1. **Ý tưởng tổng quát:**

“The principal idea is to construct solutions one component at a time and evaluate such partially constructed candidates as follows. If a partially constructed solution can be developed further without violating the problem’s constraints, it is done by taking the first remaining legitimate option for the next component. If there is no legitimate option for the next component, no alternatives for any remaining component need to be considered. In this case, the algorithm backtracks to replace the last component of the partially constructed solution with its next option.” (tr 424)

* Quay lui là một kĩ thuật thiết kế dựa trên đệ quy. Thường dùng để giải các bài toán liệt kê cấu hình. Mỗi cấu hình được xây dựng theo từng phần tử. Mỗi phần tử được chọn theo các khả năng.
* Ý tưởng là xây dựng solution theo từng thành phần dựa trên việc đánh giá khả năng của thành phần đó. Nếu thành phần của solution hiện tại có thể phát triển hơn mà không vi phạm các ràng buộc, nó sẽ kết hợp tiếp với thành phần hợp lệ tiếp theo. Nếu không có thành phần tiếp theo nào hợp lệ, nó sẽ quay lại solution hiện tại để xem xét các thành phần tiếp theo.
* Standard problems on Backtracking:

+ NQueen

+ SubSet Sum

+ Hamilton circuit

+ Sudoku

+ Permutations….

**2. State-space tree:**

Để thuận tiện thực hiện loại xử lý này ta xây dựng một cây các lựa chọn được thực hiện, được gọi là state-space tree. Gốc của nó đại diện cho trạng thái ban đầu trước khi bắt đầu tìm kiếm giải pháp. Các nút của cấp độ đầu tiên trong cây đại diện cho các lựa chọn được thực hiện cho thành phần đầu tiên của giải pháp, các nút của cấp độ thứ hai đại diện cho các lựa chọn cho thành phần thứ hai, v.v. Một nút trong state-space tree được cho là hứa hẹn nếu nó tương ứng với một giải pháp được xây dựng một phần mà vẫn có thể dẫn đến một giải pháp hoàn chỉnh; mặt khác, nó được gọi là không hứa hẹn. Các lá đại diện cho các ngõ cụt không hứa hẹn hoặc các giải pháp hoàn chỉnh mà thuật toán tìm thấy. Trong phần lớn các trường hợp, cây không gian trạng thái cho thuật toán quay lui được xây dựng theo cách tìm kiếm theo chiều sâu. Nếu nút hiện tại có triển vọng, nút con của nó được tạo bằng cách thêm tùy chọn hợp pháp còn lại đầu tiên cho thành phần tiếp theo của giải pháp và quá trình xử lý sẽ chuyển sang nút con này. Nếu nút hiện tại hóa ra không hứa hẹn, thuật toán sẽ quay lại nút cha của nút để xem xét tùy chọn khả thi tiếp theo cho thành phần cuối cùng của nó; nếu không có tùy chọn như vậy, nó sẽ quay ngược lại một cấp độ nữa của cây, v.v. Cuối cùng, nếu thuật toán đạt được một giải pháp hoàn chỉnh cho vấn đề, nó sẽ dừng lại (nếu chỉ cần một giải pháp) hoặc tiếp tục tìm kiếm các giải pháp khả thi khác.

1. **Ưu nhược điểm:**

* Ưu điểm:

+ Mang bản chất của brute force nên backtracking có thể giải quyết cặn kẽ vấn đề.

+ Trong khi thử tất cả tổ hợp để tìm solution, có thể tránh thử các solution không hoàn chỉnh, giảm thời gian thực thi. (so với brute force)

+ Các vấn đề của backtracking có thể trình bày một cách trực quan dễ hiểu và dễ cài đặt.

* Nhược điểm:

+ Vẫn khá chậm so với các phương pháp giải khác, độ phức tạp về thời gian theo cấp số nhân khiến việc giải quyết các vấn đề lớn trở nên không hợp lý.

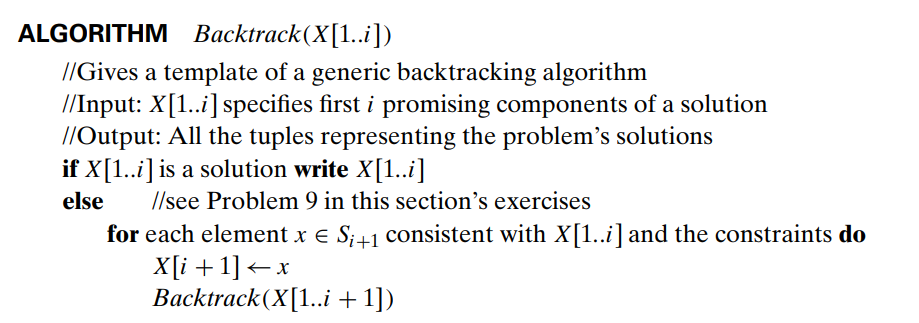
+ Có thể bị kẹt trong vòng lặp vô hạn nếu các rang buộc không được triễn khai đúng cách.

+ Thực hiện nhiều công việc dư thừa trong quá trình quay lui.

+ Không sớm phát hiện được các khả năng dẫn đến bế tắc trong tương lai.

+ Do việc lưu trữ ngăn xếp nên đòi hỏi chi phí tính toán, sử dụng nhiều bộ nhớ => Độ phức tạp không gian cao.

1. **Thuật toán:**



Mã giả cho thuật toán quay lui:

Backtracking(k)

{

for([Mỗi phương án chọn i(thuộc tập D)])

{

if([chập nhận i])

{

Chọn i cho X[k];

If ([thành công])

{

[Đưa ra kết quả];

}

Else

{

Backtracking(k+1);

[Bỏ chọn i cho X[k]];

}

}

}

}

1. **Phân loại thuật toán quay lui:**

Có 3 loại vấn đề trong quay lui:

* Decision Problem (Tìm kiếm một giải pháp khả thi): Trong bài toán N quân hậu, chúng ta phải đặt N quân hậu trên bàn cờ NxN sao cho không có hai quân hậu nào tấn công lẫn nhau.
* Optimization problem (Tìm kiếm giải pháp tốt nhất): Trong bài toán Knapsnack, kẻ trộm phải bỏ vào một cái túi những món đồ nào sao cho tổng trọng lượng của các mặt hàng không vượt quá trọng lượng tối đa có thể mang và tổng giá trị của các mặt hàng là tối đa và tổng giá trị lấy được phải lớn nhất.
* Enumeration Problem (Tìm kiếm tất cả các giải pháp khả thi): Trong bài toán liệt kê hoán vị, chúng ta phải tìm tất cả các hoán vị có thể có của một tập hợp số đã cho.

1. **Một số bài toán minh họa:**

* **N\_Queen:**

+ Tóm tắt: Cho bàn cờ vuông có kích thước NxN. Tìm cách đặt n quân hậu sao cho không có 2 quân nào cùng nằm trên một hàng hoặc một cột hoặc một đường chéo.

+ Độ phức tạp: O(N!)

+ Vì n<3 sẽ không tồn tại lời giải, trường hợp đơn giản nhất ta xét n = 4.

Diagram, engineering drawing

Description automatically generated

* **Hamilton Circuit:**

+ Tóm tắt: Cho đồ thị G = (V,E) có n đỉnh. Tìm chu trình Hamilton đi qua tất cả các đỉnh, mỗi đỉnh một lần và quay về đỉnh xuất phát.

+ Độ phức tạp: O(N!)

A picture containing necklet, accessory, locket

Description automatically generated

* **DFS:**

+ Tóm tắt: duyệt đồ thị theo chiều sâu.

+ Độ phức tạp: O(N,M)

A picture containing text, watch

Description automatically generated

Text

Description automatically generated

1. **Một số lưu ý giảm bớt kích thước và thời gian:**

* Khai thác tính đối xứng trong bài toán tổ hợp.
* Gán trước giá trị cho thành phần của solution (Hamilton)
* Branch and bound.