**BÀI 2: THUẬT TOÁN SINH – SINH KẾ TIẾP**

Câu hỏi kì này:

- Tìm hiểu về sinh nhị phân.

- Tìm hiểu về sinh tổ hợp.

- Tìm hiểu về sinh hoán vị.

**Mục lục**

[I. TÌM HIỂU CHUNG 2](#_Toc104230770)

[1. Khái quát: 2](#_Toc104230771)

[2. Điều kiện: 2](#_Toc104230772)

[3. Ý tưởng, cấu trúc: 2](#_Toc104230773)

[II. SINH NHỊ PHÂN. 3](#_Toc104230774)

[1. Khái quát: 3](#_Toc104230775)

[2. Ý tưởng: 3](#_Toc104230776)

[3. Tóm gọn: 3](#_Toc104230777)

[4. Code mẫu: 4](#_Toc104230778)

[III. SINH TỔ HỢP. 5](#_Toc104230779)

[1. Khái quát: 5](#_Toc104230780)

[2. Ý tưởng: 5](#_Toc104230781)

[3. Tóm gọn: 5](#_Toc104230782)

[4. Code mẫu: 6](#_Toc104230783)

[IV. SINH HOÁN VỊ. 7](#_Toc104230784)

[1. Khái quát: 7](#_Toc104230785)

[2. Ý tưởng: 7](#_Toc104230786)

[3. Tóm gọn: 7](#_Toc104230787)

[4. Code mẫu: 8](#_Toc104230788)

# I. TÌM HIỂU CHUNG

## 1. Khái quát:

- Khi cần liệt kê các cấu hình khác nhau phù hợp với điều kiện mà đề bài đưa ra, đối với nhưng bài toán lớn, ta khó có thể liệt kê chay trong thời gian ngắn. Chính vì thế, thuật toán sinh được ra đời để giải quyết các vấn đề này hiệu quả hơn.

## 2. Điều kiện:

Khi chạy một chương trình, muốn áp dụng được thuật toán sinh thì cần phải thỏa mãn 2 điều kiện sau:

- Xác định được cấu hình đầu, cuối và trình tự biến đổi của tập kết quả. Cấu hình đầu tiên chính là cấu hình bắt đầu hàm sinh, tập kết quả sẽ biến đổi lần lượt theo trình tự và dừng lại khi gặp cấu hình cuối cùng.

- Trong quá trình biến đổi theo chương trình, khi một cấu hình chưa phải cấu hình cuối, nó sẽ biến đổi thành cấu hình kế tiếp nó.

## 3. Ý tưởng, cấu trúc:

- Ý tưởng: Ta đưa ra cấu hình đầu tiên, kiểm tra xem nó đã phải cấu hình cuối cùng hay chưa. Nếu chưa thì ta tiếp tục biến đổi cấu hình hiện tại, còn nếu đã là cấu hình cuối cùng, thì ta dừng chương trình.

- Cấu trúc:

<Xây dựng cấu hình đầu tiên>;

While(chưa phải cấu hình cuối)

{

<Đưa ra cấu hình đang có>;

<Từ cấu hình đang có sinh ra cấu hình kế tiếp nếu còn>;

}

# II. SINH NHỊ PHÂN.

## 1. Khái quát:

- Là phương pháp sinh sinh ra một hoặc một chuỗi số nhị phân liên tiếp nhau.

- Cấu hình đầu là 0000…0 và cấu hình cuối là 1111…1 (dãy gồm n chữ số).

*Vd: 000, 001, 010, 011, 100, 101, 110, 111.*

## 2. Ý tưởng:

A screenshot of a computer

Description automatically generated with low confidence- Theo đúng tư tưởng của thuật toán sinh, ta đi từ cấu hình ban đầu là 0000…0, các cấu hình tiếp theo sẽ bằng cấu hình trước cộng thêm 1 (theo hệ đếm cơ số 2), cho đến khi nào gặp cấu hình cuối cùng. Số chuỗi nhị phân được sinh ra sẽ là 2n.

- Phương pháp: xét từ cuối của dãy lên đầu, khi ta gặp số 0 thì biến đổi thành số 1, và tất cả các chữ số đằng sau nó trở về 0. Lặp lại như thế cho đến khi không tìm thấy số 0 nào nữa là ta gặp được cấu hình cuối cùng.

*Vd: 001011*

*Ta xét từ cuối lên, tìm đến vị trí số 0 đầu tiên. Sau đó, biến đổi thành nó thành số 1. Các số ở đằng sau biến đổi về thành 0.*

*Ta được số 001100 (1210) là cấu hình kế tiếp 001011 (1110).*

*Sau đó, ta lại tiếp tục biến đổi cấu hình, đến khi nào đến được cấu hình cuối cùng thì dừng.*

- Độ phức tạp: O(n \* 2n) = O(2n).

## 3. Tóm gọn:

• Bước 1 : Khởi tạo cấu hình ban đầu : 000...0 (n số 0)

• Bước 2 : Xét từ cuối dãy về đầu, gặp số 0 đầu tiên thì thay nó bằng số 1 và đặt

tất cả các phần tử phía sau vị trí đó bằng 0.

• Bước 3 : In cấu hình và quay lại bước 2.Sinh cho đến khi gặp cấu hình cuối :

111...1 (n số 1) thì kết thúc quá trình sinh.

## 4. Code mẫu:

#include <bits/stdc++.h>

using namespace std;

int check = 0, n, a[1000] = {};

void sinh()

{

    int i = n - 1;

    while(i >= 0 && a[i] == 1)

    {

        a[i] = 0;

        i--;

    }

    if(i == -1) check = 1;

    else a[i] = 1;

}

void in()

{

    for(int i = 0; i < n; i++)

    {

        cout << a[i] << " ";

    }

    cout << endl;

}

void Progress()

{

    cin >> n;

    while(check == 0) in(), sinh();

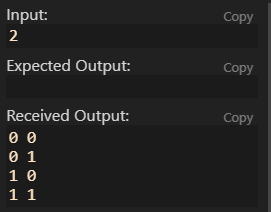
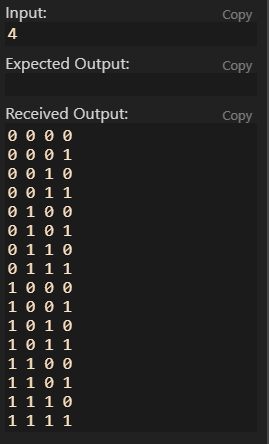
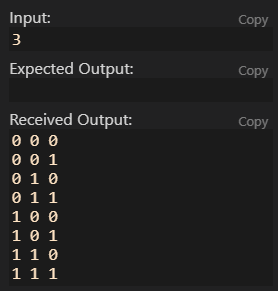
}

int main()

{

    Progress();

}



# III. SINH TỔ HỢP.

## 1. Khái quát:

- Tổ hợp: là các tập con gồm k phần tử của n phần tử ban đầu, không phân biệt về thứ tự sắp xếp các phần tử.

- Sinh tổ hợp: Là phương pháp tạo ra các tổ hợp con gồm k phần tử, rồi liệt kê chúng ra theo lần lượt. Việc này tương đương liệt kê các tập con của 1 tập hợp.

*Vd: Tập gồm {1, 2, 3, 4} thì có các tập con {{1, 2, 3}; {1, 2, 4}; {1, 3, 4}; {2, 3, 4}};*

## 2. Ý tưởng:

- Nhìn vào ví dụ mẫu, ta nhận ra được, cấu hình đầu tiên là {1, 2, 3, …, n} và cấu hình cuối là {n – k + 1, n – k + 2, …, n}. Hai cấu hình kế tiếp nhau là hai cấu hình thỏa mãn điều kiện: cấu hình sau đủ lớn hơn cấu hình trước, và giữa 2 cấu hình này không tồn tại một cấu hình nào khác thỏa mãn.

- Phương pháp: Ta sẽ in lần lượt các phần tử của tập con ra theo thứ tự tăng dần. Xét từ phần tử cuối xét lên. Nếu phần tử đó chưa tới giới hạn, thì ta cộng thêm vào nó 1 đơn vị, các phần từ đứng sau lơn hơn phần tử trước 1 đơn vị, ta được cấu hình kế tiếp của cấu hình đang xét. Ta sẽ tiếp tục làm như thế, đến khi nào tất cả các phần tử đều tới hạn, thì ta được cấu hình cuối cùng, chương trình kết thúc.

*Vd: Cho n = 5, k = 4, cấu hình hiện tại là 1245.*

*Ta xét từ cuối lên, tìm thấy a[2] = 2 là số chưa tới hạn đầu tiên được tìm thấy a[2] = 2 chưa tới hạn. Ta tăng a[2] lên 1 đơn vị, ta được cấu hình kế tiếp là 1345.*

- Độ phức tạp: O(k \* C(k, n)).

## 3. Tóm gọn:

• Bước 1 : Khởi tạo cấu hình ban đầu.

• Bước 2 : Xét dãy từ cuối lên đầu, tìm phần tử thứ i đầu tiên không thỏa mãn a[i] = n - k + i. Nếu không tìm được, kết thúc quá trình sinh.

• Bước 3 : Tăng phần tử i lên 1 đơn vị, và từ phần tử i, toàn bộ phần tử sau nó có dạng a[i+1] = a[i] + 1 cho đến phần tử cuối. In cấu hình và quay lại bước 2.

## 4. Code mẫu:

#include <bits/stdc++.h>

using namespace std;

int check = 0, n, a[1000], k;

// SINH TO HOP

void sinhTH()

{

    int i = k - 1;

    while(a[i] == n - k + i + 1) i--;

    if(i < 0) check = 1;

    else a[i]++;

    int p = a[i];

    while(i < k) a[i++] = p++;

}

void in()

{

    for(int i = 0; i < k; i++)

    {

        cout << a[i] << " ";

    }

    cout << endl;

}

void Progress()

{

    cin >> n >> k;

    for(int i = 0; i < k; i++)

    {

        a[i] = i + 1;

    }

    while(check == 0) in(), sinhTH();

}

int main()

{

    Progress();

}

Graphical user interface, application

Description automatically generated

# IV. SINH HOÁN VỊ.

## 1. Khái quát:

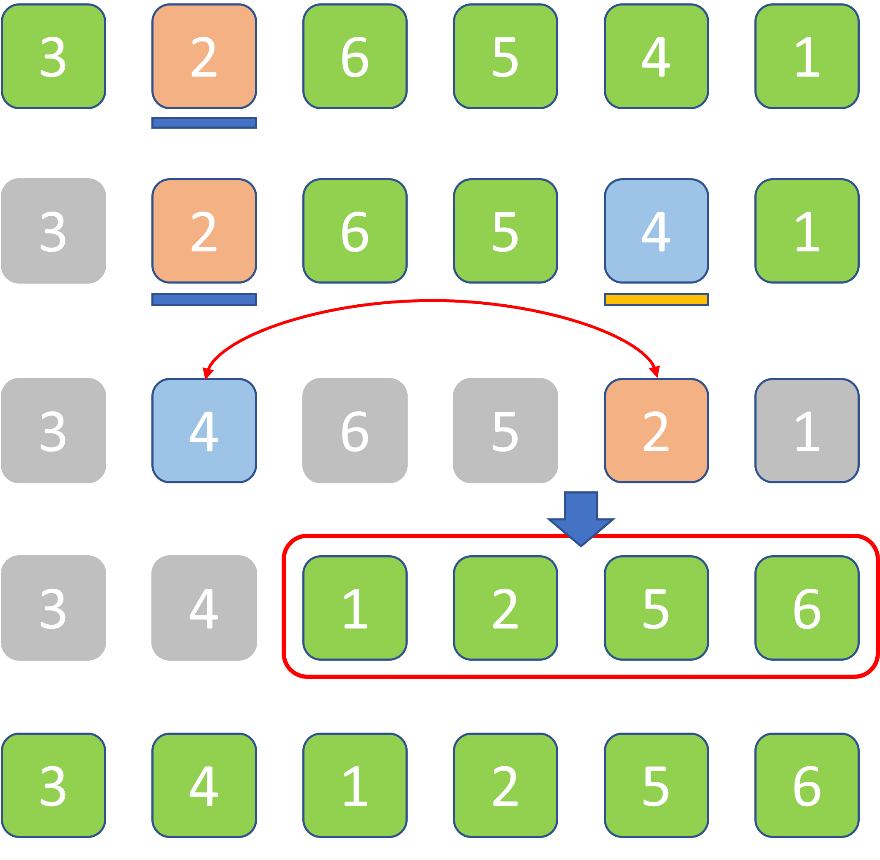
- Một hoán vị, là một dãy chứa các phần tử của 1 tập hợp theo một thứ tự xác định, chỉ chứa một là đúng 1 lần hoán vị đó.

- Hoán vị cũng giống như sinh tổ hợp, nhưng các phần tử có thứ tự sắp xếp xác định.

*Vd: các hoán vị của dãy 1, 2, 3 là: {1, 2, 3}; {1, 3, 2}; {2, 1, 3}; {2, 3, 1}; {3, 1, 2}; {3, 2, 1}*

## 2. Ý tưởng:

- Từ ví dụ bên trên, ta thấy được một tổ hợp các hoán vị sẽ có cấu hình bắt đầu là 1, 2, 3, …, n. Và có cấu hình kết thúc là n, n – 1, n – 2, …, 1. Hai hoán vị kế tiếp nhau là hai hoán vị thỏa mãn điều kiện hoán vị sau đủ lớn hơn hoán vị trước, và không có hoán vị nào khác nằm giữa 2 hoán vị này.

- Giả sử hoán vị hiện tại là x = {3, 2, 6, 5, 4, 1}, xét 4 phần tử cuối cùng, ta thấy chúng được sắp xếp giảm dần, điều đó có nghĩa là cho dù ta có hoán vị 4 phần tử này thế nào ta cũng chỉ thu được hoán vị nhỏ hơn hoán vị hiện tại. Như vậy ta phải xét đến x[2] = 2, thay nó bằng giá trị khác. Vậy ta cần chọn giá trị nào ? Câu trả lời chính là số 4. Ta sẽ được hoán vị mới là {3, 4, 1, 2, 5, 6} là hoán vị kế tiếp hoán vị hiện tại.

- Phương pháp: Xét từ cuối dãy xét lên, nếu gặp 1 số phá vỡ chuỗi tăng dần từ cuối lên, đó sẽ là số cần thay đổi, gọi là a[i]. Tiếp tục xét từ cuối xét lên, nếu gặp 1 số nhỏ nhất lớn hơn a[i], ta gọi là a[j]. Ta đổi chỗ a[i], a[j] và rồi sắp xếp tăng dần từ a[i + 1] đến hết. Ta sẽ thu được hoán vị kế tiếp.

- Độ phức tạp: O(n!)

## 3. Tóm gọn:

• Bước 1 : Khởi tạo cấu hình ban đầu : 1, 2, 3, ..., n.

• Bước 2 : Xét từ cuối dãy lên đầu dãy,tìm phần tử đầu tiên làm mất tính không giảm của dãy, đánh dẫu phần tử đó. Nếu không tìm được,kết thúc quá trình sinh.

• Bước 3 : Xét từ cuối lên đầu dãy,tìm phần tử đầu tiên lớn hơn phần tử được đánh dấu, đổi chỗ 2 phần tử đó.

• Bước 4 : Từ vị trí phần tử tìm được ở bước 2,sắp xếp tăng dần từ vị trí đó cho đến cuối dãy.In cấu hình và quay lại bước 2.

## 4. Code mẫu:

#include <bits/stdc++.h>

using namespace std;

int check = 0, n, a[1000];

// SINH HOAN VI

void sinhHV()

{

    int i = n - 1;

    while (i > 0 && a[i] > a[i + 1])

        i--;

    if (i == 0)

        check = 1;

    else

    {

        int k = n;

        while (a[i] > a[k])

            k--;

        swap(a[i], a[k]);

        int c = n, r = i + 1;

        while (r < c)

        {

            swap(a[c], a[r]);

            r++;

            c--;

        }

    }

}

void in()

{

    for (int i = 1; i <= n; i++)

    {

        cout << a[i] << " ";

    }

    cout << endl;

}

void Progress()

{

    cin >> n;

    for (int i = 1; i <= n; i++)

    {

        a[i] = i;

    }

    while (check == 0)

        in(), sinhHV();

}

int main()

{

    Progress();

}

