BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

ĐỒ ÁN
HỌC PHẦN TRÍ TUỆ NHÂN TẠO

Đề tài
CÀI ĐẶT CÁC THUẬT TOÁN TÌM ĐƯỜNG ĐI TRÊN BẢN ĐỒ

Giảng viên hướng dẫn:

TS. Lưu Tiến Đạo

Nhóm sinh viên thực hiện:

1. Huỳnh Thanh Phong –B2207555
2.Trịnh Quốc Kiên – B2207534
3.Nguyễn Hoàng Tuấn – B2013571

Cần Thơ, 10/2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ
TRƯỜNG CÔNG NGHỆ THÔNG TIN & TRUYỀN THÔNG

ĐỒ ÁN
HỌC PHẦN TRÍ TUỆ NHÂN TẠO

Đề tài
CÀI ĐẶT CÁC THUẬT TOÁN TÌM ĐƯỜNG ĐI TRÊN BẢN ĐỒ

Giảng viên hướng dẫn:

TS. Lưu Tiến Đạo

Nhóm sinh viên thực hiện:

1. Huỳnh Thanh Phong –B2207555
2. Trịnh Quốc Kiên – B2207534
3. Nguyễn Hoàng Tuấn – B2013571

Cần Thơ, 10/2024

NHẬN XÉT CỦA GIẢNG VIÊN

Cần Thơ, ngày tháng năm
(Ký và ghi rõ họ tên)

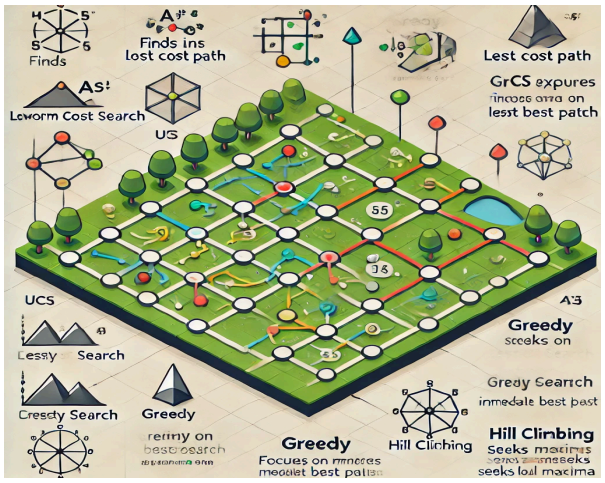
MỤC LỤC

PHÂN CÔNG CÔNG VIỆC

Họ và tên - MSSV	Nhiệm vụ phân công	Phần trăm hoàn thành
1. Huỳnh Thanh Phong – B2207555	- Cài đặt thuật toán. - Xây dựng web. - So sánh và phân tích hiệu suất các thuật toán	100%
2.Trịnh Quốc Kiên – B2207534	- Hoàn thành báo cáo tổng hợp (word, powerpoint, …)	100%
3.Nguyễn Hoàng Tuấn – B2013571	- Phân tích đề tài, thiết kế hệ thống. - Xây dựng web. - Cài đặt thuật toán. - Kiểm tra sửa lỗi tổng thể hệ thống và các file báo cáo.	100%

PHẦN NỘI DUNG

1. Mô tả bài toán



Bài toán đặt ra là tìm đường đi trên một đồ thị dựa theo thuật toán đã chọn, với các đỉnh và cạnh có trọng số. Người dùng sẽ nhập vào điểm bắt đầu (start), điểm kết thúc (end), và lựa chọn một trong bốn giải thuật: A*, Greedy, UCS (Uniform Cost Search), hoặc Hill Climbing để tìm đường. Mỗi giải thuật sẽ sử dụng phương pháp riêng để tìm kiếm, giúp người dùng tìm ra con đường đến đích.

Hệ thống sẽ hiển thị thứ tự duyệt của từng giải thuật để người dùng dễ dàng theo dõi quá trình tìm đường.

2. Phân tích bài toán

Để giải quyết bài toán tìm đường đi trên bản đồ, hệ thống cần quản lý một đồ thị với các đỉnh và cạnh có trọng số. Mỗi cạnh thể hiện một con đường giữa hai địa điểm, và trọng số là chi phí của con đường đó. Người dùng sẽ tương tác với hệ thống thông qua giao diện, tải lên file đầu vào mô tả đồ thị, sau đó chọn điểm bắt đầu, điểm kết thúc, và thuật toán tìm kiếm.

- Input của bài toán là đồ thị, đỉnh bắt đầu, và đỉnh kết thúc.
- Output là đường đi từ đỉnh bắt đầu đến đỉnh kết thúc, hoặc thông báo không tìm thấy đường đi.

3. Cơ sở lý thuyết (các giải thuật được sử dụng trong đồ án)

3.1 A* Search

Giải thuật A* là giải thuật tìm kiếm tối ưu sử dụng hàm đánh giá tổng hợp **f(n)=g(n)+h(n)**, trong đó:

- **g(n):** Chi phí từ đỉnh bắt đầu đến đỉnh hiện tại n.
- **h(n):** Ước lượng chi phí từ n đến đỉnh đích (hàm heuristic).

Mã giả mô tả thuật toán:

1. Khởi tạo:

- Đặt tập mở (open set) gồm đỉnh bắt đầu (start).
- Khởi tạo tập đóng (closed set) là rỗng.
- Thiết lập g(start) = 0 (chi phí từ điểm bắt đầu đến đỉnh hiện tại).
- Thiết lập h(start) bằng giá trị ước lượng chi phí từ đỉnh bắt đầu đến đích.

- Thiết lập f(start) = g(start) + h(start).

2. Lặp lại cho đến khi tập mở rỗng:

a. Chọn đỉnh n từ tập mở có giá trị f(n) nhỏ nhất.

b. Nếu n là đỉnh đích, trả về đường đi từ start đến n.

c. Di chuyển n từ tập mở sang tập đóng.

d. Với mỗi đỉnh kề m của n:

- Nếu m nằm trong tập đóng, bỏ qua đỉnh này.
- Tính toán g(m) = g(n) + chi phí(n, m).
- Tính h(m) = ước lượng chi phí từ m đến đích.
- Tính f(m) = g(m) + h(m).
- Nếu m chưa nằm trong tập mở hoặc g(m) hiện tại nhỏ hơn g(m) trước đó:
  - Cập nhật parent của m thành n.
  - Nếu m không nằm trong tập mở, thêm m vào tập mở.

3. Nếu không tìm thấy đường đi, trả về "không có đường đi".

Đặc điểm:

Đảm bảo tìm được giải pháp tối ưu: A* sẽ tìm được đường đi tối ưu nếu hàm heuristic là admissible và consistent.

- **Admissible**: Hàm không ước lượng quá cao chi phí thực tế từ node đến đích.
- **Consistent** (hoặc Monotonic): Chi phí từ node nnn đến node mmm không lớn hơn chi phí từ nnn đến đích cộng với chi phí từ mmm đến đích.

3.2 Uniform Cost Search (UCS)

Giải thuật UCS là phiên bản mở rộng của thuật toán Dijkstra, luôn chọn đỉnh có chi phí nhỏ nhất từ điểm bắt đầu và duyệt theo chi phí thực tế đã đi được (g(n)). Thuật toán đảm bảo tìm được đường đi có chi phí thấp nhất nếu tồn tại..

Mã giả mô tả thuật toán

1. Khởi tạo:

- Đặt tập mở (open set) là hàng đợi ưu tiên chứa điểm bắt đầu với chi phí 0.
- Đặt tập đóng (closed set) là rỗng.

2. Lặp lại cho đến khi hàng đợi ưu tiên rỗng:

a. Lấy đỉnh n từ hàng đợi ưu tiên có chi phí thấp nhất.

b. Nếu n là điểm đích, trả về đường đi từ start đến n với chi phí tối thiểu.

c. Nếu n đã được mở rộng trước đó, bỏ qua đỉnh này (để tránh lặp).

d. Mở rộng đỉnh n:

- Với mỗi đỉnh kề m của n:

- Tính tổng chi phí từ điểm bắt đầu đến m là: g(m) = g(n) + chi phí(n, m).

- Nếu m chưa có trong hàng đợi hoặc có chi phí cao hơn chi phí mới tìm được, thêm/cập nhật m vào hàng đợi với chi phí g(m).

3. Nếu không tìm thấy đường đi, trả về "không có đường đi".

Đặc điểm:

- UCS tìm đường đi từ điểm bắt đầu đến điểm đích với chi phí thấp nhất.
- Trong quá trình tìm kiếm, UCS sử dụng hàng đợi ưu tiên (priority queue) để lựa chọn node có chi phí đến thấp nhất để mở rộng tiếp, đảm bảo rằng nó luôn mở rộng node có tổng chi phí thấp nhất trước.
- UCS được đảm bảo là tìm ra đường đi tối ưu nếu tồn tại (với điều kiện chi phí của các cạnh là không âm).

3.3 Greedy Search

Giải thuật tham lam (Greedy) chọn đỉnh tiếp theo dựa trên ước lượng chi phí (h(n)) thấp nhất từ đỉnh hiện tại đến đích mà không quan tâm đến chi phí đã đi. Greedy không đảm bảo tìm được đường đi tối ưu nhưng thường tìm được một lời giải nhanh.

Mã giả mô tả thuật toán

1. Khởi tạo:

- Đặt tập mở (open set) gồm điểm bắt đầu (start node).

- Khởi tạo tập đóng (closed set) là rỗng.

2. Lặp lại cho đến khi tập mở rỗng:

a. Chọn đỉnh n từ tập mở có giá trị heuristic h(n) nhỏ nhất.

b. Nếu n là điểm đích, trả về đường đi từ start đến n.

c. Di chuyển n từ tập mở sang tập đóng.

d. Với mỗi đỉnh kề m của n:

- Nếu m nằm trong tập đóng, bỏ qua node này.

- Nếu m chưa nằm trong tập mở:

- Thiết lập giá trị heuristic h(m).

- Cập nhật parent của m thành n.

- Thêm m vào tập mở.

3. Nếu không tìm thấy đường đi, trả về "không có đường đi".

Đặc điểm:

- **Giải pháp nhanh chóng**: Greedy Search có thể tìm ra giải pháp nhanh chóng và trong thời gian ngắn, vì nó chỉ cần thực hiện các phép toán đơn giản để đưa ra

quyết định tại mỗi bước. Do đó, mặc dù giải pháp không tối ưu, thời gian tìm kiếm nhanh có thể là ưu điểm trong nhiều ứng dụng thực tế.

- **Giải pháp không tối ưu**: Trong nhiều trường hợp, Greedy Search có thể tạo ra một giải pháp kém hơn so với các phương pháp khác, vì nó không xem xét tất cả các khả năng trước khi đưa ra quyết định. Điều này dẫn đến tình trạng giải pháp có thể không tối ưu, đặc biệt trong những bài toán có cấu trúc phức tạp

3.4 Hill Climbing

Hill Climbing là một thuật toán leo đồi, nút con kế tiếp được chọn là nút con tốt nhất trong số các nút con và phải tốt hơn nút cha. Tuy nhiên, thuật toán này có khuynh hướng sa lầy ở cực đại nội bộ, cao nguyên.

Mã giả mô tả thuật toán

1. Khởi tạo:

- Chọn một trạng thái ban đầu (current state).

2. Lặp lại cho đến khi đạt điều kiện dừng:

a. Xác định tập các trạng thái kề của trạng thái hiện tại (neighbor states).

b. Tìm trạng thái kề tốt nhất (best neighbor) trong tập các trạng thái kề.

c. Nếu trạng thái kề tốt nhất có giá trị hàm đánh giá tốt hơn trạng thái hiện tại:

- Di chuyển đến trạng thái kề tốt nhất (current state = best neighbor).

d. Nếu không có trạng thái kề nào tốt hơn, dừng thuật toán và trả về trạng thái hiện tại là lời giải.

3. Kết thúc: Trả về trạng thái hiện tại là lời giải tối ưu cục bộ.

Đặc điểm:

- Thuật toán di chuyển theo hướng làm cải thiện giá trị hàm đánh giá cho bài toán tối ưu hóa.
- Mỗi bước đều chuyển đến trạng thái kề có giá trị đánh giá tốt hơn trạng thái hiện tại.
- Thuật toán dừng khi không có trạng thái kề nào có giá trị tốt hơn, tại đó đạt đến điểm cực trị cục bộ (đỉnh của "đồi").

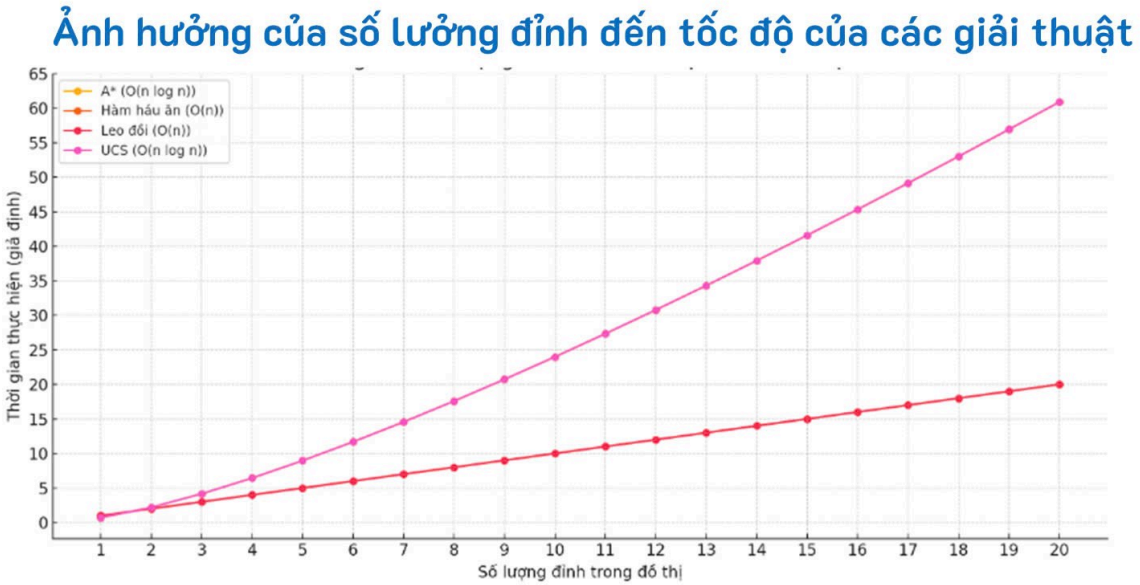
3.5 So sánh các thuật toán

Bảng so sánh:

Thuật Toán	Đặc Điểm Chính	Ưu Điểm	Nhược Điểm	Ứng Dụng Thực Tế
A Search	Kết hợp giữa chi phí đi từ nút gốc và ước	Tìm kiếm tối ưu, hiệu quả với ước lượng tốt.	Có thể tốn bộ nhớ lớn, phụ thuộc vào hàm ước lượng.	Tìm đường, AI trong trò chơi.

	lượng chi phí còn lại (h(n)).			
Uniform Cost Search (UCS)	Tìm kiếm theo chi phí nhỏ nhất từ nút gốc đến các nút con.	Đảm bảo tìm được giải pháp tối ưu.	Chậm hơn A* trong các bài toán lớn, tốn bộ nhớ.	Tìm kiếm đường đi trong mạng lưới.
Greedy Search	Chỉ sử dụng ước lượng chi phí đến đích (h(n)).	Nhanh, ít tốn bộ nhớ hơn A* và UCS.	Không đảm bảo tìm được giải pháp tối ưu.	Tìm kiếm trong không gian lớn.
Hill Climbing	Tìm kiếm theo hướng tăng dần giá trị (local optimum).	Đơn giản, nhanh chóng trong không gian nhỏ.	Dễ bị kẹt ở các cực trị địa phương (local maxima).	Tối ưu hóa, giải quyết vấn đề tìm kiếm.

Mối quan hệ giữa tốc độ tìm đường của thuật toán với tổng số đỉnh trên đồ thị



Thuật toán A*: Thời gian tìm kiếm tỷ lệ thuận với tổng số đỉnh trên đồ thị, nhưng có thể được cải thiện nhờ vào việc chọn hàm h(n) tốt.

Thuật toán háu ăn:

+ Thời gian thực hiện tăng theo cấp số nhân với tổng số đỉnh trong đồ thị, đặc biệt khi không có phương pháp cắt tỉa.
+ Không tối ưu cho các bài toán có không gian tìm kiếm lớn.

Thuật toán leo đồi: Thời gian tìm kiếm có thể nhanh hơn so với các thuật toán khác, nhưng nó có thể bị mắc kẹt trong cực tiểu địa phương.

Thuật toán UCS:

+ Tìm kiếm theo chi phí dẫn đến tốc độ tăng theo tổng số đỉnh, nhưng tốt hơn A* trong những tình huống không có hàm h(n) tốt.

+ Thời gian thực hiện có thể rất lớn nếu có nhiều đỉnh với chi phí tương đối thấp.

 Tốc độ của thuật toán còn phụ thuộc và hàm heuristic xây dựng có phù hợp hay không

4. Thiết kế và cài đặt

4.1 Thiết kế giao diện người dùng

+ **Phần hiển thị đồ thị**: Sử dụng thư viện **Vis.js** để vẽ đồ thị tương tác.
+ **Phần hiển thị giá trị heuristic**: Danh sách các giá trị h(n)h(n)h(n) của từng đỉnh.
+ **Phần nhập liệu và kết quả**:
    - Chọn file đồ thị từ máy tính.
    - Nhập đỉnh bắt đầu và đỉnh kết thúc.
    - Lựa chọn giải thuật tìm kiếm.
    - Nút thực hiện tìm kiếm.
    - Hiển thị kết quả: đường đi, thứ tự duyệt, tổng chi phí.

- Giao diện được xây dựng bằng HTML và Bootstrap để tạo sự thân thiện và dễ sử dụng cho người dùng.

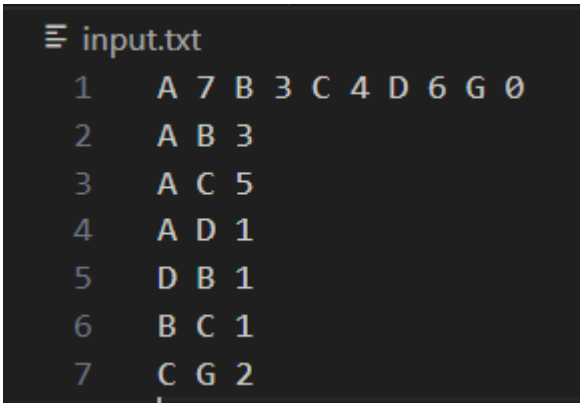
4.2 Cấu trúc mã nguồn

- **File HTML (index.html)**:
    - Chứa cấu trúc giao diện và các phần tử DOM cần thiết.
    - Kết nối với các file JavaScript để xử lý logic.
- **File JavaScript chính (javascript.js)**:
    - Xử lý việc tải file đồ thị, phân tích và lưu trữ dữ liệu.
    - Quản lý sự kiện người dùng (nhấp chuột, nhập liệu).
    - Gọi các hàm giải thuật tương ứng dựa trên lựa chọn của người dùng.
    - Cập nhật giao diện sau khi có kết quả tìm kiếm.
- **Các file JavaScript cho từng giải thuật**:
    - **gtAsao.js**: Cài đặt giải thuật A*.

- **ucs.js**: Cài đặt giải thuật Uniform Cost Search.
- **greedy.js**: Cài đặt giải thuật Tham Ăn.
- **hillClimbing.js**: Cài đặt giải thuật Leo Đồi.

4.3 Cài đặt

Input: Người dùng tải lên file đồ thị với định dạng cụ thể. Ví dụ:



- **Hàng đầu tiên**: Danh sách các đỉnh và giá trị heuristic (h) tương ứng với từng đỉnh.
- **Các hàng tiếp theo**: Thông tin về các cạnh trong đồ thị, mô tả đường đi giữa hai đỉnh và trọng số chi phí (g) của từng cạnh.

Cài đặt các giải thuật:

- **A***, **UCS**, **Greedy**, và **Hill Climbing** sẽ được triển khai trong các file JavaScript riêng biệt, mỗi giải thuật có phương thức tìm kiếm riêng để xác định đường đi từ đỉnh bắt đầu đến đỉnh kết thúc.
- Mỗi giải thuật sẽ duyệt đồ thị và tính toán đường đi dựa trên các quy tắc riêng.

Xử lý dữ liệu:

- File đầu vào được phân tích và chuyển thành cấu trúc dữ liệu đồ thị với các đỉnh, cạnh, và trọng số. Các giải thuật sẽ dựa trên cấu trúc này để tính toán đường đi.

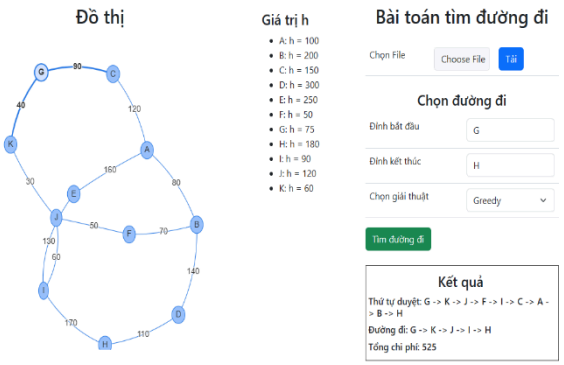
Tính năng tìm kiếm: Khi người dùng nhập đỉnh bắt đầu và đỉnh kết thúc, cùng với việc chọn giải thuật, hệ thống sẽ thực hiện tìm kiếm và trả về đường đi ngắn nhất hoặc tốt nhất theo giải thuật được chọn.

Hiển thị kết quả:

- Kết quả sẽ bao gồm danh sách các đỉnh trong đường đi, thứ tự duyệt của từng đỉnh trong quá trình tìm kiếm, chi phí tích lũy (tổng trọng số của các cạnh trong đường đi), và thời gian thực hiện tìm kiếm.
- Đường đi tìm được sẽ được hiển thị trực quan trên đồ thị sử dụng **Vis.js**, cùng với thông tin chi tiết về chi phí và thứ tự duyệt.

PHẦN KẾT LUẬN

1. Kết quả đạt được



Website đã hoàn thành và cho phép người dùng tìm đường đi trên bản đồ (biểu diễn dưới dạng đồ thị) sử dụng các giải thuật như A*, UCS, Greedy, và Hill Climbing.

Hệ thống hoạt động ổn định, đáp ứng đúng các yêu cầu đề ra về việc tìm kiếm và hiển thị đường đi, kèm theo thông tin chi phí và thứ tự duyệt.

❖ Link github của đồ án:

https://github.com/TuanB2013571/CT33202_Nhom01.git

2. Hướng phát triển

Cải thiện giao diện người dùng: Tinh chỉnh và cải thiện giao diện để tạo trải nghiệm người dùng tốt hơn, thêm tính năng tương tác trực quan với đồ thị.

Phát triển thành các ứng dụng tìm đường có Cơ sở dữ liệu, ứng dụng game trí tuệ nhân tạo tìm đường…

TÀI LIỆU THAM KHẢO

[1]. *TS.Trần Nguyễn Minh Thư.(2022).Giáo trình Nhập môn Trí Tuệ Nhân Tạo. Nhà xuất bạn Đại học Cần Thơ.*

[2]. *PGS.TS Phạm Nguyên Khang, ThS. Phạm Gia Tiến.(2018). Giáo trình Trí Tuệ Nhân Tạo. Nhà xuất bản Đại học Cần Thơ.*