

生成排列/组合

Generating permutations and combinations

按字典序生成下一个排列

- **Algorithm:** 从后往前寻找第一个比后一项小的项 a_j (即 $a_j < a_{j+1}$ 且 $a_{j+1} > a_{j+2} > \dots > a_n$)，将 a_{j+1}, \dots, a_n 中大于 a_j 的最小项 a_k 放在第 j 个位置上，然后从小到大列出 a_j, \dots, a_n 中其他元素，放在第 $j+1$ 到第 n 个位置上。
- **Complexity:** $O(n)$
- **Code:**

```
1  inline bool nextPermutation(int a[], int n){
2  // if return false, then all the permutations have been generated
3      int j = n - 1, k = n;
4      while(a[j] >= a[j+1]) j--;
5      if(!j) return false;
6      while(a[j] >= a[k]) k--;
7      swap(a[k], a[j]);
8      int l = j + 1, r = n;
9      while(r > l) swap(a[l++], a[r--]);
10     return true;
11 }
```

按字典序生成下一个位串

- **Algorithm:** 模拟二进制数加一，即从后往前找到第一个 0，将其变成 1 并将其右边所有 1 变成 0。
- **Complexity:** $O(n)$
- **Code:**

```
1  inline void nextBitString(char s[], int n){
2  //s[n-1]s[n-2]...s[0] is a bit string which is not 11...1
3      int k = 0;
4      while(s[k] == '1'){
5          s[k] = '0';
6          k++;
7      }
8      s[k] = '1';
9  }
```

按字典序生成下一个 r 组合

- **Algorithm:** 从后往前找到第一个 $a_i \neq n - r + i$ 的 a_i ，用 $a_i + 1$ 代替 a_i ，且对于 $j = i + 1, i + 2, \dots, r$ ，用 $a_j + j - i + 1$ 代替 a_j 。
- **Complexity:** $O(n)$
- **Code:**

```
1  inline bool nextCombination(int a[], int n, int r){ // n elements, r-combination
2      int i = r;
3      while(a[i] == n - r + i)    i--;
4      if(!i) return false;
5      a[i] = a[i] + 1;
6      for(int j = i + 1; j <= r; j++)
7          a[j] = a[i] + j - i;
8      return true;
9  }
```