VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
UNIVERSITY OF INFORMATION TECHNOLOGY

CS112

BÀI TẬP VỀ NHÀ NHÓM 12

Sinh Viên :
Nguyễn Văn Minh
Đồng Quốc Thắng

Giảng viên :
Nguyễn Thanh Sơn

Ngày 5 tháng 12 năm 2024

Mục lục

Chương 1

Bài tập 1

1.1 Trình bày nguyên lý cơ bản của thuật toán quay lui. 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 (Backtracking) như sau:
Thuật toán quay lui là một phương pháp tìm kiếm giải pháp trong không gian bài toán thông qua việc xây dựng dần dần một giải pháp tiềm năng, kiểm tra tính hợp lệ của nó, và quay lại (backtrack) khi gặp một trạng thái không hợp lệ hoặc không thể tiếp tục tìm kiếm.

Cơ chế hoạt động:

- **Tiến tới (Forward)**: Bắt đầu từ một trạng thái ban đầu và xây dựng giải pháp từng bước.
- **Kiểm tra tính hợp lệ**: Sau mỗi bước, kiểm tra xem giải pháp hiện tại có hợp lệ không (ví dụ: có thỏa mãn các ràng buộc của bài toán hay không).
- **Quay lui (Backtrack)**: Nếu tại bất kỳ bước nào, giải pháp không thể tiếp tục hoặc không hợp lệ, quay lại bước trước đó để thử một lựa chọn khác.
- **Tìm kiếm tiếp tục**: Quá trình này tiếp tục cho đến khi tìm được giải pháp hợp lệ hoặc khám phá hết các khả năng.

Lý do thuật toán quay lui thường được sử dụng để giải các bài toán tổ hợp:

- **Bài toán tổ hợp** thường có không gian tìm kiếm rộng lớn, trong đó các lựa chọn có thể được thử nghiệm theo từng bước. Quay lui là một phương pháp hiệu quả vì:
- **Loại bỏ các lựa chọn không hợp lệ sớm**: Quay lui giúp giảm thiểu số lượng thử nghiệm không cần thiết bằng cách dừng lại khi gặp lựa chọn không hợp lệ.
- **Xây dựng từng phần của giải pháp và quay lại khi cần**: Điều này đặc biệt quan trọng trong các bài toán tổ hợp, nơi bạn cần tìm các tập hợp, phân phối, hoặc hoán vị hợp lệ.

1.2 So sánh điểm khác biệt chính giữa thuật toán nhánh cận và quay lui khi tìm kiếm lời giải tối ưu

1.2.1 Khái niệm

- **Quay lui (Backtracking)**: Là phương pháp tìm kiếm thử nghiệm các giải pháp từng bước và quay lại khi gặp giải pháp không hợp lệ. Quay lui không nhất thiết tìm kiếm lời giải tối ưu mà chỉ cần tìm ra một giải pháp hợp lệ.
- **Nhánh cận (Branch and Bound)**: Là phương pháp tối ưu hóa, trong đó bạn chia không gian tìm kiếm thành các nhánh (branches) và tính toán các "bound"(giới hạn) để loại bỏ các nhánh không thể cho ra giải pháp tối ưu. Nhánh cận luôn tìm kiếm lời giải tối ưu trong không gian bài toán.

1.2.2 Tính tối ưu

- **Backtracking**: Chỉ tìm kiếm giải pháp hợp lệ, không tối ưu, trong khi **Branch and Bound** luôn tìm kiếm giải pháp tối ưu bằng cách sử dụng thông tin về các giới hạn (bounds) để loại bỏ các nhánh không cần thiết.
- **Branch and Bound**: Luôn tìm kiếm giải pháp tối ưu bằng cách sử dụng thông tin về các giới hạn (bounds) để loại bỏ các nhánh không có khả năng chứa lời giải tối ưu.

1.2.3 Kỹ thuật loại bỏ

- **Backtracking**: Loại bỏ các giải pháp không hợp lệ hoặc không thể tiếp tục xây dựng.
- **Branch and Bound**: Loại bỏ các nhánh có khả năng không tìm được lời giải tốt hơn giải pháp đã tìm được.

1.2.4 Phạm vi áp dụng

- **Backtracking**: Thường dùng cho các bài toán tìm kiếm và tổ hợp (ví dụ: tìm kiếm tập hợp con, phân phối, hoán vị).
- **Branch and Bound**: Thường dùng cho các bài toán tối ưu hóa, đặc biệt là các bài toán tìm kiếm lời giải tối ưu trong không gian lớn, như bài toán đi thăm tất cả các thành phố (TSP), bài toán đóng gói, v.v.

Tóm lại, **Backtracking** chỉ tìm một giải pháp hợp lệ, không tối ưu, trong khi **Branch and Bound** đảm bảo tìm giải pháp tối ưu bằng cách loại bỏ các nhánh không cần thiết dựa trên giới hạn.

1.3 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?

1.3.1 Ưu điểm của phương pháp Brute Force

- **Đơn giản và dễ hiểu**: Phương pháp brute force rất dễ triển khai vì nó chỉ cần thử tất cả các khả năng mà không cần sử dụng các chiến lược phức tạp.
- **Đảm bảo đúng kết quả**: Vì brute force thử tất cả các khả năng có thể, nó luôn đảm bảo tìm ra giải pháp đúng cho bài toán.

1.3.2 Nhược điểm của phương pháp Brute Force

- **Không hiệu quả về thời gian**: Phương pháp brute force có độ phức tạp thời gian rất lớn vì phải thử tất cả các khả năng có thể. Ví dụ, trong bài toán tổ hợp, số lượng các khả năng có thể cực kỳ lớn.
- **Không thể áp dụng cho bài toán lớn**: Khi kích thước bài toán tăng lên, số lượng các giải pháp tiềm năng mà brute force cần phải kiểm tra tăng lên theo cấp số mũ, làm cho phương pháp này không khả thi trong các bài toán lớn.
- **Không tận dụng các tính chất của bài toán**: Brute force không có chiến lược tối ưu nào để giảm số lượng phép thử, mà chỉ đơn giản thử tất cả các khả năng.

1.3.3 Tại sao phương pháp brute force là kém hiệu quả trong các bài toán lớn

- Khi kích thước bài toán tăng lên (ví dụ, số lượng biến hoặc lựa chọn trong bài toán tổ hợp), số lượng phép thử mà brute force phải kiểm tra tăng theo cấp số mũ (hoặc thậm chí theo giai thừa trong một số trường hợp). Điều này khiến cho phương pháp brute force trở nên không khả thi cho các bài toán có kích thước lớn hoặc yêu cầu tìm lời giải trong thời gian ngắn.
- Với các bài toán lớn, các phương pháp tối ưu hơn như backtracking, dynamic programming, hoặc branch and bound thường được ưa chuộng vì chúng có thể giảm đáng kể không gian tìm kiếm và thời gian tính toán.

Chương 2

Bài tập 2

Source code có ở đây