Mục lục

I. Phương pháp sinh

Nguyên tắc chung:

- Định nghĩa về cấu hình (xâu nhị phân, hoán vị, tổ hợp).
- Cấu hình đầu – cấu hình cuối.
- Quan hệ của cấu hình hiện tại với cấu hình kế tiếp.
- Phương pháp:
  - Giả sử cấu hình hiện tại.
  - Nếu là cấu hình cuối cùng, kết thúc thuật toán.
  - Sinh cấu hình kế tiếp.

1. Sinh nhị phân

- Xâu X = (x_1,..,x_n), x_i = {0,1} gọi là xâu nhị phân độ dài n.
- Cấu hình đầu tiên: (0,0,…,0) – cấu hình cuối cùng: (1,1,..,1)
- Nếu cấu hình hiện tại X chưa phải là cấu hình cuối cùng thì cấu hình kế tiếp thu được bằng cách cộng thêm 1 (trong hệ cơ số 2 có nhớ) vào cấu hình hiện tại.

❖ Phương pháp:

- Giả sử cấu hình hiện tại là X = (x_1,…,x_n).

- Nếu X là cấu hình cuối cùng, thuật toán kết thúc.
- Gọi x_k là chữ số 0 đầu tiên từ bên phải của X, tức là $X = x_1x_2\ldots x_{k-1}01\ldots1$
- Cấu hình kế tiếp $Y = (y_1,\ldots,y_n)$ được tạo ra như sau:

$y_i = x_i$, với $1 \le i \le k-1$

$y_i = 1 - x_i$ với $k \le i \le n$

Vậy $Y = x_1x_2\ldots x_{k-1}10\ldots0$

❖ Viết hàm:

```
void sinh()
{
    for(int i = 1; i <= n; i++) x[i] = 0;
    while(1)
    {
        for(int i = 1; i <= n; i++) cout << x[i];
        cout << endl;
        int i = n;
        while(x[i] == 1 && i > 0)
        {
            x[i] = 0;
            i--;
        }
        if(i == 0) break;
        else x[i] = 1;
    }
}
```

2. Sinh tổ hợp

- Một xâu biểu diễn tổ hợp chập k của n có dạng $X = (x_1,x_2,\ldots,x_k)$.
- Cấu hình đầu tiên: $(1,2,\ldots,k)$ – cấu hình cuối cùng: $(n-k+1, n-k+2,\ldots, n)$.
- Cấu hình $X = (x_1,\ldots,x_k)$ được gọi là đứng trước cấu hình $Y = (y_1,\ldots,y_k)$ nếu tồn tại chỉ số t thỏa mãn: $x_1 = y_1$, $x_2 = y_2$, …, $x_{t-1} = y_{t-1}$, $x_t < y_t$.

❖ Phương pháp:

- Giả sử cấu hình hiện tại là $X = (x_1,x_2,\ldots,x_k)$.
- Nếu X là cấu hình cuối cùng, thuật toán kết thúc.
- Tìm x_i là phần tử đầu tiên tính từ bên phải của X mà $x_i \neq n - k + i$.
- Thay x_i bởi $x_i + 1$.
- Thay x_j bằng $x_i + j - i$, với $j = i + 1, i + 2,\ldots, k$.

❖ Viết hàm:

```
void sinh()
{
    for(int i = 1; i <= k; i++) x[i] = i;
    while(1)
    {
        for(int i = 1; i <= k; i++) cout << x[i];
        cout << endl;
        int i = k;
        while(x[i] == n - k + i && i > 0) i--;
        if(i == 0) break;
        else
        {
            x[i]++;
            for(int j=i+1; j<=k; j++) x[j] = x[j-1] + 1;
        }
    }
}
```

3. Sinh hoán vị

- Một xâu biểu diễn hoán vị của n có dạng $X = (x_1,x_2,\ldots,x_n)$.
- Cấu hình đầu tiên: (1,2,…,n) – cấu hình cuối cùng: (n, n-1,…, 1).
- Cấu hình $X = (x_1,\ldots,x_n)$ được gọi là đứng trước cấu hình $Y = (y_1,\ldots,y_n)$ nếu tồn tại chỉ số t thỏa mãn: $x_1 = y_1$, $x_2 = y_2$, …, $x_{t-1} = y_{t-1}$, $x_t < y_t$.

❖ Phương pháp:

- Giả sử cấu hình hiện tại là $X = (x_1,x_2,\ldots,x_n)$.
- Nếu X là cấu hình cuối cùng, thuật toán kết thúc.
- Tìm x_i là phần tử đầu tiên tính từ bên phải của X mà $x_i < x_{i+1}$.
- Tìm x_j nhỏ nhất mà còn lớn hơn x_i trong các số bên phải x_i.
- Đổi chỗ (x_i, x_j).
- Lật ngược đoạn từ x_{i+1} đến x_n.

❖ Viết hàm:

```
void sinh()
{
    for(int i = 1; i <= n; i++) x[i] = i;
    while(1)
    {
        for(int i = 1; i <= n; i++) cout << x[i];
        cout << endl;
        int i = n - 1;
```

```
        while(x[i] > x[i+1] && i > 0) i--;
        if(i == 0) break;
        else
        {
            int j = n;
            while(x[j] < x[i]) j--;
            swap(x[i], x[j]);
            int l = i + 1, r = n;
            while(l <= r)
            {
                swap(x[l], x[r]);
                l++; r--;
            }
        }
    }
}
```

II. Quay lui

Nguyên tắc chung:

- Định nghĩa về các cấu hình cần liệt kê.
- Các khả năng ứng với phần tử x_i, điều kiện của những khả năng này.
- Làm gì với các khả năng.
- Nếu là phần tử cuối cùng, ghi nhận nghiệm bài toán. Nếu chưa, tiếp tục xác định thành phần thứ i + 1.

1. Quay lui nhị phân

- Giả sử cần xác định xâu $X = (x_1,\ldots,x_n)$ biểu diễn các xâu nhị phân độ dài n.

- Mỗi phần tử x_i có 2 khả năng lựa chọn là 0 hoặc 1. Các giá trị này mặc nhiên được chấp thuận mà không cần thỏa mãn điều kiện gì.

- Ứng với mỗi khả năng j dành cho x_i ta thực hiện: chấp thuận từng khả năng j.

- Nếu i là thành phần cuối cùng (i = n), ta ghi nhận nghiệm bài toán. Nếu chưa, ta xác định tiếp thành phần thứ i + 1.

❖ Viết hàm:

```
void Try(int i)
{
    for(int j = 0; j <= 1; j++)
    {
        x[i] = j;
        if(i == n) Result();
        else Try(i + 1);
    }
```

```
}
```

2. Quay lui tổ hợp

- Giả sử cần xác định xâu X = (x_1,…,x_k) biểu diễn các tổ hợp chập k của n.

- Mỗi phần tử x_i có các khả năng lựa chọn từ x_{i-1} + 1 đến n – k + i. Các giá trị này mặc nhiên được chấp thuận mà không cần thỏa mãn điều kiện gì.

- Ứng với mỗi khả năng j dành cho x_i ta thực hiện: chấp thuận từng khả năng j.

- Nếu i là thành phần cuối cùng (i = k), ta ghi nhận nghiệm bài toán. Nếu chưa, ta xác định tiếp thành phần thứ i + 1.

❖ Viết hàm:

```
void Try(int i)
{
    for(int j=x[i-1]+1; j<=n-k+i; j++)
    {
        x[i] = j;
        if(i == k) Result();
        else Try(i + 1);
    }
}
```

3. Quay lui hoán vị

- Giả sử cần xác định xâu X = (x_1,…,x_n) biểu diễn hoán vị của n phần tử.

- Mỗi phần tử x_i có n khả năng lựa chọn. Khi x_i = j được lựa chọn thì giá trị này sẽ không được chấp thuận cho các thành phần còn lại.

- Ứng với mỗi khả năng j dành cho x_i ta thực hiện: ghi nhận x_i = j nếu j được chấp thuận. Để thực hiện điều này, ta sử dụng mảng unused[]. Nếu unused[i] = True thì ta có thể sử dụng i. Ngược lại, unused[i] = False thì ta không thể chọn giá trị này.

- Nếu i là thành phần cuối cùng (i = n), ta ghi nhận nghiệm bài toán. Nếu chưa, ta xác định tiếp thành phần thứ i + 1. Nếu không còn khả năng nào dành cho x_i ta quay lui lại bước trước đó để thử các khả năng tiếp theo của x_{i-1}.

❖ Viết hàm:

```
void Try(int i)
{
    for(int j = 1; j <= n; j++)
        if(unused[j])
        {
            x[i] = j;
            unused[j] = 0;
            if(i == n) Result();
            else Try(i + 1);
            unused[j] = 1;
        }
}
```

III. Duyệt toàn bộ

1. Bài toán tối ưu

   Tập phương án: D = {X = (x_1, x_2,…, x_n)}

   Xét mọi X∈D, tìm X* để f(X*)→max (min)

❖ Thuật toán:

- B1: Khởi tạo

  XOPT=∅;FOPT= - ∞ (+ ∞);

- B2: Lặp

```
for(X∈D)
{
    S = f(X)
    if(FOPT < S)
    {
        FOPT = S;
        XOPT = X;
    }
}
```

- B3: Return(FOPT, XOPT)

2. Bài toán cái túi

   Đề bài: N đồ vật

   b: trọng lượng tối đa túi chứa được

   A = (a_1, a_2,…, a_n): trọng lượng từng đồ vật

   C = (c_1, c_2,…, c_n): giá trị sử dụng từng đồ vật

   Tập phương án: D = {X = (x_1, x_2,…, x_n), x_i ∈{0,1}, g(X) = $\sum_{i=1}^{n} a_i * x_i \le b$}

   Xét mọi X ∈ D, tìm X* để f(X*) →max: f(X) = $\sum_{i=1}^{n} c_i * x_i$

❖ Thuật toán:

- B1: Khởi tạo

XOPT=∅;FOPT= - ∞;

- B2: Lặp

```
for(X∈D)
{
    W = g(X)
    S = f(X)
    if(W <= b && FOPT < S)
    {
        FOPT = S;
        XOPT = X;
    }
}
```

- B3: Return(FOPT, XOPT)

IV. Nhánh cận

1. Bài toán nhánh cận

Tập phương án: $D = \{ X = (x_1, x_2, \ldots, x_n) \in A_1 \times A_2 \times \ldots \times A_n \}$

Tìm $\min\{f(X) : X \in D\}$

❖ Ý tưởng:

- Điều kiện của thành phần thứ k.
- Cận của phương án.
- Quay lui thử lại kết quả.

❖ Thuật toán:

```
Branch_And_Bound(k)
{
    for(ak∈Ak)
        if(<chấp nhận ak>)
        {
            Xk = ak;
            if(k == n) <Cập nhật kỉ lục>;
            else if(g(x1,x2,…,xk) < FOPT) Branch_And_Bound(k+1);
        }
}
```

2. Bài toán cái túi

❖ Ý tưởng:

B1: Khởi tạo

- Sắp xếp các đồ vật trong túi thỏa mãn: $\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \cdots \geq \frac{c_n}{a_n}$

FOPT = -∞; XOPT = ∅; b_0 = b; δ_0 = 0;

B2: Lặp k = 1, 2, …, n

- Trọng lượng còn lại: $b_k = b - \sum_{i=1}^{k} a_i * x_i$
- Giá trị đồ vật trong túi: $\delta_k = \sum_{i=1}^{k} c_i * x_i$
- Cận trên của phương án: $g_k = \delta_k + b_k * \frac{c_{k+1}}{a_{k+1}}$
- Nếu $b_k < 0$ thì dừng và quay lui.

B3: Return(FOPT, XOPT)

❖ Thuật toán:

```
Branch_And_Bound(k)
{
    for(j = 1; j >= 0; j--)
    {
        Xk = j;
        bk = bk – ak*xk;
        δk = δk + ck*xk;
        if(k == n) <Cập nhật kỉ lục>;
        else if(FOPT < δk + bk*ck+1/ak+1) Branch_And_Bound(k+1);
        bk = bk + ak*xk;
        δk = δk - ck*xk
    }
}
```

3. Bài toán người du lịch

Tập phương án: D = {X = (x_1,x_2,…,x_n)∧ $x_1 = 1 \wedge x_i \neq x_j$}

Tìm X* để f(X*)→ min: f(X) = $\sum_{i=1}^{n-1} C_{x_i x_{i+1}} + C_{x_n x_1}$

❖ Ý tưởng:

- Gọi cmin là chi phí nhỏ nhất của ma trận chi phí.
- Giả sử đang có hành trình qua k thành phố:

$$x_1 \rightarrow x_2 \rightarrow \cdots \rightarrow x_k$$

- Chi phí: $\delta = \sum_{i=1}^{k-1} c_{x_i x_{i+1}}$
- Hành trình còn lại: cần phải đi qua n – k thành phố, rồi quay trở lại thành phố 1. Như vậy, cần đi qua n – k + 1 đoạn đường còn lại. Vì mỗi đoạn đường có chi phí không nhỏ hơn cmin nên cận dưới của phương án:

g_k = δ + (n – k +1)*cmin

❖ Thuật toán:

```
Branch_And_Bound (k) {
    for (j = 2; j <= n; j++)
    {
        if (chuaxet[j])
        {
            x[k] = j;
            chuaxet[j] = 0;
            δ = δ + c[x[k-1], x[k]];
            if(k==n) Ghi_Nhan();
            else if (δ + (n-k+1)*cmin < FOPT)
                Branch_And_Bound (k+1);
            chuaxet[j] = 1;
            δ = δ - c[x[k-1], x[k]];
        }
    }
}
```