.
- Xử lý lỗi nếu có.

III. Ví dụ (Đa giác trực giao với 10 đỉnh)

Thông tin:

- Tổng số điểm: **12**
- Thời gian tạo hình: **0.012 giây**

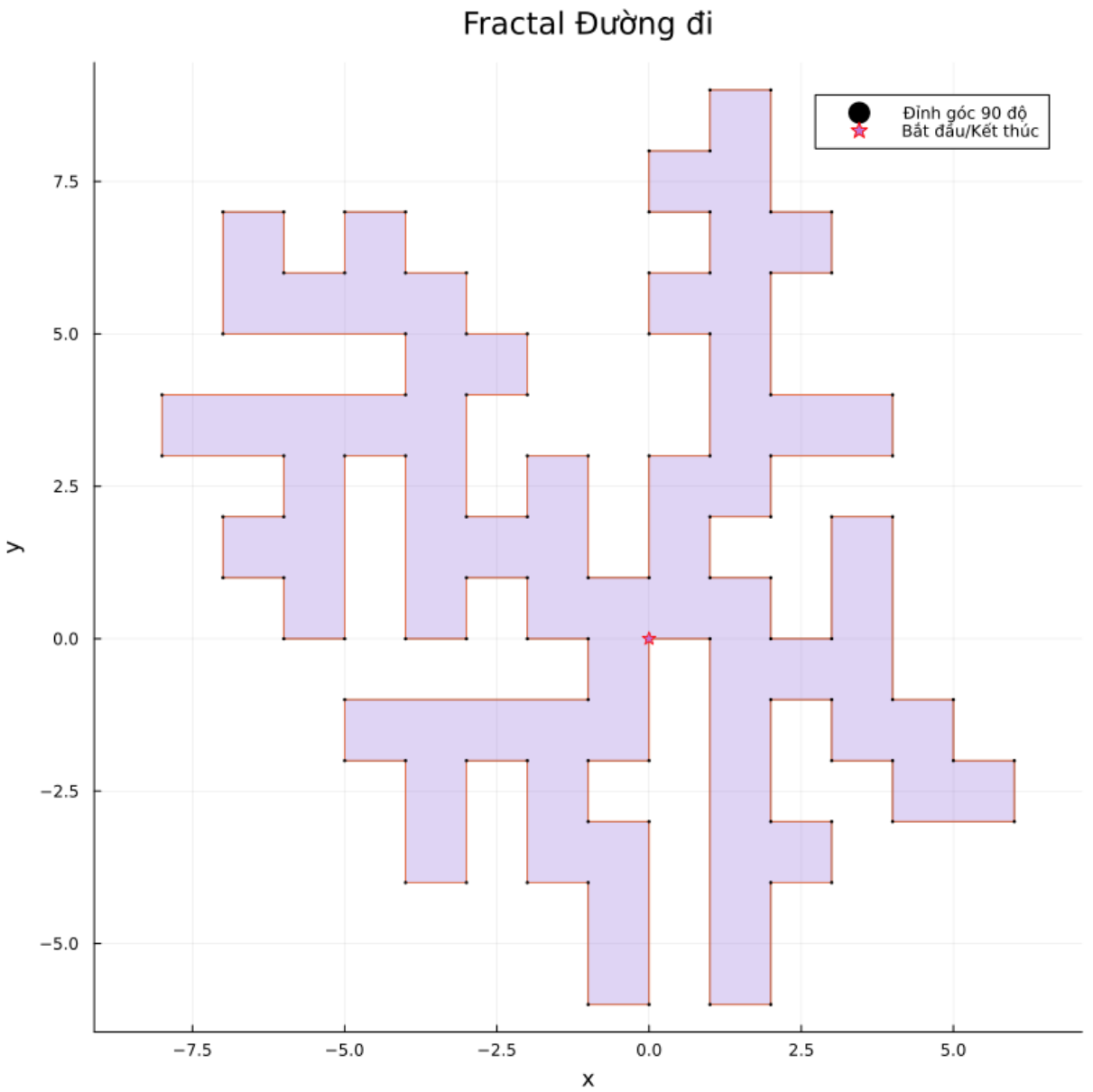
x	y
0	0
1	0
2	0
2	-1
3	-1
3	0
3	1
2	1
2	2
1	2
1	1
0	1
0	0



III. Ví dụ (Đa giác trực giao với 100 đỉnh)

Thông tin:

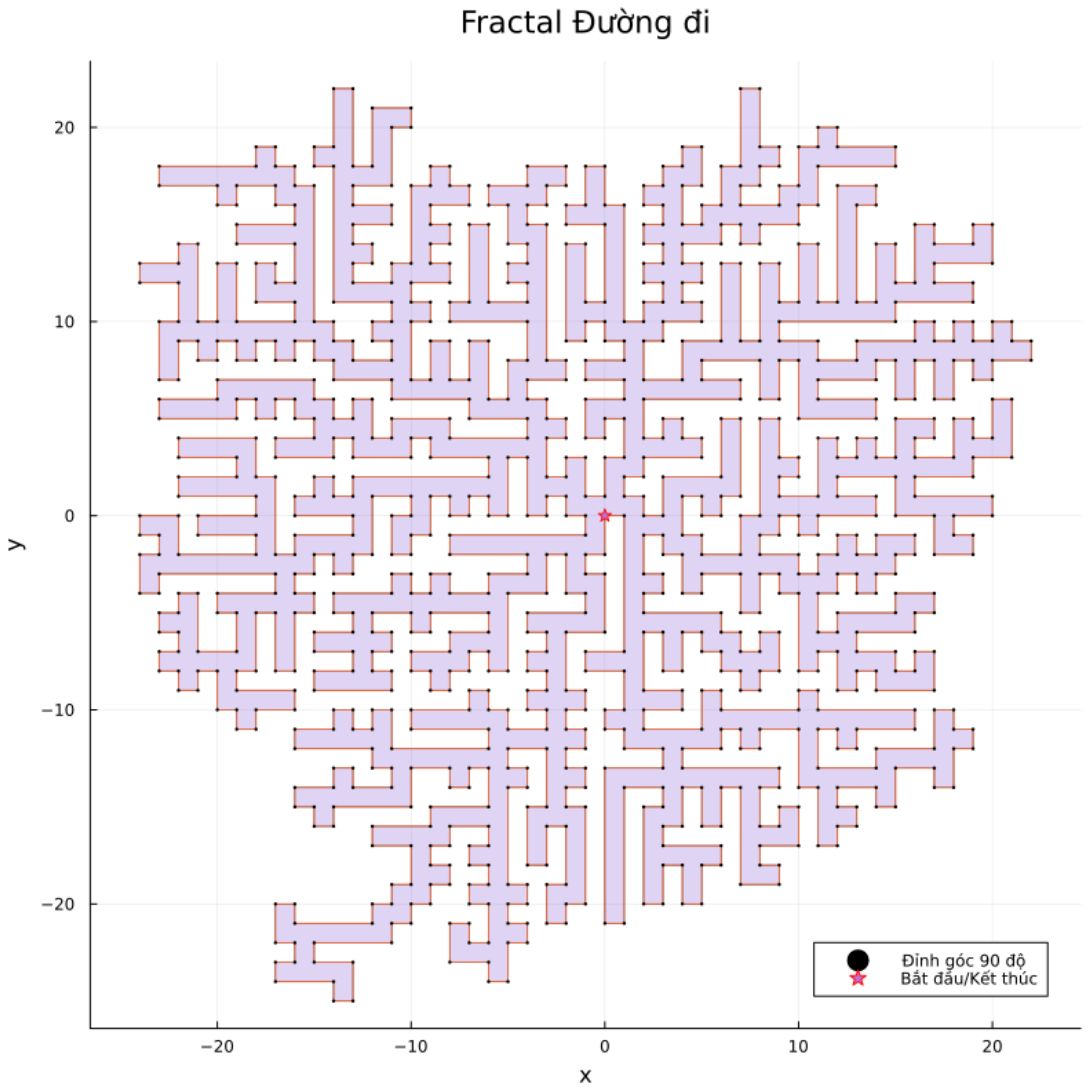
- Tổng số điểm: **141**
- Thời gian tạo hình: **0.69 giây**



III. Ví dụ (Đa giác trực giao với 1000 đỉnh)

Thông tin:

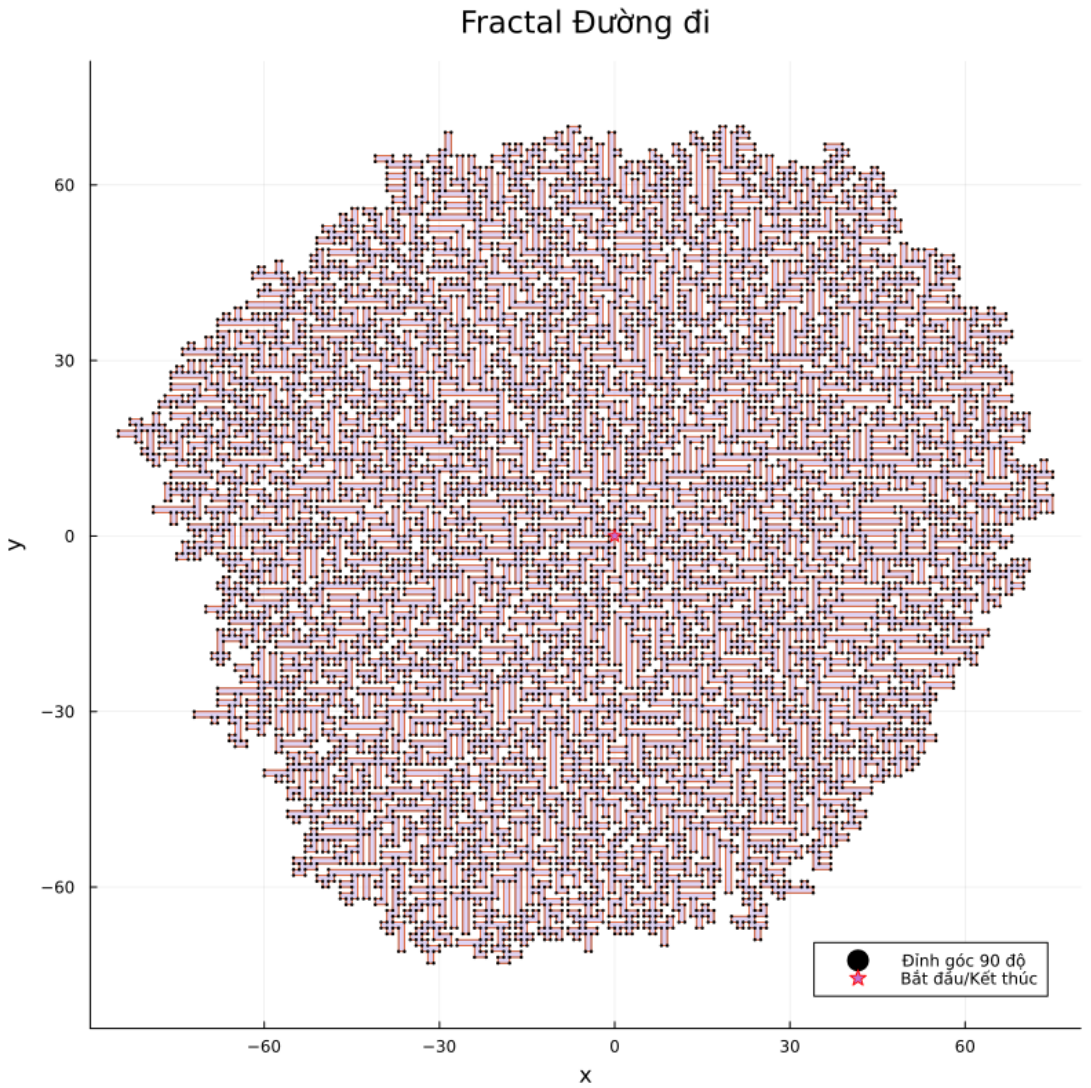
- Tổng số điểm: **1557**
- Thời gian tạo hình: **0.11 giây**



III. Ví dụ (Đa giác trực giao với 10000 đỉnh)

Thông tin:

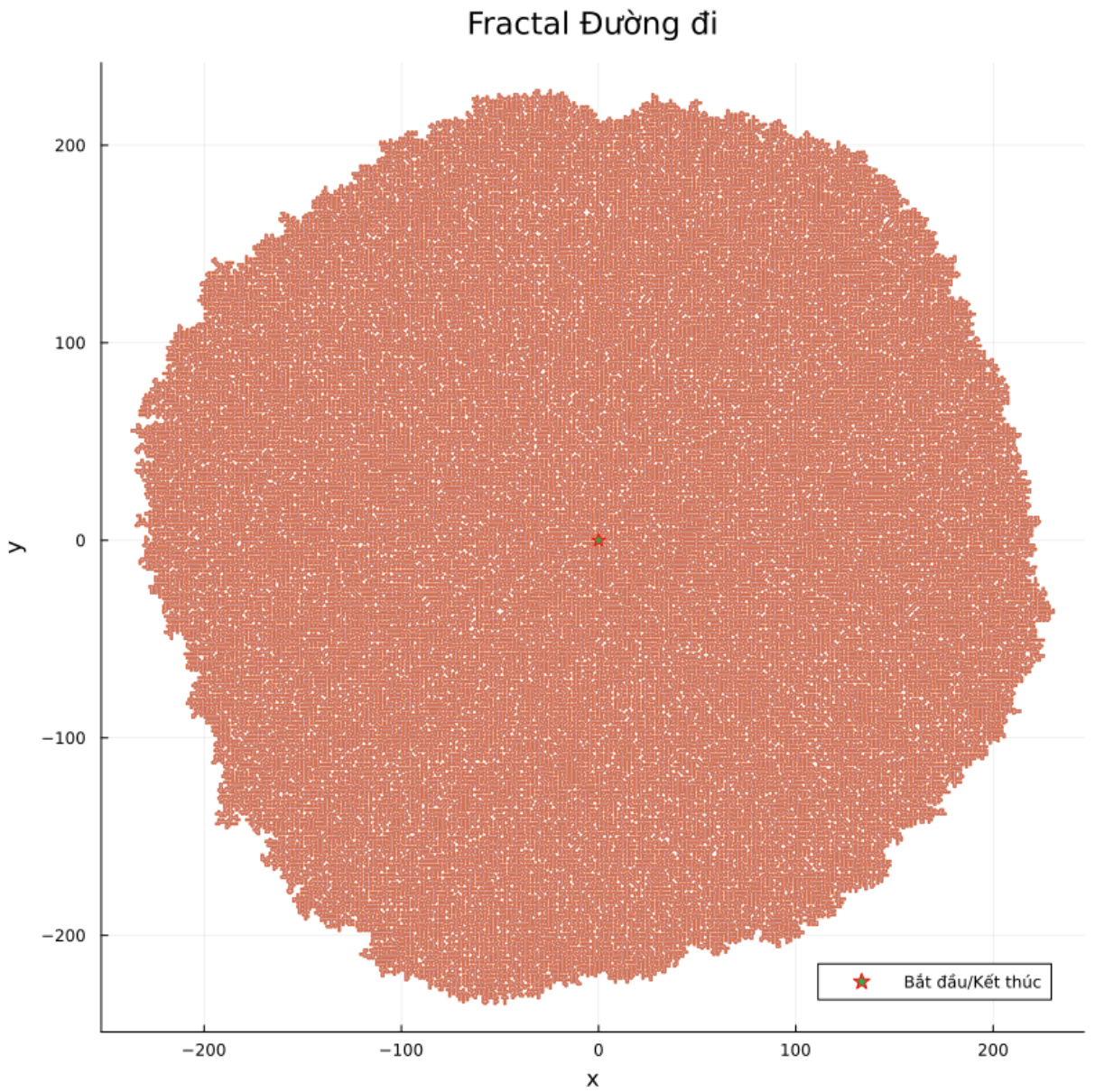
- Tổng số điểm: **15775**
- Thời gian tạo hình: **1.36 giây**



III. Ví dụ (Đa giác trực giao với 100000 đỉnh)

Thông tin:

- Tổng số điểm: **157.371**
- Thời gian tạo hình: **11.53 giây**



III. Ví dụ (Đa giác trực giao với 1000000 đỉnh)

Thông tin:

- Tổng số điểm: **1.570.385**
- Thời gian tạo hình: **797.57**