

## 1. Thuật toán DFS (Depth-First Search)

DFS (Depth-First Search) là một thuật toán duyệt hoặc tìm kiếm trên đồ thị (Graph) bằng cách bắt đầu từ một đỉnh và tiến sâu vào các đỉnh liền kề trước khi quay lại các đỉnh đã duyệt. DFS hoạt động theo cách "đi sâu" vào từng nhánh của đồ thị cho đến khi không thể tiếp tục, sau đó quay ngược lại và tìm các nhánh chưa được thăm.

-Cách hoạt động của DFS:

Bắt đầu từ một đỉnh bất kỳ (được gọi là đỉnh gốc).

Đánh dấu đỉnh hiện tại là đã thăm.

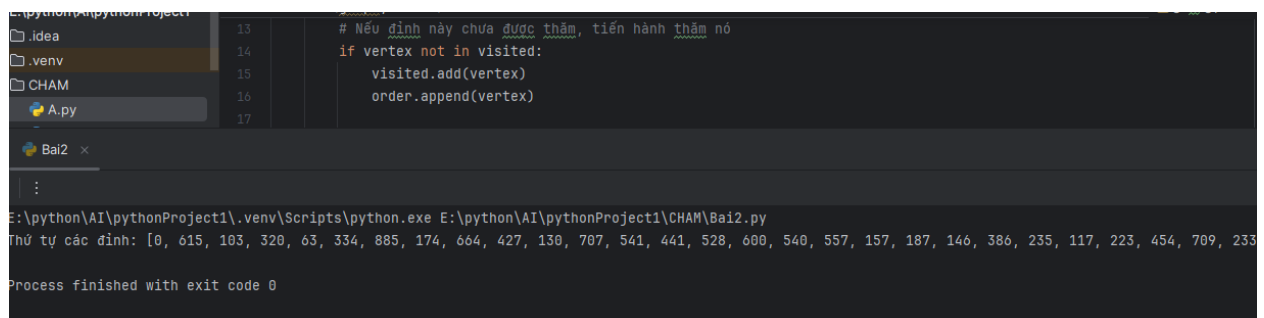
Lặp lại quá trình cho các đỉnh liền kề chưa được thăm.

Khi không còn đỉnh liền kề nào chưa được thăm, quay lại đỉnh trước đó (backtrack).

Tiếp tục quá trình cho đến khi tất cả các đỉnh có thể tới từ đỉnh gốc đã được thăm.

DFS có thể được cài đặt bằng cách sử dụng đệ quy (recursive) hoặc bằng cách sử dụng ngăn xếp (stack).

- Demo:



```
13 # Nếu đỉnh này chưa được thăm, tiến hành thăm nó
14 if vertex not in visited:
15     visited.add(vertex)
16     order.append(vertex)
17
```

Bai2 x

E:\python\AI\pythonProject1\.venv\Scripts\python.exe E:\python\AI\pythonProject1\CHAM\Bai2.py

Thứ tự các đỉnh: [0, 615, 103, 320, 63, 334, 885, 174, 664, 427, 130, 707, 541, 441, 528, 600, 540, 557, 157, 187, 146, 386, 235, 117, 223, 454, 709, 233]

Process finished with exit code 0

## 2. Thuật toán BFS (Breadth-First Search)

BFS (Breadth-First Search) là một thuật toán duyệt hoặc tìm kiếm trên đồ thị bằng cách bắt đầu từ một đỉnh và duyệt qua tất cả các đỉnh liền kề trước khi tiếp tục đến các đỉnh con của chúng. BFS hoạt động theo cách "đi rộng" từ đỉnh gốc, duyệt qua các đỉnh gần nhất trước khi tiến xa hơn.

-Cách hoạt động của BFS:

Bắt đầu từ một đỉnh bất kỳ (được gọi là đỉnh gốc).

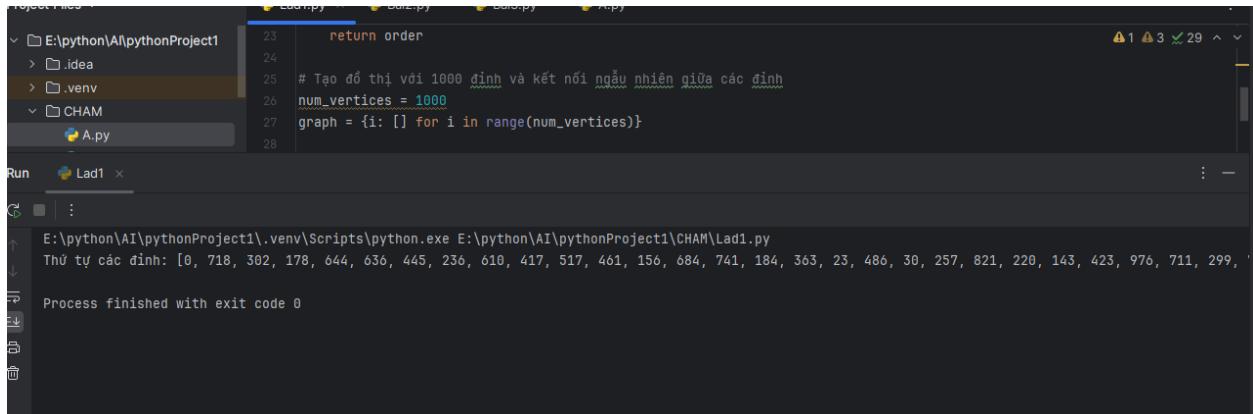
Đánh dấu đỉnh hiện tại là đã thăm.

Đưa tất cả các đỉnh liền kề vào một hàng đợi (queue).

Lấy đỉnh từ đầu hàng đợi và duyệt tiếp các đỉnh liền kề của nó.

Lặp lại quá trình cho đến khi tất cả các đỉnh có thể tới từ đỉnh gốc đã được thăm.

-Demo:



The screenshot shows an IDE with a Python script in the editor and its execution output in the console. The script defines a graph with 1000 vertices and returns an order. The console output shows the execution command and the resulting list of vertices.

```
23 return order
24
25 # Tạo đồ thị với 1000 đỉnh và kết nối ngẫu nhiên giữa các đỉnh
26 num_vertices = 1000
27 graph = {i: [] for i in range(num_vertices)}
28
```

Run: E:\python\AI\pythonProject1\.venv\Scripts\python.exe E:\python\AI\pythonProject1\CHAM\Lad1.py  
Thứ tự các đỉnh: [0, 718, 302, 178, 644, 636, 445, 236, 610, 417, 517, 461, 156, 684, 741, 184, 363, 23, 486, 30, 257, 821, 220, 143, 423, 976, 711, 299, ...]  
Process finished with exit code 0

### 3. Thuật toán Best first Search

Best-First Search (BFS) là một thuật toán tìm kiếm trên đồ thị hoặc cây, trong đó các đỉnh được mở rộng theo thứ tự dựa trên giá trị của một hàm đánh giá (heuristic). Thuật toán chọn đỉnh "tốt nhất" để mở rộng trước, dựa trên giá trị heuristic, với mục tiêu tiến gần nhất đến đích hoặc tối ưu hóa một tiêu chí nào đó.

- Cách hoạt động:

Bước 1: Khởi tạo một hàng đợi ưu tiên (priority queue) và thêm đỉnh bắt đầu vào với giá trị heuristic của nó.

Bước 2: Trong khi hàng đợi chưa trống, thực hiện:

Lấy đỉnh có giá trị heuristic thấp nhất từ hàng đợi ra.

Nếu đỉnh đó là đích, thuật toán dừng lại.

Nếu không, thêm tất cả các đỉnh kề của nó vào hàng đợi với giá trị heuristic tương ứng.

Bước 3: Thuật toán tiếp tục cho đến khi tìm thấy đích hoặc không còn đỉnh nào để mở rộng.

-Demo:

```
Run Bai3 x
Đang tham: 0
Đang thăm: 160
Đang thăm: 184
Đang thăm: 341
Đang thăm: 510
Đang thăm: 648
Đang thăm: 689
Đang thăm: 743
Đang thăm: 844
Đang thăm: 979
Đang thăm: 81
Đang thăm: 96
Đang thăm: 106
Đang thăm: 192
Đang thăm: 119
Đang thăm: 202
Đang thăm: 220
Đang thăm: 327
Đang thăm: 681
Đang thăm: 720
Đang thăm: 727
Đã tìm thấy mục tiêu: 728
Process finished with exit code 0
```

#### 4. Thuật toán A\*

A\* (A-star) là một thuật toán tìm kiếm thông minh, được sử dụng chủ yếu để tìm đường đi tối ưu trong đồ thị và trong các bài toán tìm kiếm trong không gian trạng thái. A\* kết hợp các yếu tố của cả thuật toán Dijkstra và Best-First Search, sử dụng một hàm đánh giá (heuristic) để xác định đỉnh nào nên được mở rộng tiếp theo.

-Cách hoạt động:

Khởi tạo: Bắt đầu từ đỉnh nguồn, thuật toán lưu trữ các đỉnh cần thăm trong một hàng đợi ưu tiên (priority queue).

Lặp lại: Trong khi hàng đợi chưa rỗng:

Lấy đỉnh có giá trị f thấp nhất ra khỏi hàng đợi.

Nếu đỉnh này là đỉnh đích, thuật toán dừng lại và trả về đường đi.

Nếu không, đánh dấu đỉnh đó là đã thăm và thêm các đỉnh kề vào hàng đợi với giá trị f được tính toán.

Kết thúc: Quá trình này tiếp tục cho đến khi tìm thấy đỉnh đích hoặc không còn đỉnh nào để mở rộng.

-Nguyên tắc hoạt động:

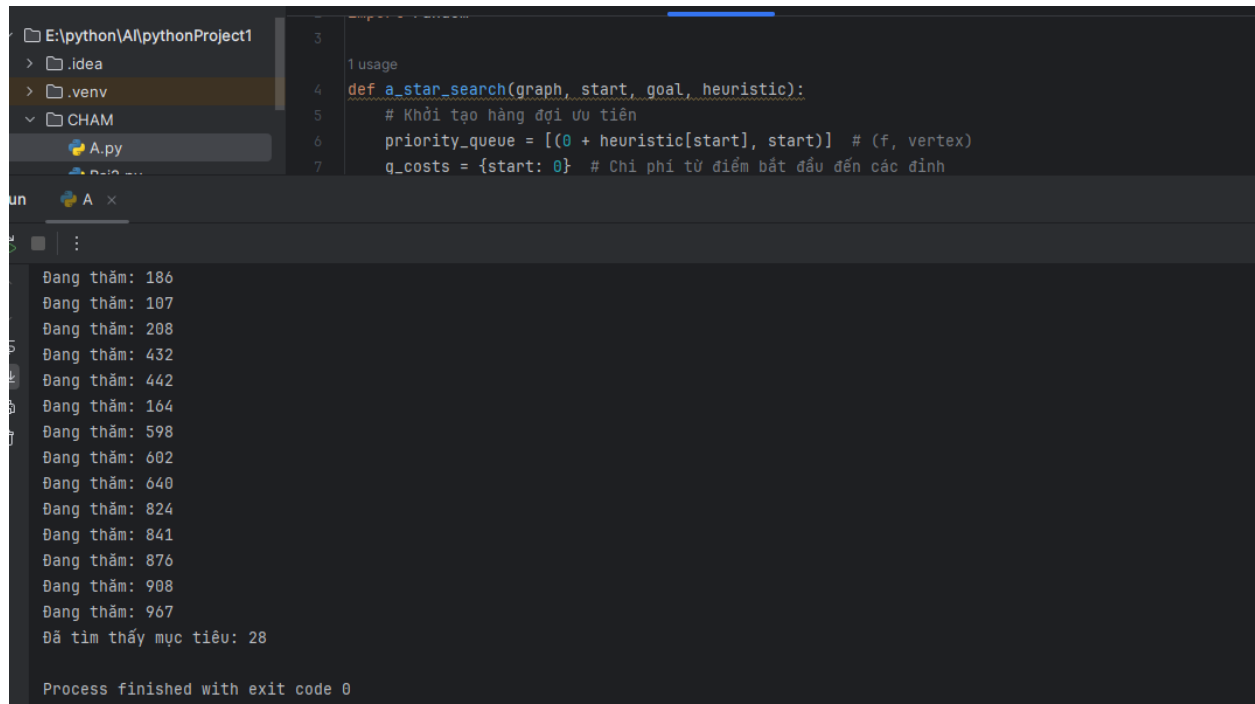
A\* sử dụng một hàm đánh giá  $f(n)$  để tính toán giá trị của mỗi đỉnh, trong đó:

$$f(n) = g(n) + h(n)$$

$g(n)$ : Chi phí từ đỉnh bắt đầu đến đỉnh hiện tại  $n$ .

$h(n)$ : Hàm heuristic ước lượng chi phí từ đỉnh hiện tại  $n$  đến đỉnh đích. Nguyên tắc hoạt động:

-Demo:



The screenshot shows a Python IDE with a project named 'E:\python\AI\pythonProject1'. The file explorer on the left shows folders '.idea', '.venv', and 'CHAM', and a file 'A.py'. The main editor displays the following Python code:

```
1 usage
2
3
4 def a_star_search(graph, start, goal, heuristic):
5     # Khởi tạo hàng đợi ưu tiên
6     priority_queue = [(0 + heuristic[start], start)] # (f, vertex)
7     q_costs = {start: 0} # Chi phí từ điểm bắt đầu đến các đỉnh
```

The output console at the bottom shows the execution results:

```
Đang thăm: 186
Đang thăm: 107
Đang thăm: 208
Đang thăm: 432
Đang thăm: 442
Đang thăm: 164
Đang thăm: 598
Đang thăm: 602
Đang thăm: 640
Đang thăm: 824
Đang thăm: 841
Đang thăm: 876
Đang thăm: 908
Đang thăm: 967
Đã tìm thấy mục tiêu: 28
Process finished with exit code 0
```