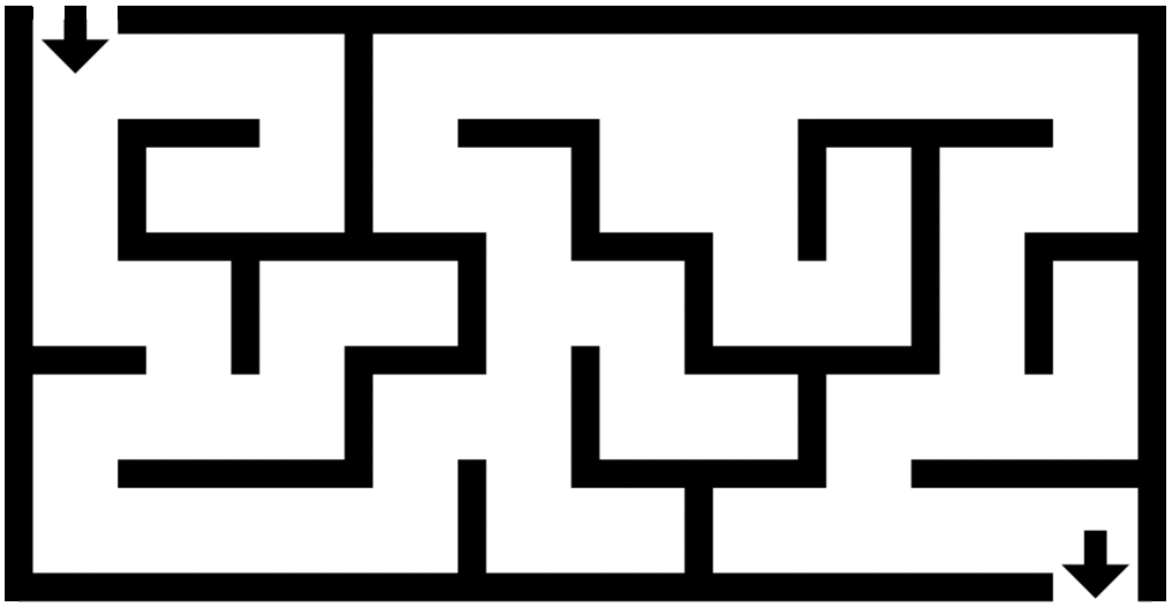
1. **Backtracking là gì ?**

- Backtracking là một tư tưởng thiết kế thuật toán dựa trên đệ quy để tìm ra tất cả (hoặc 1 số) giải pháp cho một bài toán. Đặc biệt là bài toán liên quan tới sự ràng buộc (Constraint satisfaction problem).
- Backtracking phụ thuộc vào yêu cầu (ràng buộc) và việc các "Ứng viên" (1 phần giải pháp) phát triển thành giải pháp hoàn chỉnh

2. **Backtracking hoạt động như thế nào ?**

Ý tưởng: Giống như cách mà con người chúng ta dùng để giải một mê cung

Mỗi khi vào một ngã rẽ có nhiều lựa chọn hoặc đường để đi liệu xem nó có dẫn đến đường ra hay không, nếu không thì quay lại và chọn con đường khả dụng tiếp theo và liệu xem nó có dẫn đến đường ra hay không.



Vì vậy chỉ cần lặp lại quá trình này mỗi khi phải đưa ra lựa chọn. Đó là lí do tại sao Backtracking là một thuật toán đệ quy.

Cây không gian trạng thái:

- Để tiện diễn tả giải thuật quay lui, ta xây dựng cấu trúc cây ghi những lựa chọn đã được thực hiện. Cấu trúc cây này được gọi là cây không gian trạng thái (state space tree).
- Root node của cây diễn tả trạng thái đầu tiên trước khi quá trình tìm kiếm lời giải bắt đầu.
- Các node ở tầng đầu tiên trong cây diễn tả những lựa chọn được làm ứng với thành phần đầu tiên của lời giải.
- Các node ở tầng thứ haì trong cây diễn tả những lựa chọn được làm ứng với thành phần thứ hai của lời giải và các tầng tiếp theo là tương tự như thế.
- Một node trên cây KGTT được gọi là triển vọng (promising) nếu nó tương ứng với lời giải bộ phận mà sẽ có thể dẫn đến lời giải đầy đủ; trái lại, nó được gọi là một lời giải không triển vọng (non-promising).
- Các node lá diễn tả những trường hợp bế tắc (dead node) hay những lời giải đầy đủ.

Mã giả:

```
Backtracking(k) {
        for([Mỗi phương án chọn i(thuộc tập D)]) {
                if ([Chấp nhận i]) {
                        [Chọn i cho X[k]];
                        if ([Thành công]) {
                                [Đưa ra kết quả];
                        } else {
                                Backtracking(k+1);
                                [Bỏ chọn i cho X[k]];
                        }
                }
        }
}
```

3. **Khi nào sử dụng Backtracking?**

ỨNG DỤNG VÀO PROBLEM

- Đối với những bài toán yêu cầu ràng buộc điều kiện thì backtracking là một sự lựa chọn tốt.
- Thường là các vấn đề giải đố (puzzle problems) như sudoku, crossword ,verbal arithmetic , hamiltonian cycle …

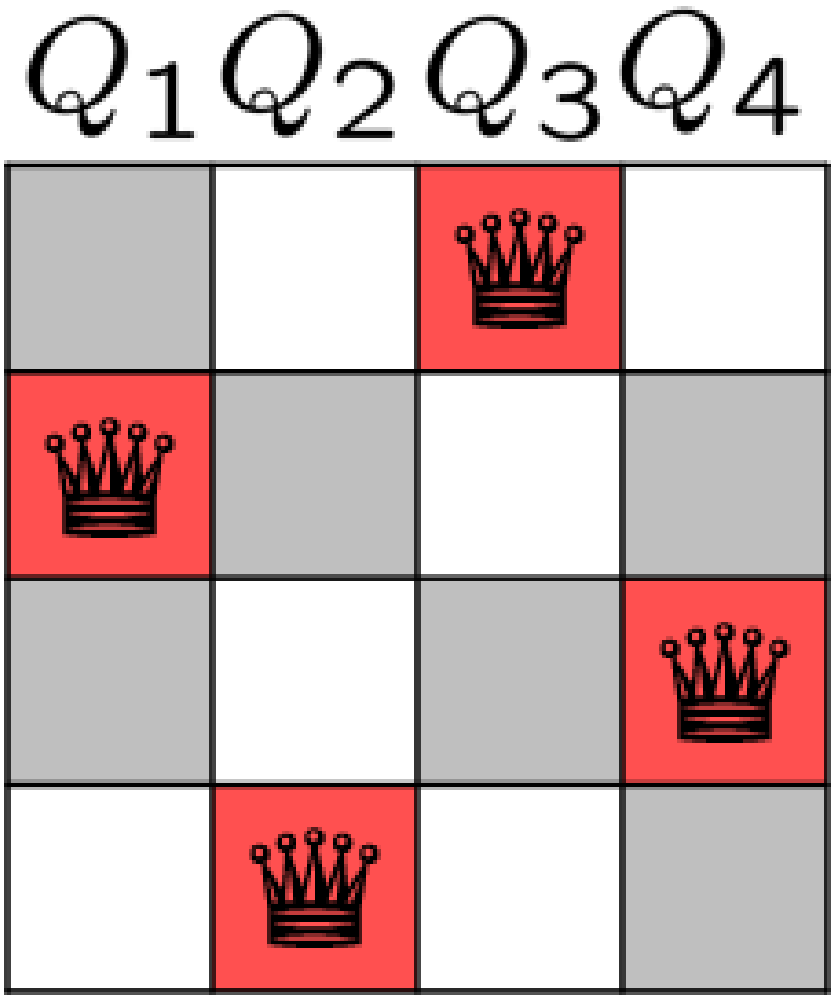
ỨNG DỤNG VÀO THỰC TẾ

- Lập trình game, giải các bài toán liên quan tới bản đồ (tô màu đồ thị, tìm đường đi ngắn nhất ,...)
- Vấn đề về phân công công việc. vd: giáo viên nào dạy lớp nào
- Vấn đề về phân bố thời gian biểu
- Cấu hình phần cứng
- Lập kế hoạch vận chuyển

4. **Minh họa bài toán N-queens và Knight's Tour**

a. N-queens problem

Mô tả bài toán: Bài toán N ($N \geq 4$) quân hậu là bài toán đặt N quân hậu trên bàn cờ vua kích thước $N \times N$ sao cho không có quân hậu nào có thể "ăn" được quân hậu khác theo quy tắc cờ vua. Màu của các quân hậu không có ý nghĩa trong bài toán này. Như vậy, lời giải của bài toán là một cách xếp tám quân hậu trên bàn cờ sao cho không có hai quân nào đứng trên cùng hàng, hoặc cùng cột hoặc cùng đường chéo.

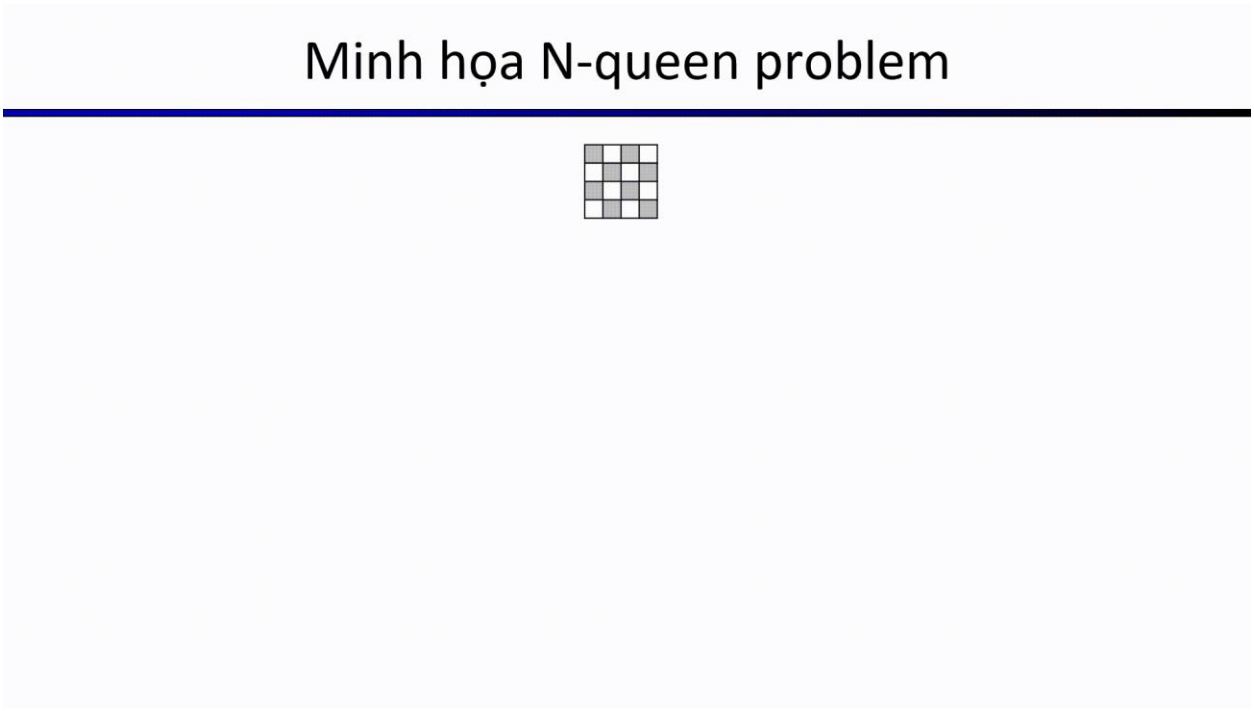


Ý tưởng giải:

Ta dùng mảng Q[1,2,…,n], trong đó Q[i]=j nếu quân hậu ở cột thứ i được đặt ở hàng j.

Cách này còn 2 lợi ích sau: Không có hai quân hậu nào cùng một hàng vì với mỗi hàng ii, chỉ có đúng một giá trị Q[i] và để kiểm tra hai quân hậu i≠j có cùng cột hay không, ta chỉ việc kiểm tra Q[i] và Q[j] có khác nhau hay không.

Kiểm tra đường chéo: Hai quân hậu đặt ở hai vị trí (i,j) và (a,b) ăn được nhau theo đường chéo khi và chỉ khi i−j=a−b hoặc i+j=a+b



Độ phức tạp: Nếu phân tích kĩ, ta sẽ thấy cây này chỉ có O(n!) nút. Ở mỗi nút của cây, ta dành thờ gian O(n) để kiểm tra tính hợp lệ của một cách đặt. Do đó tổng thời gian là O((n!)n).

b. Knight's Tour problem

Mã đi tuần hay hành trình của quân mã là bài toán về việc di chuyển một quân mã trên bàn cờ vua. Quân mã được đặt ở một ô trên một bàn cờ trống, nó phải di chuyển theo quy tắc của cờ vua để đi qua mỗi ô trên bàn cờ đúng một lần.

Ý tưởng giải:

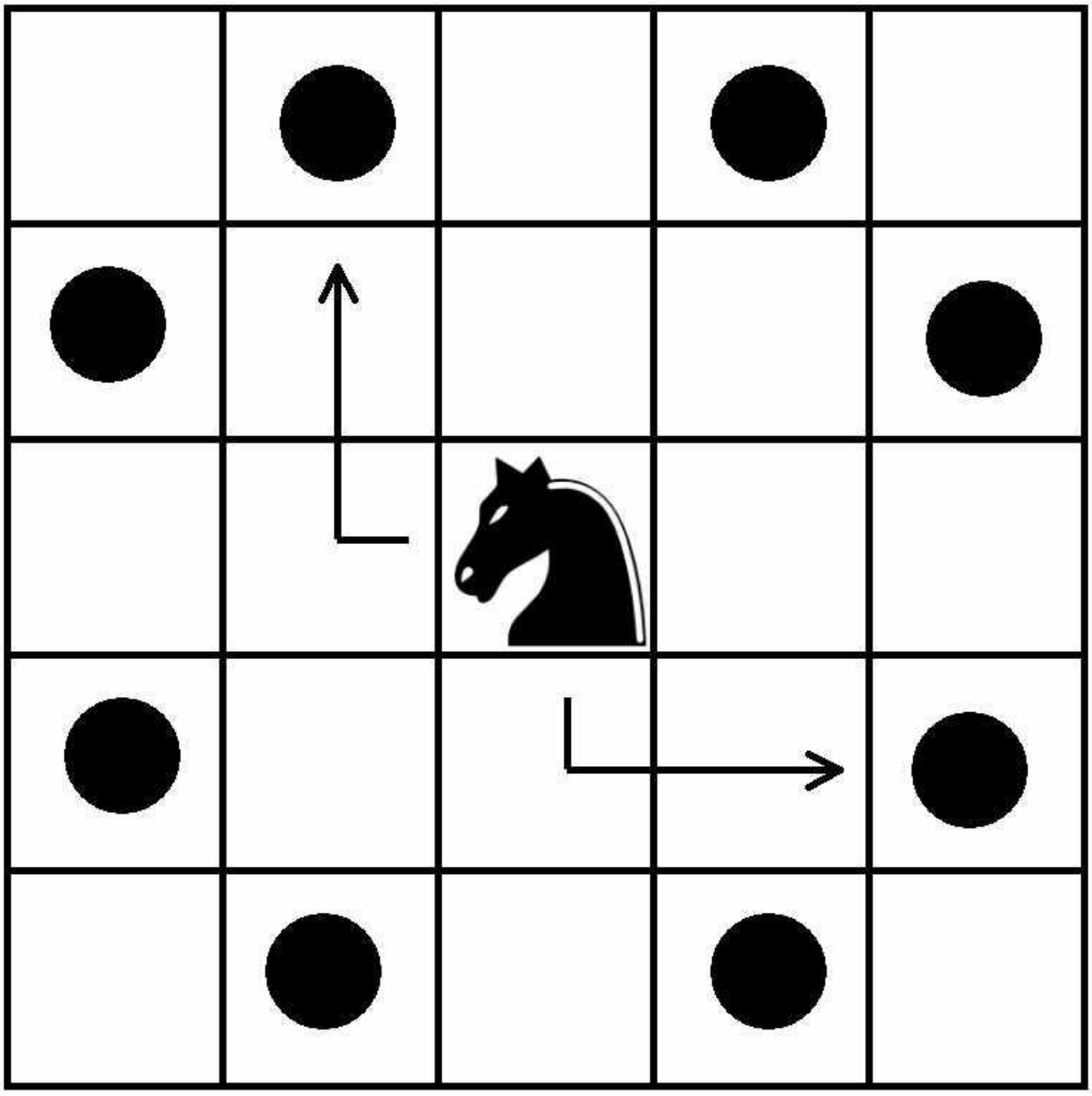
b.1 Xây dựng bước đi cho quân mã

Gọi x,y là độ dài bước đi trên các trục Oxy. Một bước đi hợp lệ của quân mã sẽ như sau: $|x| + |y| = 3$ (Với $x,y > 0$).

Khi đó ở một vị trí bất kì quân mã có có 8 đường có thể di chuyển. Chưa xét đến bước đi đó có hợp lệ hay không.

Các bước đi đó là: (2,1), (1,2), (-1,2), (-2,1), (-2,-1), (-1,-2), (1,-2), (2,-1)

Để đơn giản ta sẽ tạo hay mảng X[], Y[] để chứa các giá trị trên. Với mỗi X[i], Y[i] sẽ là một cách di chuyển của quân mã($0 \leq i \leq 7$).



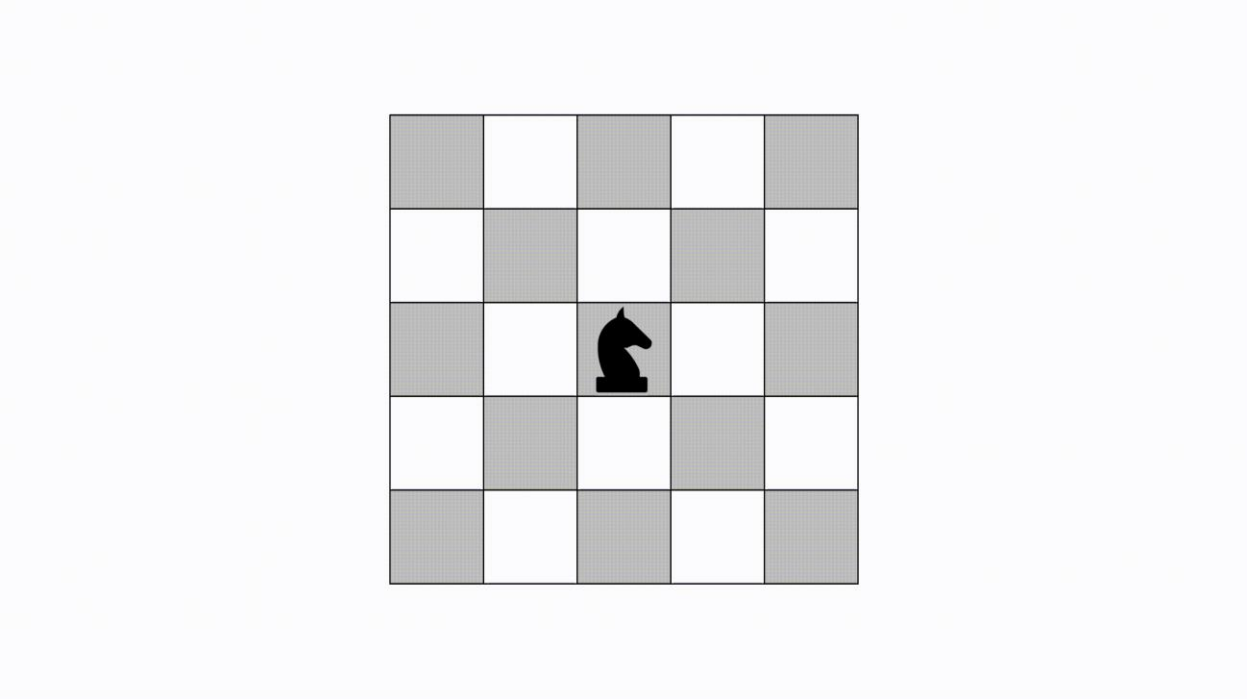
b.2 Kiểm tra tính hợp lệ của bước đi

Ta sẽ dùng một mảng hai chiều A[n*n] để lưu vị trí của từng ô trong bàn cờ. Tất cả mảng đều khởi tạo giá trị là -1 (quân mã chưa đi qua).

Giọi x,y là vị trí hiện tại của quân mã, thì vị trí tiếp theo mà quân mã đi sẽ có dạng x+X[i], y+Y[i]. Một vị trí được gọi là hợp lệ thì sẽ thỏa mãn tính chất sau:

- $0 \leq x+X[i] \leq n-1$.
- $0 \leq x+X[i] \leq n-1$.

Nếu bước đi đó là bước đi đúng thì ta sẽ lưu thứ tự của bước đi đó vào mảng A[x+X[i], y+Y[i]].



Độ phức tạp: bàn cờ có tổng cộng N^2 ô. Từ ô bắt đầu ta thực hiện đệ quy N nước đi, ứng với 8 ô con (là số nước đi của quân mã). Và cứ tiếp tục như vậy cho đến khi hết N^2 ô. Do đó độ phức tạp của thuật toán là $O(8^{N^2})$

5. **Nhận xét**

   Bản chất của quay lui là một quá trình tìm kiếm theo chiều sâu(Depth-First Search).

- Ưu điểm: Việc quay lui là thử tất cả các tổ hợp để tìm được một lời giải (vét cạn). Thế mạnh của phương pháp này là nhiều cài đặt tránh được việc phải thử nhiều trường hợp chưa hoàn chỉnh, nhờ đó giảm thời gian chạy.

- Nhược điểm: Trong trường hợp xấu nhất độ phức tạp của quay lui vẫn là cấp số mũ. Vì nó mắc phải các nhược điểm sau:
    - Rơi vào tình trạng "thrashing": quá trình tìm kiếm cứ gặp phải bế tắc với cùng một nguyên nhân.
    - Thực hiện các công việc dư thừa: Mỗi lần chúng ta quay lui, chúng ta cần phải đánh giá lại lời giải trong khi đôi lúc điều đó không cần thiết.
    - Không sớm phát hiện được các khả năng bị bế tắc trong tương lai. Quay lui chuẩn, không có cơ chế nhìn về tương lai để nhận biết đc nhánh tìm kiếm sẽ đi vào bế tắc.