**CƠ SỞ TRÍ TUỆ NHÂN TẠO**

|  |
| --- |
| **ĐỒ ÁN 1 - TÌM KIẾM**  **YÊU CẦU 1 - TRIỂN KHAI CÁC CHIẾN LƯỢC TÌM KIỀM TRÊN ĐỒ THỊ** |

* **GV: LÊ HOÀI BẮC**
* **GV: LÊ NGỌC THÀNH**
* **GV: NGUYỄN NGỌC THẢO**
* **GV: NGUYỄN HẢI MINH**

**Sinh viên thực hiện:**

|  |  |
| --- | --- |
| * **Họ tên:** * **MSSV:** * **Email:** * **SĐT:** | * **Cao Nguyễn Minh Hiếu** * **1512157** * 1512157@students.hcmus.edu.vn * **0165.259.2239** |

1. **Các lưu ý để chạy chương trình thành công:**

\_ Chương trình được viết bằng ngôn ngữ Python, phiên bản Python 3.6 - 64bit, và chỉ gói gọn trong một file duy nhất là *main.py*.

\_ Source code của Yêu cầu 1 này không import thư viện nào cả, nên chỉ cần Thầy/Cô có cài đặt Python trên máy là có thể biên dịch và chạy được chương trình.

\_ Để chạy chương trình, Thầy/Cô mở tập tin *main.py* trong thư mục Yêu cầu 1 bằng IDE của Python và chạy.

\_ Python 3.6 tải tại: <https://www.python.org/downloads/>

1. **Cấu trúc dữ liệu lưu trữ đồ thị:**
2. **Cấu trúc tập tin input.txt:**

● Chương trình đọc input từ tập tin *input.txt*, đặt cùng cấp với tập tin *main.py*.

*► Tất cả chiến lược đều sử dụng chung tập tin input này để minh họa.*

● Trong tập tin *input.txt*:

\_ Dòng đầu tiên chứa một số nguyên n (n ≥ 3) thể hiện số đỉnh của đồ thị.  
 \_ Dòng thứ hai chứa hai số nguyên biểu diễn chỉ mục của đỉnh nguồn và đỉnh đích, biết rằng chỉ mục của đỉnh nằm trong đoạn [0, n-1].  
 \_ n dòng tiếp theo chứa n số nguyên trên mỗi dòng để biểu diễn ma trận kề, các số nguyên cách nhau bằng khoảng trắng. Gọi [i, j] là giá trị tại dòng i cột j (i, j = 0,…, n- 1). [i, j] = 0: không có cạnh nối từ đỉnh i đến đỉnh j, [i, j] = x > 0: có cạnh nối từ i đến j với trọng số x.  
 \_ Dòng cuối cùng chứa n nguyên không âm để biểu diễn giá trị heuristic tương ứng cho các đỉnh từ 0 đến n-1, các số nguyên cách nhau bằng khoảng trắng. Giá trị heuristic đã được đảm bảo điều kiện chấp nhận được.

1. **Cấu trúc dữ liệu lưu trữ đồ thị trong chương trình:**

\_ Dữ liệu trong mã nguồn được tổ chức dưới dạng ma trận nxn, với n là số đỉnh của đồ thị.

\_ Giá trị tại dòng thứ i, cột thứ j cho biết trọng số của đường đi từ đỉnh i đến đỉnh j của đồ thị. Giá trị này bằng 0 nếu không có đường đi từ đỉnh i đến đỉnh j.

\_ Giá trị heuristic của mỗi đỉnh được lưu dưới dạng mảng một chiều (trong Python là kiểu dữ liệu *list*). Phần tử thứ i của mảng lưu giá trị heuristic của đỉnh thứ i trong đồ thị.

1. **Cấu trúc các tập tin output:**

● Mỗi chiến lược tìm kiếm cho ra một tập tin output khác nhau, bao gồm: *dfs.txt*, *bfs.txt*, *ucs.txt*, *gbfs.txt* và *astar.txt*, được lưu cùng cấp với file *input.txt*.

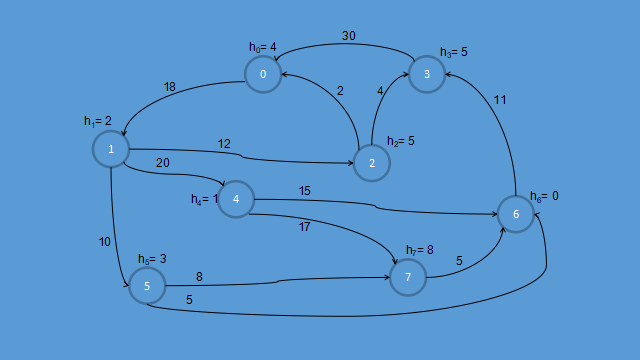
● Các tập tin output trên đều có cùng một định dạng:

\_ Dòng đầu tiên ghi danh sách các đỉnh đã được duyệt trong quá trình tìm kiếm, mỗi đỉnh cách nhau một khoảng trắng

\_ Dòng thứ 2 ghi danh sách các đỉnh mà chiến lược tương ứng cho rằng đường đi qua các đỉnh đó là ngắn nhất, mỗi đỉnh cách nhau một khoảng trắng.

1. **Các chiến lược tìm kiếm được cài đặt và ví dụ minh họa:**

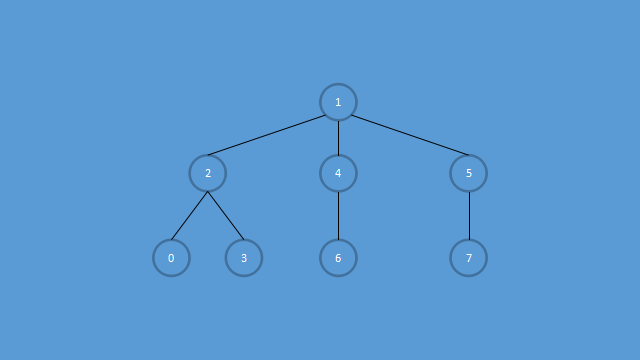
❖ Đồ thị trong tập tin input.txt được visualize như sau:



\_ Đỉnh start = 1, đỉnh Goal = 6.

1. **Tìm kiếm theo chiều sâu (DFS):**

\_ Cây đường đi được xây dựng từ đồ thị như sau:



\_ Các đỉnh được duyệt theo chiến lược DFS là : 1 → 2 → 0 → 3 → 4 → 6

\_ Đường đi là: 1 → 4 → 6

1. **Tìm kiếm theo chiều rộng (BFS):**

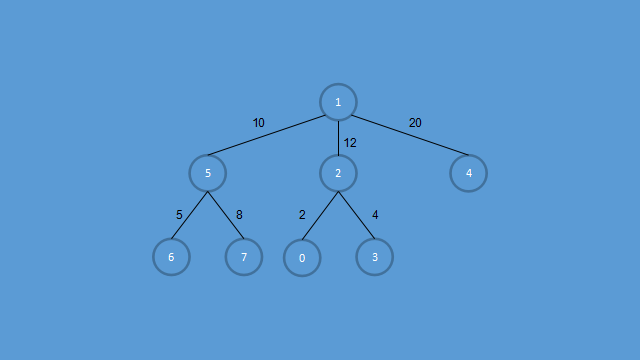
\_ Cây đường đi được xây dựng tương tự như chiến lược DFS.

\_ Các đỉnh được duyệt theo chiến lược BFS là : 1 → 2 → 4 → 5 → 0 → 3 → 6

\_ Đường đi là: 1 → 4 → 6

1. **Tìm kiếm chi phí đồng nhất (UCS):**

\_ Chiến lược này có tính đến trọng số của các đường đi, nên để dễ quan sát, cây đường đi được vẽ lại có đôi chút khác với 2 chiến lược trên:



\_ Thứ tự đẩy vào và lấy phần tử ra khỏi hàng đợi ưu tiên được minh họa theo bảng sau:

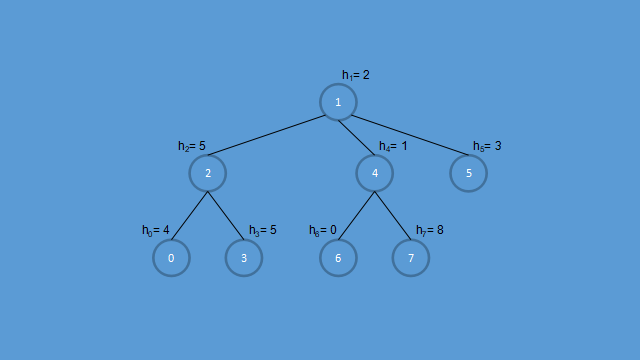
|  |  |  |  |
| --- | --- | --- | --- |
| Bước | Hàng đợi | Đỉnh được lấy ra | Đỉnh được mở |
| 0 | (1, 0) | (1, 0) | (5, 10) , (2, 12) , (4, 20) |
| 1 | (5, 10) , (2, 12) , (4, 20) | (5, 10) | (6, 15) , (7, 18) |
| 2 | (2, 12) , (4, 20) , (6, 15) , (7, 18) | (2, 12) | (0, 14) , (3, 16) |
| 3 | (0, 14) , (6, 15) , (3, 16) , (7, 18) , (4 20) | (0, 14) | *// mở đỉnh 1 nhưng 1 đã duyệt* |
| 4 | (6, 15) , (3, 16) , (7, 18) , (4 20) | (6, 15) | Dừng |

\_ Các đỉnh được duyệt theo chiến lược UCS là: 1 → 5 → 2 → 0 → 6

\_ Đường đi là: 1 → 5 → 6

1. **Tìm kiếm tham lam (GBFS):**

\_ Chiến lược Tham lam dựa vào giá trị heuristic của mỗi đỉnh để chọn nước đi tiếp theo:



\_ Thứ tự các đỉnh được đưa vào và lấy ra khỏi ngăn xếp được minh họa theo bảng sau:

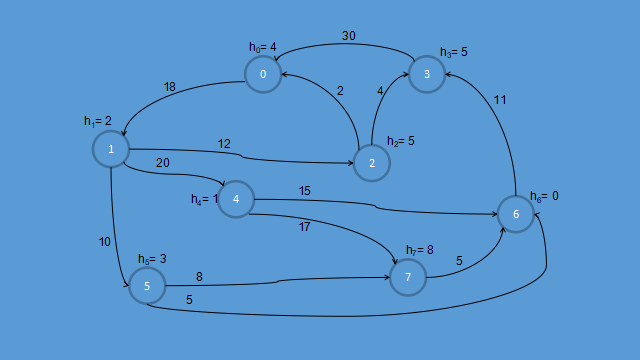
|  |  |  |  |
| --- | --- | --- | --- |
| Bước | Ngăn xếp | Đỉnh được lấy ra | Đỉnh được mở |
| 0 | (1 , 2) | (1, 2) | (2, 5) , (4, 1) , (5, 3) |
| 1 | (2, 5) , (5, 3) , (4, 1) | (4, 1) | (6, 0) , (7, 8) |
| 2 | (2, 5) , (5, 3) , (7, 8) , (6, 0) | (6, 0) | Dừng |

\_ Các đỉnh được duyệt theo chiến lược GBFS là: 1 → 4 → 6

\_ Đường đi là: 1 → 4 → 6

1. **Tìm kiếm A\* (ASS):**

\_ Tìm kiếm A\* sử dụng hàm f(x) = g(x) + h(x) , với g(x) là chi phí từ nút start tới nút x (được sử dụng trong chiến lược UCS) , và h(x) là giá trị heuristic của nút x (được sử dụng trong chiến lược GBFS).



\_ Thứ tự các đỉnh được đưa và và lấy ra hàng đợi ưu tiên được minh họa theo bảng sau ( Mỗi đỉnh có dạng ( chỉ số đỉnh x, g(x), h(x) ) )

|  |  |  |  |
| --- | --- | --- | --- |
| Bước | Hàng đợi ưu tiên | Đỉnh được lấy ra | Đỉnh được mở |
| 0 | (1 , 0, 2) | (1, 0, 2) | (2, 12, 5) , (4, 20, 1) , (5, 10, 3) |
| 1 | (5, 10, 3) , (2, 12, 5) , (4, 20, 1) | (5, 10, 3) | (6, 15, 0) , (7, 18, 8) |
| 2 | (6, 15, 0) , (2, 12, 5) , (4, 20, 1) , (7, 18, 8) | (6, 15, 0) | Dừng |

\_ Các đỉnh được duyệt theo chiến lược UCS là: 1 → 5 → 6

\_ Đường đi là: 1 → 5 → 6

-------------------- HẾT --------------------