CS112.P11.KHTN - Bài tập nhóm 9

Nhóm 3
Nguyễn Hữu Đặng Nguyên - 23521045
Trần Vạn Tấn - 23521407

Ngày 5 tháng 12 năm 2024

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 (Backtracking):

Thuật toán quay lui là một phương pháp thử và sai có hệ thống để giải các bài toán liên quan đến việc lựa chọn các tập hợp con từ một không gian tìm kiếm lớn. Ý tưởng của quay lui là tìm lời giải từng bước, mỗi bước chọn một trong số các lựa chọn khả dĩ và đệ quy. Nguyên lý cơ bản như sau:

- Bắt đầu từ trạng thái ban đầu và cố gắng mở rộng lời giải từng bước.
- Tại mỗi bước, kiểm tra xem lựa chọn hiện tại có hợp lệ không:
  - Nếu hợp lệ: tiếp tục tiến thêm một bước.
  - Nếu không hợp lệ: quay lui về bước trước đó và thử lựa chọn khác.
- Quá trình tiếp tục cho đến khi tìm được lời giải hoặc thử hết mọi khả năng.

Ứng dụng trong bài toán tổ hợp:

Backtracking thường được sử dụng để giải các bài toán tổ hợp như bài toán liệt kê các tập hợp con, bài toán sắp xếp, bài toán tìm đường, bài toán n-queens, bài toán sudoku, ...vì:

- Không cần thiết phải xây dựng các thuật toán quá phức tạp. Không cần phải đi sâu phân tích chi tiết đặc điểm bài toán.
- Bài toán có nhiều trạng thái và cách chuyển đổi trạng thái đơn giản.

- Có thể cắt giảm không gian tìm kiếm thông qua điều kiện được cung cấp trong bài toán.
- Thử tất cả các phương án một cách có hệ thống, đảm bảo không bỏ sót kết qủa chính xác.

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 thuật toán đều duyệt không gian của bài toán một cách có hệ thống.
- Cả hai đều sử dụng phương pháp cắt tỉa để giảm kích thước không gian tìm kiếm và giảm đi kích thước bài toán.

Điểm khác biệt:

- **Backtracking:**
  - Được sử dụng cho các bài toán liệt kê hoặc tìm kiếm tất cả lời giải hợp lệ.
  - Không tập trung vào việc tối ưu hóa giá trị của lời giải.
  - Việc cắt tỉa dựa vào các điều kiện ràng buộc của bài toán.
  - Dễ cài đặt và triển khai. Đảm bảo đưa ra được lời giải mong muốn.
- **Branch and Bound:**
  - Được sử dụng nhiều trong bài toán tìm lời giải tối ưu cho các bài toán tối ưu hóa (như tìm giá trị lớn nhất, nhỏ nhất) hay xác định trạng thái hợp lệ của bài toán.
  - Sử dụng các nhánh (branch) để phân tách không gian tìm kiếm và cận trên/cận dưới, tính hợp lệ của trạng thái để không phải duyệt qua các trạng thái chắc chắn sai nhằm tối ưu chi phí.
  - Tập trung vào đánh giá và so sánh các cận để giảm thiểu số nhánh cần xét.
  - Khó cài đặt và suy nghĩ hơn so với Backtracking.

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?

Ưu điểm:

- Đơn giản và dễ cài đặt.
- Đảm bảo tìm được lời giải đúng nếu tồn tại.
- Độ chính xác cao, cách giải đơn giản, thường chỉ cần theo đúng yêu cầu bài toán, không yêu cầu hiểu biết phức tạp về bài toán.

Nhược điểm:

- Tốn nhiều thời gian do phải duyệt qua toàn bộ không gian tìm kiếm.
- Tốn nhiều bộ nhớ cho các bài toán cần lưu vết, bài toán lớn.
- Không tận dụng được đặc điểm đặc trưng hay tính chất của bài toán để tối ưu được thời gian và bộ nhớ.

Lý do kém hiệu quả trong bài toán lớn:

Brute Force thường kém hiệu quả trong các bài toán lớn do độ phức tạp thời gian khi dùng Brute Force thường cao, thường là hàm luỹ thừa, số mũ hay giai thừa. Vì thế nên khi n tăng, số lượng các bước tính toán cần xét tăng theo cấp số mũ, cấp số nhân dẫn đến thời gian tính toán quá lớn.

1 Bài tập: 24

Ý tưởng chính và các bước giải

1. Ý tưởng chính Bài toán yêu cầu tìm giá trị lớn nhất không vượt quá 24 bằng cách thực hiện các phép toán (+, -, ×, ÷) trên bốn thẻ bài. Để giải quyết bài toán này, ta cần kiểm tra tất cả các thứ tự sắp xếp và cách ghép các biểu thức với các phép toán, đồng thời đảm bảo các phép chia chỉ có kết quả nguyên.
2. Các bước giải

- Tạo hoán vị của các thẻ bài và Tạo tất cả tổ hợp các phép toán.
- Tạo các cấu trúc biểu thức: Sử dụng các cách ghép dấu ngoặc để xác định thứ tự thực hiện phép toán.
- Đánh giá biểu thức: Với mỗi biểu thức, tính toán giá trị, đảm bảo các phép chia chỉ có kết quả nguyên và giá trị không vượt quá 24. Sau đó cập nhật kết quả.

Mã giả

```
function find_max_under_24(cards):
    max_result = 0
    all_permutations = permute(cards)
    operators = ['+', '-', '*', '/']
    for perm in all_permutations:
        for op1 in operators:
            for op2 in operators:
                for op3 in operators:
                    results = evaluate(perm, op1, op2, op3)
                    for result in results:
                        if result <= 24:
                            max_result = max(max_result, result)

    return max_result

function evaluate(perm, op1, op2, op3):
    expressions = [
        (perm[0] op1 (perm[1] op2 (perm[2] op3 perm[3]))),
        ((perm[0] op1 perm[1]) op2 (perm[2] op3 perm[3])),
        ((perm[0] op1 (perm[1] op2 perm[2])) op3 perm[3]),
        (perm[0] op1 ((perm[1] op2 perm[2]) op3 perm[3])),
        (((perm[0] op1 perm[1]) op2 perm[2]) op3 perm[3])
    ]
    results = []
    for expr in expressions:
        result = eval(expr)
        results.append(result)
    return results
```

Phân tích độ phức tạp

- Hoán vị các thẻ bài: - Có $4! = 24$ hoán vị.
- Tổ hợp các phép toán: - Có $4^3 = 64$ cách sắp xếp ba phép toán từ tập $(+, -, \times, \div)$.
- Cấu trúc biểu thức: Có 5 cấu trúc biểu thức khác nhau với dấu ngoặc.
- Tổng số biểu thức:

$$24 \times 64 \times 5 = 7680 \text{ biểu thức cần kiểm tra.}$$

- Đánh giá mỗi biểu thức: O(1)
- Tổng độ phức tạp: O(7680)