Thuật toán sinh tổ hợp chập K của N giúp liệt kê tất cả các tập hợp con gồm K phần tử được chọn từ tập {1, 2, ..., N} mà không quan tâm đến thứ tự.

**1. Khai báo biến**

* n, k: Số phần tử của tập hợp (n) và số phần tử trong mỗi tổ hợp (k).
* X[100]: Mảng lưu tổ hợp hiện tại.
* ok: Biến kiểm tra xem đã sinh xong tổ hợp cuối cùng chưa.

**2. Khởi tạo cấu hình đầu tiên**

Cấu hình đầu tiên được tạo bằng cách gán các giá trị từ 1 đến k vào mảng X.  
Ví dụ với n = 5, k = 3, ta có cấu hình ban đầu:

1 2 3

Hàm ktao() thực hiện như sau:

void ktao(){

for(int i = 1; i <= k; i++){

X[i] = i;

}

}

**3. Sinh cấu hình tiếp theo**

Mục tiêu là tìm cấu hình tiếp theo của tổ hợp hiện tại.

* Duyệt từ phần tử cuối cùng (X[k]) trở về đầu (X[1]).
* Nếu X[i] đạt giá trị lớn nhất có thể (n - k + i), cần dịch sang trái để tiếp tục sinh tổ hợp mới.
* Nếu X[i] chưa đạt giá trị lớn nhất, tăng nó lên 1 và cập nhật các phần tử phía sau nó theo quy luật tăng dần.

**Ví dụ** với n = 5, k = 3:

* 1 2 3 → 1 2 4
* 1 2 4 → 1 2 5
* 1 2 5 → 1 3 4
* 1 3 4 → 1 3 5
* 1 3 5 → 1 4 5
* 1 4 5 → 2 3 4
* 2 3 4 → 2 3 5
* 2 3 5 → 2 4 5
* 2 4 5 → 3 4 5 (Cấu hình cuối cùng)

**Cách kiểm tra và cập nhật giá trị:**

* Bắt đầu từ i = k, kiểm tra xem X[i] đã đạt max chưa (X[i] == n - k + i).
* Nếu X[i] đạt max, lùi về X[i-1] để kiểm tra tiếp.
* Nếu tìm thấy một X[i] chưa đạt max, tăng nó lên 1, rồi cập nhật các phần tử sau nó theo quy tắc X[j] = X[j-1] + 1.

Hàm sinh() thực hiện điều này:

void sinh(){

int i = k;

while(i >= 1 && X[i] == n - k + i){

--i;

}

if(i == 0){

ok = false; // Đã sinh đến cấu hình cuối cùng

}

else{

X[i]++; // Tăng giá trị phần tử tìm được

for(int j = i + 1; j <= k; j++){

X[j] = X[j - 1] + 1; // Cập nhật các phần tử sau nó

}

}

}

**4. Vòng lặp sinh tổ hợp**

Sau khi khởi tạo cấu hình đầu tiên, chương trình liên tục sinh ra tổ hợp mới cho đến khi đạt cấu hình cuối cùng.

Hàm main() thực hiện:

int main(){

cin >> n >> k;

ktao();

while (ok) {

for(int i = 1; i <= k; i++){

cout << X[i] << " ";

}

cout << endl;

sinh();

}

}

Các bước thực hiện:

1. Nhập n và k.
2. Khởi tạo cấu hình đầu tiên.
3. In ra tổ hợp hiện tại.
4. Gọi sinh() để tạo tổ hợp tiếp theo.
5. Lặp lại cho đến khi không còn tổ hợp mới.

**5. Điều kiện dừng**

Biến ok kiểm soát vòng lặp:

* Nếu ok = false, nghĩa là đã sinh đến cấu hình cuối cùng (3 4 5 khi n=5, k=3).
* Khi đó chương trình dừng lại.

**6. Độ phức tạp thuật toán**

* Vì có đúng C(n, k) = n! / (k! \* (n-k)!) tổ hợp được sinh ra, thuật toán có độ phức tạp **O(C(n, k))**.
* Mỗi lần sinh tổ hợp mới mất **O(k)** thời gian.
* Vì vậy, tổng thời gian chạy là **O(k \* C(n, k))**, phù hợp với bài toán tổ hợp.

**7. Tổng kết**

* Thuật toán sử dụng mảng X để lưu tổ hợp hiện tại.
* Duyệt từ cuối về đầu để tìm phần tử có thể tăng giá trị.
* Nếu tìm được, tăng giá trị phần tử đó và cập nhật các phần tử sau.
* Dừng lại khi không còn cấu hình mới.
* Hiệu quả và đơn giản để sinh tất cả tổ hợp chập K của N.