ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

CƠ SỞ TRÍ TUỆ NHÂN TẠO – CSC 14003_21CLC07

LAB 1

N – QUEENS PROBLEM

21127191 – Nguyễn Nhật Truyền

I. MỨC ĐỘ HOÀN THÀNH:

Thuật toán	Mức độ hoàn thành
Uniform cost search	100%
A*	100%
Genetic algorithm	100%

II. PHÂN TÍCH YÊU CẦU:

- Xét bài toán đặt N quân hậu lên bàn cờ kích thước NxN (N được nhập từ bàn phím) sao cho không có bất kì cặp quận hậu nào tấn công được nhau.
- Với mỗi trạng thái tương ứng với 1 mảng 1 chiều N phần tử. Con hậu ở hàng i sẽ có giá trị là phần tử thứ i của mảng trạng thái.
  - Ví dụ: mảng queens[0, 1, 2, 3] có nghĩa là một bàn cờ kích thước 4x4 và các quân hậu nằm đường chéo chính của bàn cờ. Queens[2] = 2.
- Với yêu cầu bài toán, trạng thái bắt đầu đầy đủ(complete-state) thì tất cả các quân hậu sẽ đều ở trên bàn cờ và sẽ ở khác cột với nhau.
- Những trạng thái tiếp theo sẽ được sinh ra bằng cách di chuyển một quân hậu sang 1 ô vuông khác trong cùng 1 cột ở trạng thái trước.

III. MÔ TẢ THUẬT TOÁN:

Những định nghĩa chung:

- Trạng thái ban đầu(initial state): ở mỗi hàng chúng ta sẽ random(một giá trị trong khoảng 0 -> n – 1, với n là kích thước bàn cờ) tương tứng cho chỉ số cột đặt quân hậu.

```
initial_board = [random.randint(0, n - 1) for _ in range(n)]
```

- Trạng thái hoàn thành(goal state): là trạng thái không có bất kì cặp quân hậu nào tấn công nhau tức giữa 2 cặp quân hậu không cùng nằm trên một hàng và đường chéo chính/ phụ (không xét tới cột vì với trạng thái ban đầu thì các quân hậu đã nằm khác cột với nhau và cách trạng thái cũng được sinh ra bằng cách di chuyên trên cột của nó). Kết quả sẽ được in ra dưới dạng hiển thị bên dưới

```
Solution found:
* Q * *
* * * Q
Q * * *
* * Q *
```

- Hàm chi phí (cost): sẽ là số bước từ trạng thái đầu sinh ra trạng thái đó. Với mỗi bước hay mỗi trạng thái thay đổi 1 quân hậu sẽ có chi phí là 1.
- Hàm heuristic: theo đề bài thì hàm heuristic là "số cặp quân hậu tấn công nhau".

1. Uniform cost search:

- Bắt đầu với trạng thái ban đầu và hàng đầu tiên. Sinh ra các trạng thái khác bằng cách di chuyên quân hậu ở hàng đầu tiên sang ô vuông khác trong cùng cột đó và tăng chi phí của trạng thái vừa sinh ra lên 1.
- Ở mọi bước chọn trạng thái để sinh trạng thái mới chúng ta đều phải chọn trạng thái có chi phí thấp nhất. Ở đây chúng ta có thể sử dụng cấu truc heap-min để tối ưu cho thuật toán.
- Mỗi khi sinh ra trạng thái mới chúng ta sẽ tính toán chi phí dựa trên trạng thái cũ và đưa chúng vào hàng đợi ưu tiên dựa trên chi phí.
- Lặp lại quá trình đó cho cái cột khác cho tới khi tìm được trạng thái mục tiêu hoặc khi hàng đợi trống.

➢ Nhận xét: UCS không hiệu quả cho bài toán N-Queen vì không tận dụng được thông tin đặc biệt của thuật toán và việc phải sinh ra mọi trạng thái cho từng cột cho đến khi tìm được trạng thái đích.

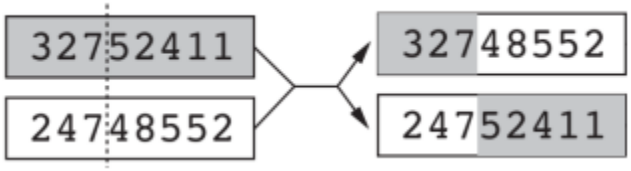
2. A*:

- Tính số cặp quân hâu xung đột ở trạng thái đầu tiên và khởi tạo chi phí ban đầu. Với thuật toán này chúng ta sẽ xác định độ ưu tiên hàm f = g + h (g là hàm chi phí và h là hàm heuristic).
- Bắt đầu với trạng thái ban đầu. Lần lượt duyệt qua các cột mỗi lần di chuyển 1 quân hậu trên 1 cột, tăng chi phí lên 1 và tính toán lại số xung đột và đưa chúng vào hàng đợi ưu tiên.
- Ở đây chúng ta sẽ tối ưu việc sinh trạng thái bằng cách thêm 1 chỉ số đánh dấu. Tức là chúng ta sẽ chỉ sinh trạng thái mới bằng cách chỉ thay đổi vị trí quân hậu từ cột đánh dấu trở về sau để tránh việc phải tạo ra các trạng thái trùng lặp không cần thiết. Dễ thấy, giả sử chúng ta thay đổi quân hậu ở cột 1 và sinh ra được 1 số trạng thái mới, từ những trạng thái mới đó chúng ta xét tới cột 2 thì việc sinh trạng thái mới bằng cách di chuyển lại những quân hậu từ cột 1 sẽ tạo ra những trạng thái trùng lặp và ta phải tốn thêm chi phí để xử lí những trường hợp vô nghĩa đó.
- Lặp lại quá trính đó cho các trạng thái được lấy từ hàng đợi ưu tiên cho đến khi số xung đột bằng 0, tức là không có quân hậu nào tần công nhau(goal state).

- Nhận xét: A* sẽ tìm kiếm các trạng thái theo thứ tự ưu tiên dựa trên tổng của chi phí(cost) và hàm ước lượng(heuristic). Điều này giúp thuật toán tìm kiếm dần về trạng thái mục tiêu và tìm ra lời giải tối ưu hơn cho bài toán.

3. Genetic Algorithm:

- Khởi tạo quần thể ban đầu: Ở đây chúng ta có 1 hàm tính số xung đột của các cặp quân hậu. Chúng ta sẽ random số lượng cá thể cho quần thể đó(cá thể tương ứng cho 1 trạng thái). Và cũng sẽ random ra số cá thể cho quần thể đó. Đưa chúng vào hàng đợi ưu tiên theo độ tốt của cá thể (tức là cá thể tương ứng số cặp quân hậu tấn công nhau ít nhất)
- Từ quần thể ban đầu chọn ra một số cá thể tốt nhất từ hàng đợi ưu tiên và tiến hành lai ghép (crrossover) theo cặp.
- Lai ghép: Chọn ra 1 vị trí ngẫu nhiên. Kết quả lai ghép sẽ tạo ra 2 cá thể con bằng cách ghép 1 phần của cá thể cha(từ vị trí đầu tới vị trí được chọn) và 1 phần của cá thể mẹ(từ vị trí được chọn trở về sau). Tương tự với cá thể con còn lại (phần đầu của cá thể mẹ tới vị trí được chọn nối với vị trí được chọn trở về sau của cá thể cha)



- Đột biến(mutate): Mỗi vị trí của cá thể con sẽ mang 1 xác suất nhỏ để xảy ra đột biến(tức thay đổi 1 quân hậu trong 1 cột). Điều này xảy ra để giải thoát cho trạng thái ngõ cụt(không sinh ra được goal state) tuy nhiên nếu chọn xác suất đột biến quá cao sẽ dẫn đến việc trạng kém hội tụ về trạng thái hoàn thành.(10% xác suất đột biến cho lần xử lí thuật toán này)
- Sau khi tạo ra đươc các cá thể con mới chúng ta đưa chúng vào hàng đợi và tiếp tục quá trính chọn lọc, lai ghép, đột biến trên cho đến khi tìm được trạng thái hoàn thành.
- Nhận xét: thuật toán Genetic Algorithm giúp tìm kiếm không gian lời giải rộng lớn và tìm ra những giải pháp xấp xỉ tốt nhất cho bài toán N Queens. Việc lựa chọn phù hợp các hàm chọn lọc, lai ghép và đột viến có vai trò quan trọng trong hiệu suất của thuật toán.

IV. SỐ LIỆU THỐNG KÊ:

Giới hạn thời gian:

Giới hạn bộ nhớ:

Mỗi thuật toán chúng ta sẽ chạy 3 lần với mỗi test và lấy trung bình giá trị đó làm giá trị cuối cùng.

	Running time (s)			Memory (MB)		
Algorithm	N = 8	N = 100	N = 500	N = 8	N = 100	N = 500
UCS	23.4868	X	X	199,93	X	X
A*	14.3668	14542.324	X	232.79	1546.82	X
GA	17.5771	X	X	0.05	X	X

➢ Nhận xét:

- **Thời gian:**

- Uniform Cost Search (UCS): Thời gian thực thi của UCS phụ thuộc vào số lượng trạng thái cần duyệt qua. Với kích thước bàn cờ tăng lên, số lượng trạng thái sẽ tăng đáng kể. Do đó, UCS không phù hợp cho các bàn cờ lớn.
- A* Algorithm: A* sử dụng hàm chi phí ước lượng để ưu tiên các trạng thái có tiềm năng tốt hơn. Thời gian thực thi của A* phụ thuộc vào cách ước lượng chi phí và độ phức tạp của hàm chi phí ước lượng. Với bàn cờ kích thước lớn, A* cũng sẽ gặp phải các vấn đề về thời gian tính toán và có thể mất rất nhiều thời gian để tìm lời giải.
- Genetic Algorithm (GA): GA thực hiện quá trình tiến hóa trên quần thể và số thế hệ cần thiết để tìm ra lời giải tốt có thể tăng lên đáng kể với kích thước bàn cờ lớn. Việc tính toán và so sánh các cá thể trong quần thể cũng đòi hỏi thời gian tính toán đáng kể.

- **Bộ nhớ:**

- Uniform Cost Search (UCS): UCS yêu cầu lưu trữ các trạng thái và thông tin liên quan trong quá trình tìm kiếm. Với kích thước bàn cờ lớn việc sinh ra các trạng thái và lưu trữ sẽ là 1 vấn đề của thuật toán này/
- A* Algorithm: A* cần lưu trữ các trạng thái, thông tin chi phí, và thông tin ước lượng trong quá trình tìm kiếm. Với kích thước bàn cờ lớn, bộ nhớ trung bình của A* có thể tăng lên đáng kể và yêu cầu bộ nhớ lớn.
- Genetic Algorithm (GA): GA yêu cầu lưu trữ quần thể và các cá thể trong quá trình tiến hóa. Với kích thước bàn cờ lớn, bộ nhớ trung bình của GA sẽ tăng lên đáng kể và yêu cầu bộ nhớ lớn, phụ thuộc vào kích thước quần thể và cách lưu trữ dữ liệu(ở lần thử nghiệm này vì lấy số lượng quần thể không vượt quá N, với N là số lượng quân hậu nên bộ nhớ yêu không quá lớn).

✓ Tóm lại, với bài toán N-Queen trên bàn cờ kích thước lớn dần, cả A* và Genetic Algorithm sẽ đối mặt với các vấn đề về thời gian tính toán và bộ

nhớ do độ phức tạp của bài toán. Để giải quyết bài toán này hiệu quả, cần sử dụng các thuật toán và kỹ thuật tối ưu hơn.