# 组合

# 简介

int array[N];

现在要在array中区k个元素，共有种情况，遍历每种情况。

# 举例说明

N=6;

k= 3;

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **第i次** | **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | 1 | 2 | 3 |  |  |  |
| 2 | 1 | 2 |  | 4 |  |  |
| 3 | 1 | 2 |  |  | 5 |  |
| 4 | 1 | 2 |  |  |  | 6 |
| 5 | 1 |  | 3 | 4 |  |  |
| 6 | 1 |  | 3 |  | 5 |  |
| 7 | 1 |  | 3 |  |  | 6 |
| 8 | 1 |  |  | 4 | 5 |  |
| 9 | 1 |  |  | 4 |  | 6 |
| 10 | 1 |  |  |  | 5 | 6 |
| 11 |  | 2 | 3 | 4 |  |  |
| 12 |  | 2 | 3 |  | 5 |  |
| 13 |  | 2 | 3 |  |  | 6 |
| 14 |  | 2 |  | 4 | 5 |  |
| 15 |  | 2 |  | 4 |  | 6 |
| 16 |  | 2 |  |  | 5 | 6 |
| 17 |  |  | 3 | 4 | 5 |  |
| 18 |  |  | 3 | 4 |  | 6 |
| 19 |  |  | 3 |  | 5 | 6 |
| 20 |  |  |  | 4 | 5 | 6 |

20=

# 时间复杂度

O()

# 空间复杂度

O(k)

# 算法

## 容器

int index[k+1]; // 存储索引值

## 步骤

1. 初始化index

for i = 1 🡪 k

index[i] = i

2. 循环，当is\_finish 为true时，循环结束

while (!is\_finish)

{

while (index[k] != N + 1)

{

// 最后一个索引递增

index[k]++;

}

for j = k – 1 🡪 0

{

if (1==j && index[1] == 当前位置的最大值)

{

is\_finish = true;

break;

}

// 当前位置的最大值为N – (k-j)=N-k+j

if (index[j] < 当前位置的最大值)

{

index[j]++;

// 更新之后的位置，即j+1到k

for p = j+1 🡪k

index[p] = index[j] + p – j;

// 跳出继续循环，即index.last()继续++

break;

}

}

}

## [示例源码](2-src/combination.cc)1

## [示例源码2](2-src/ACM常用模板_排列组合生成.txt)

# 题目

## [hdoj 2660](../../../7-hdoj/2-problems/2-2660/2660.docx)