CS112.P11.KHTN - Bài tập nhóm 7

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

Ngày 3 tháng 12 năm 2024

Mục lục

Bài 1: Tìm tổ hợp các chữ số

Đề bài: Cho một chuỗi số đầu vào, tìm tất cả các tổ hợp của các số có thể tạo thành.

Giải pháp: Sử dụng đệ quy để duyệt qua các chữ số trong chuỗi. Mỗi lần gọi đệ quy sẽ xét hai trường hợp:

1. Bao gồm chữ số hiện tại trong tổ hợp.
2. Bỏ qua chữ số hiện tại.

Độ phức tạp: $O(2^n)$, với n là độ dài chuỗi đầu vào.

Code:

```
def find_combinations(input, index, current, result):
    if index == len(input):
        if current:
            result.append(current)
        return

    # Không chọn chữ số hiện tại
    find_combinations(input, index + 1, current, result)

    # Chọn chữ số hiện tại
    find_combinations(input, index + 1, current + input[index], result)

input = "123"
result = []
```

```
find_combinations(input, 0, "", result)
print(result)
```

—

Bài 2: Sinh xâu độ dài k

Đề bài: Cho tập ký tự và số nguyên dương k, sinh tất cả các xâu độ dài k.

Giải pháp: Dùng đệ quy để xây dựng xâu từ tập ký tự. Khi độ dài xâu đạt k, in ra kết quả.
Độ phức tạp: $O(m^k)$, với m là số lượng ký tự trong tập.

Code:

```
def generate_strings(char_set, k, current):
    if len(current) == k:
        print(current)
        return

    for char in char_set:
        generate_strings(char_set, k, current + char)

char_set = ['a', 'b']
k = 2
generate_strings(char_set, k, "")
```

—

Bài 3: Tìm tổ hợp các ước số của n

Đề bài: Cho số n, in tất cả tổ hợp các ước số của n.

Giải pháp: Dùng đệ quy để phân tích các ước số. Chỉ xem xét các ước từ *start* đến $\sqrt{n}$ để tránh lặp lại.
Độ phức tạp: $O(2^{\sqrt{n}})$ trong trường hợp xấu nhất.

Code:

```
def factor_combinations(n, start, current, result):
    if n == 1 and len(current) > 1:
        result.append(list(current))
        return

    for i in range(start, n + 1):
```

```
        if n % i == 0:
            current.append(i)
            factor_combinations(n // i, i, current, result)
            current.pop()

n = 12
result = []
factor_combinations(n, 2, [], result)
print(result)
```

—

Bài 4: Biểu diễn x thành tổng của các số mũ

Đề bài: Với x và n, tìm số cách biểu diễn x thành tổng của n-th lũy thừa của các số tự nhiên duy nhất.

Giải pháp: Dùng đệ quy để kiểm tra từng số mũ k^n từ 1 đến $\sqrt[n]{x}$. Tránh chọn lại số đã sử dụng.
Độ phức tạp: $O(x^{1/n})$.

Code:

```
def power_sum(x, n, current):
    if x == 0:
        return 1
    if x < 0:
        return 0

    total = 0
    for i in range(current, int(x**(1/n)) + 1):
        total += power_sum(x - i**n, n, i + 1)
    return total

x = 10
n = 2
print(power_sum(x, n, 1))
```

—

Bài 5: Tháp Hà Nội

Đề bài: Với n đĩa, di chuyển toàn bộ từ cọc A sang cọc C theo quy tắc:

1. Chỉ được di chuyển một đĩa mỗi lần.
2. Đĩa lớn không được nằm trên đĩa nhỏ hơn.

Giải pháp: Dùng đệ quy:

1. Di chuyển $n-1$ đĩa từ A sang B.
2. Di chuyển đĩa lớn nhất từ A sang C.
3. Di chuyển $n-1$ đĩa từ B sang C.

Độ phức tạp: $O(2^n)$.

Code:

```
def tower_of_hanoi(n, source, target, auxiliary):
    if n == 1:
        print(f"Move disk 1 from {source} to {target}")
        return
    tower_of_hanoi(n - 1, source, auxiliary, target)
    print(f"Move disk {n} from {source} to {target}")
    tower_of_hanoi(n - 1, auxiliary, target, source)

n = 3
tower_of_hanoi(n, 'A', 'C', 'B')
```