Bài 1.6: Quay lui

- ✓ Định nghĩa
- ✓ Đặc điểm
- ✓ Quy trình & cài đặt
- ✓ Ví dụ minh họa
- ✓ Bài tập thực hành

Định nghĩa

- Tiếng Anh - backtracking là một thuật toán để giải quyết các vấn đề một cách đệ quy bằng cách cố gắng xây dựng một giải pháp từng bước, từng phần, loại bỏ những giải pháp không đáp ứng được các ràng buộc của vấn đề tại bất kì thời điểm nào.
- Hoặc: quay lui là một kĩ thuật tổng quát xem xét việc tìm kiếm mọi kết hợp có thể có để giải quyết một bài toán.
- Các bài toán thỏa mãn ràng buộc là các bài toán có một lời giải đầy đủ, trong đó thứ tự của các phần tử không quan trọng.
- Các bài toán này bao gồm một tập các biến mà mỗi biến cần được gán một giá trị tùy theo các ràng buộc cụ thể của bài toán.
- Việc quay lui là để thử tất cả các tổ hợp nhằm tìm được một lời giải.

Đặc điểm

Có ba loại vấn đề trong quay lui:

- Vấn đề ra quyết định: ở đây ta cần tìm kiếm một giải pháp khả thi.
- Vấn đề tối ưu hóa: ở đây ta cần tìm ra giải pháp tốt nhất.
- Vấn đề liệt kê: ở đây ta cần tìm tất cả các giải pháp khả thi.

Sự khác nhau giữa đệ quy và quay lui:

- Đệ quy gọi lại bản than nó tới khi gặp trường hợp cơ sở.
- Quay lui sử dụng đệ quy để mở rộng tất cả các khả năng đến khi tìm được kết quả tốt nhất cho vấn đề.

Đặc điểm

- Nói chung, mọi vấn đề về thỏa mãn ràng buộc có các ràng buộc được xác định rõ ràng đối với bất kì giải pháp khách quan nào, trong đó sử dụng các ứng viên để từng bước xây dựng lời giải và loại bỏ các ứng viên ngay khi xác định rằng nó không thể góp phần tìm ra lời giải thì có thể giải quyết bằng quay lui.
- Hầu hết các bài toán được đặt ra đều có thể sử dụng các thuật toán khác để giải nhằm tối ưu hóa độ phức tạp. Độ phức của backtracking thường là số mũ.
- Ví dụ các bài toán giải được bằng quay lui: mã đi tuần, N quân hậu, sudoku, tổng tập con…

Quy trình

- Tư tưởng của phương pháp: thử từng khả năng đến khi tìm được lời giải đúng.
- Đây là một quá trình tìm kiếm theo chiều sâu trong tập các lời giải.
- Khi tìm kiếm nếu gặp hướng lựa chọn không thỏa, ta quay lui về điểm lựa chọn trước đó, nơi có các hướng khác và thử hướng lựa chọn tiếp theo.
- Khi đã hết các lựa chọn xuất phát từ điểm lựa chọn đó, ta quay lui về điểm lựa chọn trước đó và thử hướng lựa chọn tiếp theo tại đó.
- Quá trình tìm kiếm thất bại khi không còn điểm lựa chọn nào nữa.

Cài đặt

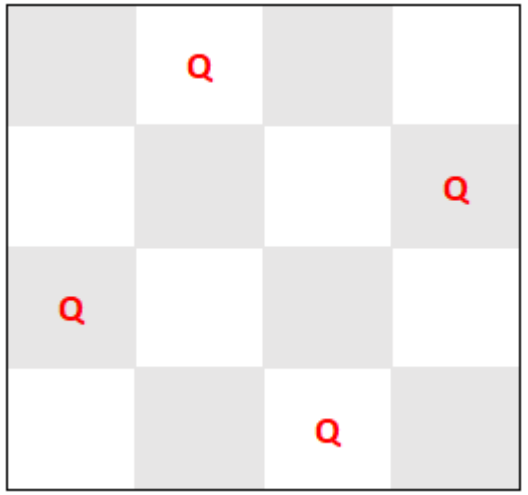
- Quy trình trên thường được cài đặt bằng phương thức đệ quy, trong đó mỗi thể hiện của phương thức lấy thêm một biến và lần lượt gán tất cả các giá trị có thể có cho biến đó.
- Mỗi lần gán giá trị ta lại gọi chuỗi đệ quy tiếp theo để thử các biến tiếp theo.
- Chiến lược quay lui tương tự với tìm kiếm theo chiều sâu nhưng sử dụng ít không gian bộ nhớ hơn, tức chỉ lưu giữ trạng thái của một lời giải hiện thời và cập nhật nó.
- Để tăng tốc quá trình tìm kiếm, khi một giá trị được chọn, trước khi thực hiện lời gọi đệ quy, thuật toán thường xóa bỏ giá trị đó khỏi miền xác định của các biến có mâu thuẫn chưa được gán và kiểm tra tất cả các hằng số để tìm các giá trị khác đã bị loại trừ bởi giá trị bừa được gán.

Heuristic

- Người ta thường sử dụng một số phương pháp heuristic để tăng tốc cho quá trình quay lui.
- Do các biến có thể được xử lý theo thứ tự bất kì, việc thử các biến bị ràng buộc chặt nhất thường có hiệu quả do nó tỉa cây tìm kiếm từ sớm.
- Các cài đặt quay lui phức tạp thường sử dụng một hàm biên để kiểm tra xem từ lời giải chưa đầy đủ hiện tại có thể thu được một lời giải hay không.
- Nhờ đó một kiểm tra biên có thể phát hiện các lời giải dở dang chắc chắn thất bại do vậy nâng cao hiệu quả tìm kiếm.
- Do hàm này hay được chạy, có thể tại mỗi bước, nên chi phí tính toán các biên cần tối thiểu nếu không sẽ ảnh hưởng đến kết quả tổng thể của thuật toán.
- Các hàm kiểm tra biên được tạo theo kiểu gần như các hàm heuristic khác: nới lỏng một số điều kiện của bài toán.

Ví dụ giải bài toán N quân hậu

- Bài toán: đặt N quân hậu trên bàn cờ kích thước NxN sao cho chúng không ăn lẫn nhau theo từng đôi.
- Sau đây lời giải của bài toán với N = 4:



Các bước thực hiện

- B1: xuất phát từ cột đầu tiên trong bàn cờ.
- B2: Nếu cột đang xét >= N, return true. Đã tìm thấy lời giải.
- B3: Duyệt tất cả các hàng tại cột hiện tại, lặp:
  - B3.1: Nếu đặt hậu tại vị trí hàng thứ i là an toàn, đánh dấu vị trí đó làm một phần trong lời giải.
  - B3.2: Gọi đệ quy để tìm hàng trên cột kế tiếp có thể đặt được quân hậu. Đồng thời đánh giá xem liệu việc đặt hậu tại vị trí này có đưa tới lời giải hay không.
  - B3.3: Nếu việc đặt hậu tại vị trí này đưa tới lời giải, return true.
  - B3.4: Ngược lại, bỏ đánh dấu vị trí hiện tại và quay lại bước 3.1 và thử lại với các hàng khác.
- B4: Nếu tất cả các hàng đã được thử và không có kết quả, return false.

Code mẫu

```
private static boolean solveNQueen(int[][] board, int col) {
    if (col >= N) {
        return true;
    }
    // kiểm tra cột hiện tại và đặt hậu vào đúng vị trí
    for (int i = 0; i < N; i++) {
        if (isSafe(board, i, col)) {
            board[i][col] = 1; // 1 == hậu được đặt tại vị trí đó
            if (solveNQueen(board, col: col + 1)) {
                return true;
            } else {
                board[i][col] = 0;
            }
        }
    }
    return false;
}
```

Kiểm tra quân hậu có đặt được tại vị trí [x, y] không

- Các quân hậu được đặt tại các cột theo vị trí từ trái sang phải. Giả định cần kiểm tra xem việc đặt hậu tại vị trí row, col có an toàn không(khả thi).
- B1: kiểm tra xem trên hàng hiện tại có quân hậu nào chưa.
- B2: kiểm tra đường chéo trên xem có quân hậu nào không.
- B3: Kiểm tra đường chéo dưới có quân hậu nào không.
- B4: Nếu tất cả các kiểm tra trên không trả về false, bước này trả về true.

Code mẫu

```
private static boolean isSafe(int[][] board, int row, int col) {
    // kiểm tra hàng hiện thời xem ở phía bên trái đã có quân hậu nào chưa?
    for (int i = 0; i < col; i++) {
        if (board[row][i] == 1) {
            return false;
        }
    }
    // kiểm tra đường chéo trên của ô hiện thời xem có quân hậu nào chưa
    for (int i = row, j = col; i >= 0 && j >= 0; i--, j--) {
        if (board[i][j] == 1) {
            return false;
        }
    }
    // kiểm tra đường chéo dưới của ô hiện thời xem có quân hậu nào chưa
    for (int i = row, j = col; i < N && j >= 0; i++, j--) {
        if (board[i][j] == 1) {
            return false;
        }
    }
    return true;
}
```

Phương thức main()

```
public static void main(String[] args) {
    int[][] board = new int[N][N];
    // giả sử ban đầu chưa đặt quân hậu nào, bàn cờ rỗng, tất cả các ô
    // nhận giá trị 0
    boolean result = solveNQueen(board, col: 0);
    if (result) {
        System.out.println("Một trong các lời giải là: ");
        showResult(board);
    } else {
        System.out.println("Không tìm thấy lời giải.");
    }
}
```

```
Một trong các lời giải là:
  1  0  0  0  0  0  0  0
  0  0  0  0  0  0  1  0
  0  0  0  0  1  0  0  0
  0  0  0  0  0  0  0  1
  0  1  0  0  0  0  0  0
  0  0  0  1  0  0  0  0
  0  0  0  0  0  1  0  0
  0  0  1  0  0  0  0  0
```

Nội dung tiếp theo

Sinh hoán vị chính tắc