1.3: Liệt kê dãy nhị phân có độ dài n

Code trình bày tại file hw4_1_3.py

1.1: Tìm đường đi trong mê cung:

Code trình bày tại file hw4_1_1.py

2.1: Liệt kê các hoán vị của phần tử: *Code được trình bày trong hw4_2_1*

- Phân tích bài toán:

  Input là mảng a với n phần tử. Ta dùng mảng visited để đánh dấu với phần tử đã được sử dụng (visited[v] = False nếu phần tử I chưa được sử dụng)

  Sử dụng hàm hoanvi(i) để sinh ra các b[i] với i là các giá trị chỉ số thuộc mảng input a. v được chấp nhận nếu chưa đi qua.

  Khi giá trị i = n -1, kết thúc hàm và in ra giá trị nghiệm. Ngược lại thì tiếp tục sinh b[i] bằng hàm hoanvi(i+1)

- Lược đồ:

```
hoanvi(i)
        for v in range(0,n): //v thuộc tập khả năng thành phần nghiệm
                if visited[v] == False: //Nếu i đã được sử dụng
                        b[i] = v
                        if i==n-1:
                                for j in b: print(a[j])
                        else hoanvi(i+1)
                        visited[v] = False
                endif
        endfor
end.
```

2.2 Liệt kê các tổ hợp của k phần tử của các số từ 1 đến n: *Code được trình bày trong hw4_2_1*

- Phân tích bài toán:
  Lời gọi ban đầu tohop(0) với:

- Sử dụng hàm tohop(i) để sinh ra các w[i] với w[i] là các giá trị thuộc range(1, n)
- Ta dùng mảng visited để đánh dấu với phần tử đã được sử dụng (visited[v] = False nếu phần tử I chưa được sử dụng)
- Nếu i = k thì mảng c với k phần tử là 1 tổ hợp k phần tử các số từ 1 đến n
  - Sắp xếp mảng c thành một mảng được sắp xếp có thứ tự tăng dần
  - Kiểm tra mảng c có trong a hay chưa nếu có thì thêm c vào mảng a
  - Ngược lại sinh tiếp b[i] bằng tohop(i+1)

- Lược đồ:

```
Tohop(i)
        for v in range(0,n): //v thuộc tập khả năng thành phần nghiệm
                if visited[v] == False: //Nếu i đã được sử dụng
                        b[i] = v +1
                        if i == k
                                c = []
                                for j in range(0, k):
                                        c += [b[j] //Thêm vào phần tử chạy từ 0
        đến k
                                        c = sorted(c) //sắp xếp cho mảng thành
                                        dãy tăng dần
                                        if (c not in a):
                                                a += [c]
                        else tohop(i+1)
                        visited[v] = False
                endif
        endfor
end.
```

3.

3.1: **Bài toán Backtracking: *Liệt kê số chỉnh hợp chập k của n phần tử (chỉnh hợp không lặp)***

- Phân tích bài toán:

Sử dụng hàm chinhhop(i) duyệt mọi khả năng chọn. Để biết giá trị nào chưa được chọn vào làm chỉnh hợp chập k của n phần tử ta sử dụng mảng visited để đánh dấu. Mảng visited có các phần tử mảng được khởi tạo là False (từ 1 đến n).

Khi chinhhop(i) có i = k thì in ra kết quả. Ngược lại thì tiếp tục sinh b[i] bằng hàm chinhhop(i+1)

- <u>Xây dựng giải thuật:</u>

```
chinhhop(i):
        for v in range(0,n):
                if visited[v] = False
                        b[i] = v + 1
                        visited[v] = True
                        if i = k: c = []
                                for j in range(0, k):
                                        c += [b[j] //Thêm vào phần tử chạy từ 0
        đến k
                                        if (c not in a):
                                                a += [c]
                        else chinhhop(i+1)
                        visited[v] = False
                endif
        endfor
end.
```

(Code ở file hw4_3_1.py)

3.2: **Bài toán nhánh cận: *Tìm ký tự xuất hiện ít nhất trong xâu***