**CS112-Backtracking**

**Nhóm 8**

Hoàng Ngọc Quân & Trần Thị Cẩm Giang

**BTVN**: Phân tích và so sánh ưu điểm và nhược điểm giữa 2 thuật toán Backtracking vs Brute-force và chỉ rõ các ưu/nhược điểm trên trong các trường hợp cụ thể.

Bài làm:

Thuật toán Backtracking và Brute-force đều là hai phương pháp được sử dụng trong việc giải quyết các bài toán **tìm kiếm và liệt kê các giải pháp**. Mỗi thuật toán có ưu và nhược điểm khác nhau.

# Thuật toán brute-force:

Thuật toán Brute-force là một thuật toán tìm kiếm tất cả các *lời giải khả thi* của một bài toán. Thuật toán Brute-force bắt đầu với *tất cả các lời giải khả thi* của bài toán và sau đó loại bỏ các lời giải *không hợp lệ* (không thỏa mãn các ràng buộc của bài toán).

* Ưu điểm của thuật toán Brute-force:
* **Dễ hiểu và dễ triển khai**: đơn giản, không đòi hỏi nhiều kiến thức về thuật toán và dễ dàng viết code.
* Có thể **áp dụng cho nhiều loại bài toán khác nhau**.
* **Chắc chắn tìm được giải pháp**: Vì nó kiểm tra tất cả các lời giải khả thi nên nó đảm bảo tìm ra lời giải thỏa mãn (nếu tồn tại).
* Nhược điểm của thuật toán Brute-force:
* **Độ phức tạp cao**: vì kiểm tra tất cả các lời giải, dẫn đến độ phức tạp cao trong nhiều trường hợp.
* Có thể **tốn nhiều thời gian và bộ nhớ** khi bài toán có nhiều lời giải khả thi.
* Ví dụ:
* Bài toán tìm số nguyên nhỏ nhất: Thuật toán Brute-force có thể được sử dụng để tìm số nguyên nhỏ nhất trong một dãy số bằng cách duyệt qua tất cả các số trong dãy và lưu lại số nhỏ nhất.
* Bài toán tìm số nguyên tố: Thuật toán Brute-force có thể được sử dụng để tìm số nguyên tố bằng cách duyệt qua tất cả các số nguyên và kiểm tra xem số đó có phải là số nguyên tố hay không.

# Thuật toán Backtracking:

Thuật toán Backtracking là một thuật toán tìm kiếm đệ quy, bắt đầu với một *lời giải rỗng* và *mở rộng lời giải từng bước*. Nếu lời giải hiện tại là không hợp lệ, thuật toán sẽ quay lui và thử một lựa chọn khác. Thuật toán Backtracking đảm bảo tìm thấy tất cả các lời giải thỏa mãn của một bài toán.

* Ưu điểm của thuật toán Backtracking:
* **Đảm bảo tìm thấy tất cả các lời giải thỏa mãn của bài toán**.
* Có thể áp dụng cho nhiều loại bài toán khác nhau.
* Có thể được sử dụng để **giải các bài toán có cấu trúc cây**.
* **Tối ưu hóa quá trình tìm kiếm**: Quay lui kiểm tra ràng buộc ngầm sau mỗi lựa chọn, không chờ đến khi tất cả các lời giải đã được tạo ra. Nếu một lời giải không thỏa mãn ràng buộc, nó có thể bị loại bỏ ngay lập tức.
* **Tiết kiệm thời gian và tài nguyên**: Không cần xây dựng tất cả các lời giải trước khi kiểm tra ràng buộc.
* Nhược điểm của thuật toán Backtracking:
* Có thể **tốn nhiều thời gian và bộ nhớ** khi bài toán có nhiều lời giải khả thi.
* Có thể **khó viết code cho thuật toán Backtracking**.
* **Nguy cơ bị kẹt**: Quay lui có thể gặp tình trạng tràn stack.
* Ví dụ:
* Bài toán 8 quân hậu: Thuật toán Backtracking có thể được sử dụng để giải bài toán 8 quân hậu bằng cách đặt từng quân hậu lên bàn cờ theo từng hàng. Nếu một vị trí đặt quân hậu không hợp lệ, thuật toán sẽ quay lui và thử một vị trí khác. Thuật toán Backtracking đảm bảo tìm thấy tất cả các cách đặt 8 quân hậu trên bàn cờ mà không có quân nào cùng hàng, cùng cột hoặc cùng đường chéo.
* Bài toán tìm đường đi trong mê cung: Thuật toán Backtracking có thể được sử dụng để tìm đường đi trong mê cung bằng cách bắt đầu từ điểm đầu và thử tất cả các hướng đi có thể. Nếu một hướng đi không dẫn đến đích, thuật toán sẽ quay lui và thử hướng đi khác. Thuật toán Backtracking đảm bảo tìm thấy một đường đi từ điểm đầu đến đích nếu có đường đi tồn tại.

# Ưu và nhược điểm trong các trường hợp cụ thể:

* Các trường hợp bài toán có số lượng các lựa chọn khả thi không quá lớn, thuật toán Brute-force và thuật toán Backtracking đều có thể giải được bài toán. Tuy nhiên, thuật toán Backtracking có thể hiệu quả hơn thuật toán Brute-force. Lý do là thuật toán Backtracking chỉ cần duyệt qua các lựa chọn khả thi có thể dẫn đến lời giải. Trong khi đó, thuật toán Brute-force cần duyệt qua tất cả các lựa chọn khả thi, kể cả các lựa chọn không thể dẫn đến lời giải.
* Các trường hợp bài toán có số lượng các lựa chọn khả thi lớn, thuật toán Brute-force có thể không khả thi. Lý do là thuật toán Brute-force cần duyệt qua tất cả các lựa chọn khả thi, kể cả các lựa chọn không thể dẫn đến lời giải. Điều này có thể tốn kém thời gian và bộ nhớ.
* Các trường hợp bài toán mà các thuật toán khác không thể giải được, thuật toán Brute-force có thể là lựa chọn duy nhất.