Lớp: CS112.P11.CTTN
Nhóm: 14
Sinh viên: Lê Nguyễn Anh Khoa. MSSV: 23520742
Sinh viên: Cáp Kim Hải Anh. MSSV: 23520036

BÀI TẬP
Phương pháp thiết kế thuật toán Completed search - Brute force, Backtracking, Branch and Bound

I. Bài 1:

Câu hỏi 1: Trình bày nguyên lý cơ bản của thuật toán quay lui (Backtracking). Tại sao thuật toán này thường được sử dụng để giải các bài toán tổ hợp?

Thuật toán quay lui (Backtracking) là một kỹ thuật thiết kế thuật toán để tìm lời giải bằng cách xây dựng từng phần của lời giải và loại bỏ các khả năng không thỏa mãn yêu cầu.

Nguyên lý cơ bản:

1. **Xây dựng lời giải từng bước:**

- Chia bài toán thành nhiều bước nhỏ
- Tại mỗi bước, thử các khả năng có thể xảy ra
- Kiểm tra tính hợp lệ của từng khả năng

2. **Quay lui khi gặp trường hợp không khả thi:**

- Nếu một khả năng không thỏa mãn → bỏ qua và thử khả năng tiếp theo
- Nếu đã thử hết các khả năng → quay lại bước trước đó
- Quá trình lặp lại cho đến khi tìm được lời giải hoặc kết luận không có lời giải

Thuật toán thường được dùng cho bài toán tổ hợp vì:

1. **Không gian tìm kiếm lớn:**

- Bài toán tổ hợp thường có số lượng cách chọn rất lớn
- Quay lui giúp duyệt có chọn lọc, không cần thử tất cả các trường hợp

2. **Ràng buộc phức tạp:**

- Các bài toán tổ hợp thường có nhiều điều kiện cần thỏa mãn
- Quay lui cho phép kiểm tra và loại bỏ sớm các nhánh không khả thi

3. **Cấu trúc bài toán phù hợp:**

- Có thể xây dựng lời giải theo từng bước
- Dễ xác định điều kiện dừng và điều kiện hợp lệ

Câu hỏi 2: So sánh điểm khác biệt chính giữa thuật toán nhánh cận (Branch and Bound) và quay lui (Backtracking) khi tìm kiếm lời giải tối ưu.

1. **Mục tiêu và phạm vi**

- Nhánh cận: Tập trung tìm lời giải tối ưu (min/max) trong không gian các lời giải khả thi
- Quay lui: Tìm tất cả hoặc một lời giải thỏa mãn các ràng buộc đã cho

2. **Cách đánh giá nhánh**

- Nhánh cận: Sử dụng hàm cận (bound) để ước lượng giá trị tối ưu có thể đạt được
- Quay lui: Chỉ kiểm tra tính hợp lệ của từng phần lời giải theo ràng buộc

3. **Chiến lược cắt tỉa**

- Nhánh cận: Loại bỏ các nhánh dựa trên giá trị cận và lời giải tốt nhất hiện tại
- Quay lui: Chỉ loại bỏ nhánh khi vi phạm ràng buộc của bài toán

4. **Hiệu quả tìm kiếm**

- Nhánh cận: Thường hiệu quả hơn cho bài toán tối ưu vì tận dụng thông tin cận
- Quay lui: Có thể phải duyệt nhiều nhánh không cần thiết khi tìm lời giải tối ưu

5. **Cấu trúc dữ liệu**

- Nhánh cận: Thường dùng hàng đợi ưu tiên để chọn nhánh triển khai tiếp theo
- Quay lui: Thường dùng đệ quy và stack để quản lý quá trình tìm kiếm

6. **Ứng dụng phù hợp**

- Nhánh cận: Bài toán tối ưu tổ hợp như TSP, Knapsack
- Quay lui: Bài toán liệt kê như N-Queens, Sudoku

Câu hỏi 3: Trình bày ưu điểm và nhược điểm của phương pháp Brute Force. Tại sao nó thường được xem là phương pháp kém hiệu quả trong các bài toán lớn?

Phương pháp Brute Force (vét cạn) có các ưu điểm và nhược điểm như sau:

Ưu điểm:

1. **Tính đơn giản**

- Dễ hiểu và cài đặt
- Ít rủi ro lỗi logic
- Phù hợp để kiểm chứng các thuật toán khác

2. **Tính chính xác**

- Đảm bảo tìm được lời giải tối ưu
- Không bỏ sót trường hợp nào
- Phù hợp cho bài toán cần độ chính xác tuyệt đối

3. **Khả năng áp dụng**

- Có thể áp dụng cho hầu hết các bài toán
- Không đòi hỏi kiến thức chuyên sâu
- Làm cơ sở để phát triển thuật toán tốt hơn

Nhược điểm:

1. **Độ phức tạp thời gian**

- Thường có độ phức tạp lớn (O(n!), O(2^n))

- Tốc độ chậm với dữ liệu lớn
- Không khả thi với bài toán quy mô lớn

2. **Tốn tài nguyên**
- Tiêu tốn nhiều bộ nhớ
- Sử dụng CPU không hiệu quả
- Lãng phí tài nguyên hệ thống

3. **Thiếu tối ưu**
- Không tận dụng đặc điểm của bài toán
- Không áp dụng được các kỹ thuật cắt tỉa
- Xử lý cả các trường hợp hiển nhiên không cần thiết

Kém hiệu quả trong bài toán lớn vì:

1. **Tăng trưởng theo hàm mũ**
- Thời gian xử lý tăng đột biến khi n tăng
- Không thể xử lý trong thời gian chấp nhận được
- Ví dụ: với n=50, 2^50 là một số rất lớn

2. **Giới hạn thực tế**
- Vượt quá giới hạn phần cứng
- Không đáp ứng yêu cầu thời gian thực
- Tốn kém chi phí vận hành

3. **Có các giải pháp thay thế tốt hơn**
- Thuật toán tối ưu hơn đã được phát triển
- Các phương pháp tiếp cận thông minh hơn
- Kỹ thuật tối ưu và cắt tỉa hiệu quả