BÀI TẬP THỰC HÀNH

NHẬP MÔN TRÍ TUỆ NHÂN TẠO

TUẦN 3

LOCAL SEARCH – THUẬT TOÁN LEO ĐỒI

Yêu cầu: giải quyết bài toán N hậu bằng thuật toán thuật toán leo đồi

Input: số nguyên N từ 4 đến 8

Output: danh sách vị trí các quân hậu trên bàn cờ

Hướng dẫn:

Thuật toán Local Search dùng để giải quyết các bài toán có hai thuộc tính sau:

- Mỗi state cũng là một solution, solution này có thể khả thi (tức hợp lệ) hoặc không khả thi
- Đường đi đến goal state không quan trọng, chỉ quan trọng kết quả goal state chi tiết như thế nào

Vì vậy, bài toán N hậu khi giải quyết bằng thuật toán Local Search sẽ có những đặc điểm:

- Mỗi state có chứa đủ N quân hậu
- Các quân hậu có thể xung đột lẫn nhau, hoặc không xung đột lẫn nhau
- Initial state được tạo ngẫu nhiên
- Goal state là giải pháp được tạo ra dựa trên initial state

Đặc điểm của thuật toán leo đồi là không đảm bảo tìm được giải pháp tối ưu, chỉ có thể tìm được lân cận tốt nhất mà thôi (không chắc có được tối ưu toàn cục, chỉ có thể tìm được tối ưu cục bộ).

Từng bước giải quyết bài toán này, có thể tóm tắt:

- Tạo initial state ngẫu nhiên (tham khảo bài thực hành 2, dùng danh sách để lưu trữ)
- Đánh giá mức độ hợp lý của initial state bằng cách kiểm tra tổng xung đột ngang và chéo
- Tạo các neighbour state bằng cách thay đổi vị trí các quân hậu trên bàn cờ, lưu ý neighbour state chỉ khác với initial state một vị trí duy nhất
- Đánh giá mức độ hợp lý của tất cả các neighbor state
- Chọn neighbour state tốt nhất để đi tiếp, nếu có nhiều state cùng một độ tốt thì chọn ngẫu nhiên
- Lặp lại quá trình sinh neighbour state từ state được chọn, cho đến khi không thể cải thiện độ tốt của state
- In kết quả ra màn hình, nếu kết quả tìm được hoàn toàn không có xung đột, tức là ta tìm được goal state (tối ưu toàn cục), nếu không ta chỉ tìm được một state tốt hơn initial state mà thôi (tối ưu cục bộ)

Thực hiện cài đặt từng bước theo hướng dẫn, chú ý đây chỉ là một gợi ý, nếu sinh viên có thể nghĩ ra cách cài đặt khác thì nên thử thách bản thân viết ra những gì chính bản thân mình mong muốn.

Bước 1: Xây dựng cấu trúc dữ liệu Node

```
public class Node {
	int n;
	List<Integer> state;
	public Node (int n)
	{
		this.n=n;
		this.state=new ArrayList<Integer>();
		for (int i=0;i<n;i++)
		{
			Random r=new Random();
			this.state.add(r.nextInt(n));
		}
	}
	public Node(int n, List<Integer> state) {
		this.n = n;
		this.state = state;
	}
}
```

Bước 2: Cài đặt class LocalSearch không chứa thuộc tính, không constructor, chỉ có các hàm sau

1. `public int checkHorizontal(Node node)`: trả về tổng xung đột ngang trong một node
2. `public int checkDiagonal(Node node)`: trả về tổng xung đột chéo trong một node
3. `public int heuristic(Node node)`: hàm đánh giá mức độ hợp lý của một node, trả về tổng xung đột ngang và chéo
4. `public int tryMovingOneQueen(Node node, int x, int y)`: thay đổi vị trí quân hậu của node tại cột y (đang ở bất kỳ dòng nào) thành dòng x, trả về heuristic của node mới
5. `public SortedMap<Integer,Node> generateNeighbours(Node node)`: tạo tất cả các neighbour node của node đang xét, đánh giá các neighbour node này, nếu có nhiều node có cùng một kết quả đánh giá, chỉ cần lưu lại một node trong chúng. Trả về một SortedMap trong đó key là kết quả đánh giá, value là node mới.
6.
```
public void run()
{
	Node initial = new Node(8); //hoặc 4,5,6,7
	if (heuristic(initial)==0) //goal
	{
		System.out.println(initial.state);
		return;
	}
	System.out.println("Initial state is: "+initial.state);
	Node node=initial;
	SortedMap<Integer,Node> neighbours=
generateNeighbours(node);
	Integer bestHeuristic = neighbours.firstKey();
	while (bestHeuristic<heuristic(node))
	{
		node = neighbours.get(bestHeuristic);
		neighbours= generateNeighbours(node);
```

```
                bestHeuristic=neighbours.firstKey();
            }
            if (heuristic(node)==0)
            {
                System.out.println("Goal is: "+node.state);
            }
            else
                System.out.println("Cannot find goal state! Best state
                is: "+ node.state);
    }
```

Bước 3: viết hàm main, tạo class Local Search và chạy hàm run của class này