BÀI TẬP TOÁN RỜI RẠC CHƯƠNG 3

HỌ TÊN: Nguyễn Hữu Nam

MSV: B23DCCC121

1. Cho tập A = 1, 2, 3, 4, 5, 6, 7, 8, 9. Sử dụng phương pháp sinh hoán vị theo thứ tự từ điển, tìm 4 hoán vị kiền kề tiếp theo của hoán vị 568397421.

Theo phương pháp sinh hoán vị:
- Duyệt từ phải qua trái tìm vị trí i có pi < pi+1
- Tìm j > i có pj > pi nhỏ nhất
- Đổi chỗ pi và pj
- Lật ngược đoạn từ i+1 đến n

Tìm 4 hoán vị kế tiếp:

[5, 6, 8, 4, 1, 2, 3, 7, 9]

[5, 6, 8, 4, 1, 2, 3, 9, 7]

[5, 6, 8, 4, 1, 2, 7, 3, 9]

[5, 6, 8, 4, 1, 2, 7, 9, 3]

2. Cho tập A = 1, 2, 3, 4, 5, 6, 7, 8, 9. Sử dụng phương pháp sinh hoán vị theo thứ tự từ điển, tìm 4 hoán vị kiền kề tiếp theo của hoán vị 458796321.

Theo phương pháp sinh hoán vị:
- Duyệt từ phải qua trái tìm vị trí i có pi < pi+1
- Tìm j > i có pj > pi nhỏ nhất
- Đổi chỗ pi và pj
- Lật ngược đoạn từ i+1 đến n

Tìm 4 hoán vị kế tiếp:

[4, 5, 8, 9, 1, 2, 3, 6, 7]

[4, 5, 8, 9, 1, 2, 3, 7, 6]

[4, 5, 8, 9, 1, 2, 6, 3, 7]

[4, 5, 8, 9, 1, 2, 6, 7, 3]

3. Cho tập A = 1, 2, 3, 4, 5, 6, 7, 8, 9. Sử dụng phương pháp sinh hoán vị theo thứ tự từ điển, tìm 4 hoán vị kiền kề tiếp theo của hoán vị 236897541.

Theo phương pháp sinh hoán vị:
- Duyệt từ phải qua trái tìm vị trí i có pi < pi+1
- Tìm j > i có pj > pi nhỏ nhất
- Đổi chỗ pi và pj
- Lật ngược đoạn từ i+1 đến n

Tìm 4 hoán vị kế tiếp:

[2, 3, 6, 9, 1, 4, 5, 7, 8]

[2, 3, 6, 9, 1, 4, 5, 8, 7]

[2, 3, 6, 9, 1, 4, 7, 5, 8]

[2, 3, 6, 9, 1, 4, 7, 8, 5]

4. Cho tập A = 1, 2, 3, 4, 5, 6, 7, 8, 9. Sử dụng phương pháp sinh tổ hợp chập k của một tập hợp theo thứ tự từ điển, hãy tạo 4 tổ hợp chập 4 liền kề tiếp theo của tổ hợp 2, 6, 8, 9.

Quy tắc sinh tổ hợp chập k kế tiếp:
1. **Duyệt từ phải qua trái** để tìm vị trí `i` đầu tiên sao cho `c[i] < n - k + i`. Điều này đảm bảo rằng phần tử tại vị trí `i` có thể tăng lên mà không vượt quá giới hạn cho phép, đồng thời vẫn có đủ không gian để các phần tử sau nó có thể tăng lên theo quy tắc tổ hợp.
2. **Tăng giá trị tại vị trí `i`** lên 1 đơn vị (`c[i] = c[i] + 1`).
3. **Đặt tất cả các phần tử sau `i`** theo quy tắc tổ hợp, tức là mỗi phần tử tiếp theo phải lớn hơn phần tử trước đó một đơn vị. Điều này được thực hiện bằng cách gán `c[j] = c[i] + j - i` cho mọi `j > i`.

[2, 7, 8, 9]

[3, 4, 5, 6]

[3, 4, 5, 7]

[3, 4, 5, 8]

5. Cho tập A = 1, 2, 3, 4, 5, 6, 7, 8, 9. Sử dụng phương pháp sinh tổ hợp chập k của một tập hợp theo thứ tự từ điển, hãy tạo 4 tổ hợp chập 4 liền kề tiếp theo của tổ hợp 3, 5, 6, 9.

Quy tắc sinh tổ hợp chập k kế tiếp:
1. **Duyệt từ phải qua trái** để tìm vị trí `i` đầu tiên sao cho `c[i] < n - k + i`. Điều này đảm bảo rằng phần tử tại vị trí `i` có thể tăng lên mà không vượt quá giới hạn cho phép, đồng thời vẫn có đủ không gian để các phần tử sau nó có thể tăng lên theo quy tắc tổ hợp.
2. **Tăng giá trị tại vị trí `i`** lên 1 đơn vị (`c[i] = c[i] + 1`).
3. **Đặt tất cả các phần tử sau `i`** theo quy tắc tổ hợp, tức là mỗi phần tử tiếp theo phải lớn hơn phần tử trước đó một đơn vị. Điều này được thực hiện bằng cách gán `c[j] = c[i] + j - i` cho mọi `j > i`.

[3, 5, 7, 8]

[3, 5, 7, 9]

[3, 5, 8, 9]

[3, 6, 7, 8]

6. Cho tập A = 1, 2, 3, 4, 5, 6, 7, 8, 9. Sử dụng phương pháp sinh tổ hợp chập k của một tập hợp theo thứ tự từ điển, hãy tạo 4 tổ hợp chập 4 trước liền kề của tổ hợp 3, 5, 6, 9.

Quy tắc sinh tổ hợp chập k trước liền kề:

- **Duyệt từ phải qua trái** để tìm vị trí `i` đầu tiên sao cho `c[i] > i + 1`. Điều này đảm bảo rằng phần tử tại vị trí `i` có thể giảm xuống mà không vi phạm quy tắc tổ hợp, đồng thời vẫn đảm bảo các phần tử sau nó có thể được cập nhật theo quy tắc tổ hợp.
- **Giảm giá trị tại vị trí `i`** xuống 1 đơn vị (`c[i] = c[i] - 1`).
- **Đặt tất cả các phần tử sau `i`** theo quy tắc tổ hợp, tức là mỗi phần tử tiếp theo phải lớn hơn phần tử trước đó một đơn vị. Điều này được thực hiện bằng cách gán `c[j] = c[i] + (j - i)` cho mọi `j > i`.

[3, 5, 6, 8]

[3, 5, 6, 7]

[3, 4, 5, 9]

[3, 4, 5, 8]

7. Cho dãy A = (a_1, a_2, ..., a_n). Viết chương trình trong C/C++/Python liệt kê các dãy con k phần tử giảm dần của dãy số A? Ví dụ: Cho A = (1, 5, 3, 4, 2, 0), k=3, các dãy con thỏa mãn yêu cầu đề bài: (5,3,2); (5,2,0);(5,4,2)... Lưu ý: Không sử dụng các thư viện có sẵn để sinh hoán vị, tổ hợp.

```python
def day_giam_dan(A, k):
    n = len(A)
    result = []
    current = []

    def backtrack(start, current):
        # Nếu đã đủ k phần tử và dãy giảm dần
        if len(current) == k:
            result.append(current[:])
            return

        # Thử các phần tử từ vị trí start
        for i in range(start, n):
            # Nếu current rỗng hoặc phần tử mới nhỏ hơn phần tử cuối của current
            if not current or A[i] < current[-1]:
                current.append(A[i])
                backtrack(i + 1, current)
                current.pop()

    # Bắt đầu với mảng rỗng
    backtrack(0, current)

    # Sắp xếp kết quả theo thứ tự giảm dần
    result.sort(reverse=True)
    return result


# Test với ví dụ trong đề
A = [1, 5, 3, 4, 2, 0]
k = 3

# Tìm các dãy con giảm dần
subsequences = day_giam_dan(A, k)

# In kết quả
for seq in subsequences:
    print(seq)
```

```
C:\Users\ASUS\AppData\Local\Programs\Python\Python39\python.exe
[5, 4, 2]
[5, 4, 0]
[5, 3, 2]
[5, 3, 0]
[5, 2, 0]
[4, 2, 0]
[3, 2, 0]

Process finished with exit code 0
```

8. Viết chương trình C/C++/Python liệt kê các hoán vị của tập {1, 2, ..., n} sử dụng phương pháp sinh theo thứ tự từ điển.

```python
def next_permutation(perm):
    n = len(perm)

    # Bước 1: Tìm từ phải sang trái vị trí i đầu tiên mà perm[i] < perm[i+1]
    i = n - 2
    while i >= 0 and perm[i] >= perm[i + 1]:
        i -= 1
    Ctrl+L to chat, Ctrl+K to generate
    # Nếu không tìm thấy, đây là hoán vị cuối cùng
    if i < 0:
        return False

    # Bước 2: Tìm từ phải sang trái vị trí j đầu tiên mà perm[j] > perm[i]
    j = n - 1
    while perm[j] <= perm[i]:
        j -= 1

    # Bước 3: Đổi chỗ perm[i] và perm[j]
    perm[i], perm[j] = perm[j], perm[i]

    # Bước 4: Lật ngược đoạn từ i+1 đến n-1
    left = i + 1
    right = n - 1
    while left < right:
        perm[left], perm[right] = perm[right], perm[left]
        left += 1
        right -= 1

    return True


def generate_permutations(n):
    # Tạo hoán vị đầu tiên [1, 2, ..., n]
    perm = list(range(1, n + 1))

    # In hoán vị đầu tiên
    print(perm)

    # Sinh và in các hoán vị tiếp theo
    while next_permutation(perm):
        print(perm)


# Test với n = 3
n = 3
print(f"Các hoán vị của tập {{1, 2, ..., {n}}}:")
generate_permutations(n)
```

```
C:\Users\ASUS\AppData\Local\Programs\Python\Python39\python.exe
Các hoán vị của tập {1, 2, ..., 3}:
[1, 2, 3]
[1, 3, 2]
[2, 1, 3]
[2, 3, 1]
[3, 1, 2]
[3, 2, 1]
```

9. Trình bày phương pháp liệt kê các tổ hợp chập k của tập {1, 2, ...n} sử dụng phương pháp quay lui.

```python
def print_combination(arr):
    print(arr)

def try_combination(i, n, k, arr):
    for j in range(arr[i-1] + 1, n - k + i + 1):
        arr[i] = j
        if i == k:
            print_combination(arr[1:k+1])
        else:
            try_combination(i + 1, n, k, arr)

def generate_combinations(n, k):
    # Khởi tạo mảng arr với k+1 phần tử, arr[0] = 0
    arr = [0] * (k + 1)
    try_combination(1, n, k, arr)

# Test với n = 5, k = 3
n = 5
k = 3
print(f"Các tổ hợp chập {k} của tập {{1, 2, ..., {n}}}:")
generate_combinations(n, k)
```

```
C:\Users\ASUS\AppData\Local\Programs\Python\Python39\python.exe
Các tổ hợp chập 3 của tập {1, 2, ..., 5}:
[1, 2, 3]
[1, 2, 4]
[1, 2, 5]
[1, 3, 4]
[1, 3, 5]
[1, 4, 5]
[2, 3, 4]
[2, 3, 5]
[2, 4, 5]
[3, 4, 5]

Process finished with exit code 0
```

10. Viết hàm trong C/C++/Python sử dụng phương pháp quay lui liệt kê các xâu nhị phân độ dài n. Sau đó trình bày cây biểu diễn quá trình hoạt động của hàm khi sinh các xâu nhị phân độ dài 3.

```python
def print_binary(arr, n):
    # In xâu nhị phân hiện tại
    for i in range(n):
        print(arr[i], end='')
    print()

def try_binary(i, n, arr):
    # Thử các giá trị 0, 1 cho vị trí i
    for j in range(2):  # j chạy từ 0 đến 1
        arr[i] = j
        if i == n - 1:  # Nếu đã đủ n phần tử
            print_binary(arr, n)
        else:
            try_binary(i + 1, n, arr)  # Quay lui cho vị trí tiếp theo

def generate_binary(n):
    arr = [0] * n  # Khởi tạo mảng n phần tử
    try_binary(0, n, arr)

# Test với n = 3
n = 3
print(f"Các xâu nhị phân độ dài {n}:")
generate_binary(n)
```

```
C:\Users\ASUS\AppData\Local\Programs\Python\Python39\python.exe
Các xâu nhị phân độ dài 3:
000
001
010
011
100
101
110
111
```

11. Viết hàm trong C/C++/Python sử dụng phương pháp sinh theo thứ tự từ điển liệt kê các tổ hợp chập k của tập n phần tử {1, 2, ...n}.

```python
def next_combination(a, n, k):
    i = k
    # Tìm từ phải sang trái phần tử đầu tiên chưa đạt giới hạn trên
    while i > 0 and a[i] >= n - k + i:
        i -= 1

    if i > 0:  # Nếu tìm thấy
        a[i] += 1  # Tăng phần tử tại vị trí i
        # Điều chỉnh các phần tử phía sau
        for j in range(i + 1, k + 1):
            a[j] = a[j - 1] + 1
        return True
    return False  # Không còn cấu hình tiếp theo


def generate_combinations(n, k):
    # Khởi tạo cấu hình đầu tiên: 1,2,3,...,k
    a = [0] * (k + 1)  # a[0] không sử dụng
    for i in range(1, k + 1):
        a[i] = i

    # In cấu hình đầu tiên
    print(a[1:])

    # Sinh và in các cấu hình tiếp theo
    while next_combination(a, n, k):
        print(a[1:])


# Test với n = 5, k = 3
n = 5
k = 3
print(f"Các tổ hợp chập {k} của tập {{1, 2, ..., {n}}}:")
generate_combinations(n, k)
```

```
C:\Users\ASUS\AppData\Local\Programs\Python\Python39\python.exe
Các tổ hợp chập 3 của tập {1, 2, ..., 5}:
[1, 2, 3]
[1, 2, 4]
[1, 2, 5]
[1, 3, 4]
[1, 3, 5]
[1, 4, 5]
[2, 3, 4]
[2, 3, 5]
[2, 4, 5]
[3, 4, 5]
```