ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

PHÂN TÍCH VÀ THIẾT KẾ THUẬT TOÁN
CS112.P11.KHTN

BÀI TẬP NHÓM 12

Sinh viên :
Nguyễn Văn Hồng Thái - 23521418
Hoàng Đức Dũng - 23520328

Giảng viên :
Nguyễn Thanh Sơn

Ngày 5 tháng 12 năm 2024

Mục lục

Bài tập 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?

Nguyên lý cơ bản của thuật toán quay lui: Thuật toán quay lui (Backtracking) là một phương pháp tìm kiếm dựa trên việc xây dựng lời giải theo từng bước, kiểm tra tính khả thi của mỗi bước, và nếu phát hiện lời giải hiện tại không thỏa mãn điều kiện, thuật toán sẽ "quay lui" để thử một nhánh khác. Nguyên lý cơ bản gồm các bước:

- **Xây dựng từng phần lời giải:** Từng bước chọn một phần tử hoặc quyết định để thêm vào lời giải.
- **Kiểm tra tính khả thi:** Sau mỗi bước, kiểm tra xem lời giải hiện tại có còn phù hợp với ràng buộc của bài toán hay không.
- **Quay lui:** Nếu lời giải không phù hợp, loại bỏ bước vừa chọn và quay lại bước trước đó để thử nhánh khác.
- **Lưu lời giải hoàn chỉnh:** Nếu tìm được lời giải thỏa mãn toàn bộ điều kiện, lưu lại và tiếp tục tìm các lời giải khác (nếu cần).

Lý do thuật toán Backtracking phù hợp cho các bài toán tổ hợp:

- **Tính toàn diện:** Backtracking duyệt qua tất cả các khả năng để đảm bảo không bỏ sót trường hợp nào.
- **Giảm thiểu không gian tìm kiếm:** Bằng cách kiểm tra tính khả thi ngay tại mỗi bước, thuật toán loại bỏ sớm các nhánh không cần thiết, giúp giảm số lượng trường hợp phải thử.
- **Ứng dụng phổ biến:** Các bài toán tổ hợp như liệt kê tổ hợp, hoán vị, bài toán N-queens, và bài toán phân chia thường có không gian tìm kiếm lớn, nhưng Backtracking có thể giúp giảm số lượng tính toán bằng cách loại bỏ sớm các trường hợp không 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.

Điểm giống nhau:

- Cả hai đều là phương pháp tìm kiếm có hệ thống trên không gian trạng thái của bài toán.
- Đều sử dụng chiến lược loại bỏ các nhánh không cần thiết để giảm bớt không gian tìm kiếm.

Điểm khác biệt chính:

Tiêu chí	Backtracking
Mục tiêu chính	Tìm tất cả lời giải hoặc lời giải hợp lệ đầu tiên.
Tiêu chí loại bỏ nhánh	Kiểm tra tính khả thi dựa trên ràng buộc của bài toán.
Tìm kiếm lời giải	Thử toàn bộ các khả năng theo cách giảm thiểu không gian tìm kiếm.
Tối ưu hóa	Không có cơ chế tối ưu hóa cụ thể.
Ứng dụng	Bài toán tổ hợp (như liệt kê, phân chia).

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: Brute Force là phương pháp thử tất cả các khả năng có thể có để tìm ra lời giải cho bài toán. Không sử dụng thuật toán hay chiến lược phức tạp, Brute Force đảm bảo tìm ra lời giải đúng bằng cách kiểm tra từng trường hợp.

Ưu điểm:

- **Đơn giản và dễ hiểu:** Không yêu cầu thiết kế thuật toán phức tạp hay hiểu sâu về bài toán.
- **Chính xác:** Nếu kiểm tra hết tất cả các trường hợp, phương pháp này chắc chắn tìm ra lời giải đúng.
- **Tổng quát:** Có thể áp dụng cho nhiều bài toán khác nhau mà không cần điều chỉnh nhiều.

Nhược điểm:

- **Kém hiệu quả:**
  - Số lượng trường hợp cần kiểm tra tăng theo cấp số nhân hoặc giai thừa, khiến thời gian tính toán trở nên quá lớn với các bài toán lớn.
  - Ví dụ: Bài toán hoán vị N phần tử có $N!$ trường hợp, với $N = 10$, số trường hợp đã là $3,628,800$.
- **Tốn tài nguyên:** Yêu cầu rất nhiều thời gian và bộ nhớ để xử lý các bài toán phức tạp.
- **Không thực tế:** Với các bài toán lớn, Brute Force không thể hoàn thành trong thời gian hợp lý.

Lý do Brute Force kém hiệu quả trong các bài toán lớn:

- **Độ phức tạp cao:** Đối với các bài toán có không gian tìm kiếm lớn, độ phức tạp của Brute Force thường là $O(n!)$, $O(2^n)$, hoặc $O(n^k)$, không khả thi khi n hoặc k tăng.
- **Không loại bỏ nhánh vô nghĩa:** Brute Force không có cơ chế loại bỏ sớm các trường hợp không khả thi, dẫn đến việc tốn công xử lý những trường hợp không cần thiết.

Ví dụ minh họa:

- **Bài toán N-queens:** Với Brute Force, phải thử tất cả các hoán vị của N quân hậu, sau đó kiểm tra xem mỗi trường hợp có hợp lệ hay không. Trong khi đó, Backtracking loại bỏ sớm các trường hợp đặt hậu không hợp lệ.
- **Bài toán người du lịch:** Với N thành phố, Brute Force phải kiểm tra tất cả $N!$ cách đi, trong khi Branch and Bound giảm bớt rất nhiều nhánh không tối ưu.

Bài tập 2

Source code:

- Link Github: Click here.