KHOA CÔNG NGHỆ THÔNG TIN
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN,
ĐẠI HỌC QUỐC GIA TP HỒ CHÍ MINH

MÔN CƠ SỞ TRÍ TUỆ NHÂN TẠO

Sinh viên thực hiện: 22120054 - Lê Văn Thành Đạt
GV phụ trách: Nguyễn Thị Thu Hằng

CƠ SỞ TRÍ TUỆ NHÂN TẠO
HỌC KỲ I – NĂM HỌC 2024-2025

Khoa Công nghệ thông tin
Đại học Khoa học tự nhiên TP HCM

Mục lục

I) Student info

- Mã số sinh viên: 22120054
- Họ và tên: Lê Văn Thành Đạt

II) Completion Level of Each Requirement

- 10/10

III) Presentation of Basic Theories

1) Breadth-first Search
    - Khái niệm: Breadth-first Search là chiến lược tìm kiếm đơn giản, duyệt đồ thị theo chiều rộng, tất cả các nút ở 1 độ sâu nhất định phải được mở trước khi mở các nút ở độ sâu tiếp theo. BFS thường được dùng để tìm đường đi ngắn nhất trong đồ thị không trọng số.
    - Độ phức tạp:
        - Thời gian: O(V+E) với V là số đỉnh còn E là số cạnh
        - Không gian: O(V) do sử dụng hàng đợi để lưu trữ các đỉnh chờ duyệt
    - Tính chất:
        - Complete: có (nếu b hữu hạn)
        - Optimal: chi phí di chuyển như nhau
        - Tìm kiếm không có trọng số.
        - Đảm bảo tìm ra đường đi ngắn nhất trong đồ thị không trọng số.
2) Depth-first Search
    - Khái niệm: Depth-first Search - DFS (tìm kiếm theo chiều sâu) mở các nút sâu nhất trước. DFS duyệt sâu vào các nhánh trước khi quay lại để duyệt các nhánh khác

- Độ phức tạp:
  - Thời gian: O(V + E).
  - Không gian: O(V + E), sử dụng ngăn xếp để lưu trữ các đỉnh.
- Tính chất:
  - Complete: có (nếu không gian hữu hạn)
  - Optimal: lời giải "phải nhất"
  - Có thể không tìm được đường đi ngắn nhất.
  - Hữu ích trong việc kiểm tra tính liên thông, phát hiện chu trình

3) Uniform-cost Search
   - Khái niệm: UCS là thuật toán tìm kiếm trong đồ thị có trọng số, tìm kiếm đường đi có chi phí thấp nhất từ đỉnh nguồn đến đỉnh đích. UCS chọn các đỉnh có chi phí thấp nhất và mở rộng dần.
   - Độ phức tạp:
     - Thời gian: $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$
     - Không gian: $O(b^{1+\lfloor C^*/\varepsilon \rfloor})$
   - Tính chất:
     - Complete: có
     - Optimal: có
     - Đảm bảo tìm ra đường đi ngắn nhất trong đồ thị có trọng số dương.
     - Không phụ thuộc vào độ sâu của đỉnh đích.
4) Greedy Best-first Search
   - Khái niệm: Greedy Best-first Search là thuật toán tìm kiếm chọn giải pháp cục bộ tối ưu tại mỗi bước nhằm hướng tới giải pháp tối ưu toàn cục.
   - Độ phức tạp:
     - Thời gian: Phụ thuộc vào bài toán và cách cài đặt, thường từ O(V^2) đến O(V + E).
     - Không gian: Tùy thuộc vào cấu trúc dữ liệu được sử dụng.

- Tính chất:
  - Dễ cài đặt, tốc độ nhanh trong nhiều trường hợp.
  - Không đảm bảo tìm ra giải pháp tối ưu toàn cục.

5) A* Search
   - Khái niệm: A* Search là dạng tìm kiếm Best-first Search phổ biến nhất. A* là sự kết hợp của UCS và Greedy Best-first Search, đảm bảo tìm ra đường đi tối ưu nếu hàm heuristic thỏa mãn tính khả thi.
   - Độ phức tạp:
     - Thời gian: O(E), tùy thuộc vào độ chính xác của hàm heuristic.
     - Không gian: O(V), lưu trữ đỉnh đã duyệt và hàng đợi ưu tiên.
   - Tính chất:
     - Complete: có nếu chi phí di chuyển thấp nhất > 0 và không gian trạng thái hữu hạn
     - Optimal: có nếu heuristic hợp lí và nhất quán
     - Đảm bảo tìm ra đường đi ngắn nhất nếu hàm heuristic không đánh giá quá cao.
     - Độ chính xác phụ thuộc vào hàm heuristic.

IV) Comparison of UCS and A* Algorithms

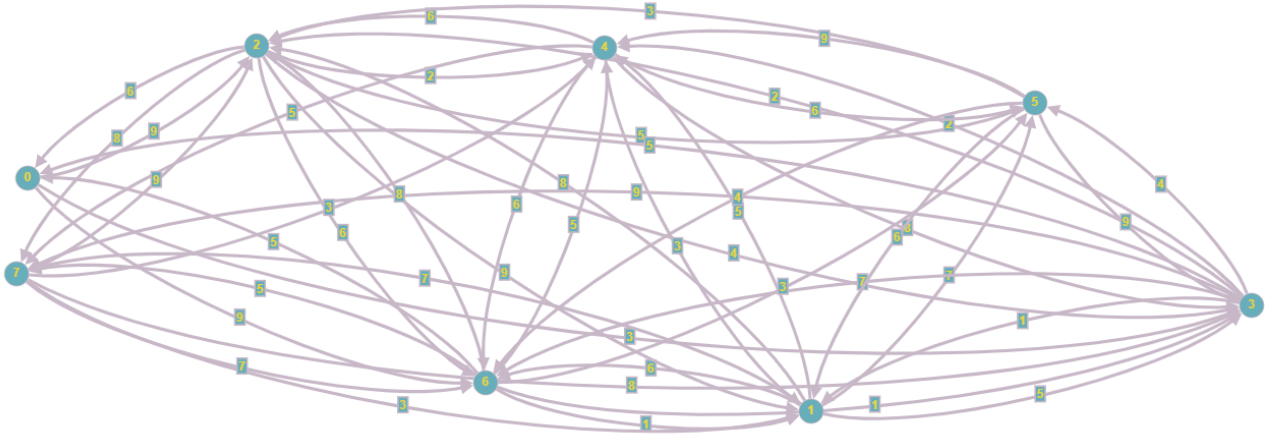
Tiêu chí	Uniform Cost Search	A* Search
Chiến lược	Mở nút dựa trên chi phí thực tế từ đỉnh bắt đầu đến đỉnh hiện tại	Mở nút dựa trên chi phí thực tế và chi phí ước lượng
Ưu tiên	Ưu tiên mở nút có chi phí thực tế thấp nhất từ đỉnh bắt đầu đến đỉnh hiện tại	Ưu tiên mở nút có chi phí g(n) + h(n) thấp nhất

Optimal	Đảm bảo tìm được đường đi ngắn nhất trong đồ thị có trọng số không âm	Đảm bảo tối ưu nếu hàm heuristic thỏa mãn tính khả thi và nhất quán (tối ưu với h không vượt quá chi phí thực tế).
Ứng dụng	Tìm đường đi ngắn nhất trong đồ thị không âm	Điều hướng bản đồ, AI trong trò chơi, robot tự hành…
Ưu điểm	Thích hợp cho các đồ thị trọng số dương chưa có hàm heuristic phù hợp	Có thể tìm kiếm nhanh hơn UCS khi hàm heuristic tốt
Nhược điểm	Không tối ưu thời gian do kiểm tra nhiều đỉnh	Hiệu quả phụ thuộc vào hàm heuristic

V) Test case

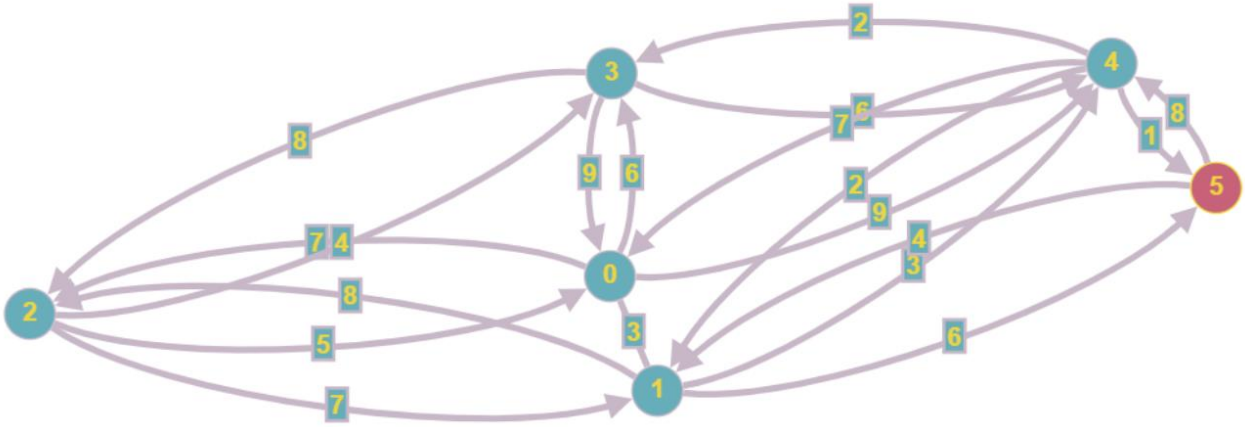
- Input01.txt:

```
16
00930090
00855767
69042568
51202479
03680665
06399040
51815305
03983070
```



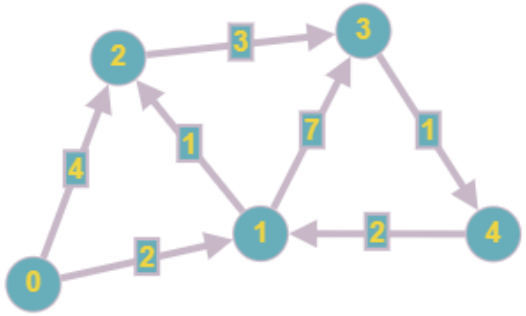
- Input02.txt:

```
4 0
0 3 7 6 9 0
3 0 8 0 3 6
5 7 0 4 0 0
9 0 8 0 6 0
7 2 0 2 0 1
0 4 0 0 8 0
```



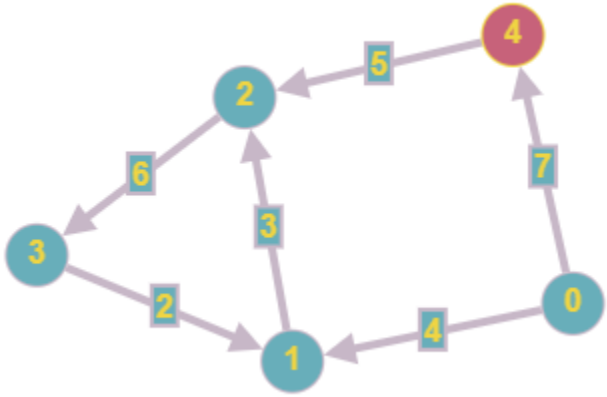
- Input03.txt:

```
0 4
0 2 4 0 0
0 0 1 7 0
0 0 0 3 0
0 0 0 0 1
0 2 0 0 0
```



- Input04.txt:

```
0 4
0 4 0 0 7
0 0 3 0 0
0 0 0 6 0
0 2 0 0 0
0 0 5 0 0
```



- Input05.txt:

```
0 6
0 2 0 0 4 0 0
2 0 3 8 0 0 0
0 3 0 0 5 0 7
0 8 0 0 0 6 0
4 0 5 0 0 1 0
0 0 0 6 1 0 9
0 0 7 0 0 9 0
```

VI) Tài liệu tham khảo

- Geeksforgeeks
- ChatGPT
- Graphonline