[10] 暴力枚举

深入浅出程序设计竞赛

第 2 部分 – 初涉算法

V 2021-02

洛谷

www.luogu.com.cn

洛谷

版权声明

- 其它《深基》配套资源、购买本书等请参阅：https://www.luogu.com.cn/blog/kkksc03/IPC-resources
- 如果课件有任何错误，请在这里反馈 https://www.luogu.com.cn/discuss/show/296741

本章知识导图


暴力枚举
循环枚举
例 1：统计方形加强版
减少枚举量
去掉重复情况
转换思维，枚举其他要素
例 2：烤鸡
例 3：三连击升级版
多重循环枚举
减少枚举的对象
子集枚举
例 4：选数
抽象成集合
并集、交集、包含、属于、补集
例 5：组合的输出
集合的枚举
枚举子集的复杂度分析
排列枚举
例 3（重现）：三连击升级版
枚举排列的原理
使用 next_permutation() 求下一个序列
例 6：全排列问题
例 7：火星人
next_permutation() 的熟练使用
枚举排列的复杂度分析

第 10 章 暴力枚举

求解问题

一般而言，算法问题可以分为两类：

模拟问题（第8章）

构造具体描述，使用计算机还原现实规则。

每一步的行为大多是简单而确定的，只需按照既定指示行动。

常基于数据结构，优化单步操作的复杂度。

求解问题（本章）

在全部的可能性中，搜索可行解或最优解。

解是不确定的，需要枚举/遍历/检索以尝试所有可能性。

常使用迭代、搜索等算法求解，优化主要为记忆化、剪枝。

枚举

本章我们介绍最简单的枚举求解算法，这一做法非常直接。

所谓枚举，即按照一定顺序，不重复、不遗漏地逐个尝试。

虽然需要消耗大量的时间，但是思路和编程都非常简单，保证可以取得正确结果。因而往往也被称为暴力（Brute-Force）算法。

用户名：kkksc03
密码：******

已知 kkksc03 的密码是六位数字
可从 000000 枚举到 999999
如果不限时间次数，一定能获得答案！

验证复杂的程序的正确性，可写功能一致的暴力对照程序，并构造小规模输入数据，比较二者输出。

这一过程称作对拍，在赛场上是非常实用的查错技巧。

循环枚举

循环枚举的意思是使用多重循环枚举所有的情况。简单，粗暴，有效。

请翻至课本 P142

循环枚举

循环是最简单的枚举方案。

我们将基于几个例题，讲解循环枚举的思路，并简介几种优化：

1. 基于数学的优化
2. 基于剪枝的优化
3. 技巧：打表法

统计方形加强版

例10.1 （洛谷 P2241，NOIP1997 普及组 加强）

有一个 $n\times m(n,m\leq 5000)$ 的棋盘，求其方格包含多少个（四边平行于坐标轴的）正方形和长方形。

本题中，长方形中不包括正方形。

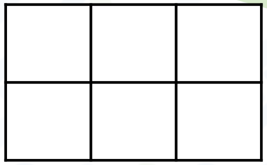
样例输入：

```
2 3
```

样例输出：

```
8 10
```

样例解释：



如图，正方形一共 8 个，长方形 10 个

正方形中，边长为 1 的 6 个，边长为 2 的 2 个；

长方形中，1×2 的 4 个，1×3 的 2 个，2×1 的 3 个，2×3 的 1个。

统计方形加强版

四个参数可以确定一个长方形：左下顶点坐标 (x1,y1) 和右上顶点坐标 (x2,y2)，也可以理解为左右下上 (x1,x2,y1,y2)。

思路 0：用四重循环，直接枚举 4 个参数，即两横边两竖边。

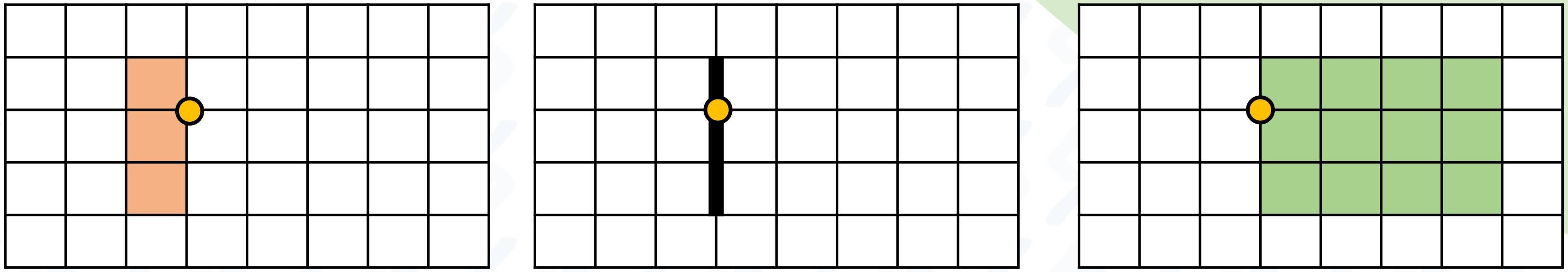
```
LL rec = 0, squ = 0; // LL是已经定义好的long long

for (LL x1 = 1; x1 <= n ; x1++)
for (LL y1 = 1; y1 <= m; y1++)
for (LL x2 = x1 + 1; x2 <= n; x2++)  // x2从x1+1起，不重复不遗漏
for (LL y2 = y1 + 1; y2 <= m; y2++)  // 同上
    if (x2-x1 == y2-y1)  // 正方形，两边长相等
        square++;
    else                 // 长方形
        rectangle++;
```

统计方形加强版

为什么是 x2 从 x1+1 开始，y2 从 y1+1 开始枚举？

- 如果 x1 > x2，那么 x1 就不再是左侧了，x2 才是左侧（左图）
- 如果 x1 = x2，那么无法构成长方形，退化为一根线段（中图）
- 只有 x1 < x2，才能正常构成长方形（右图）。y1、y2 同理



所以，这样可以保证不重复地遍历所有的方形。

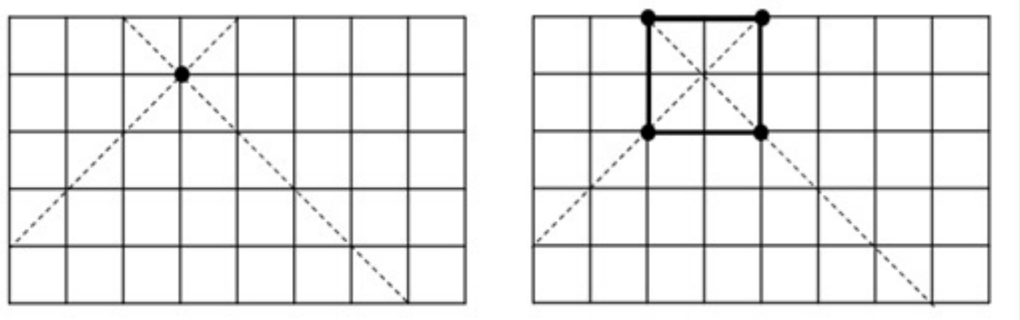
根据循环的范围可知，我们也没有遗漏任何的方形。

时间复杂度$O(n^2m^2)$……似乎有点慢。但是至少，答案正确了。

统计方形加强版

为了减少枚举量，可以考虑拆解问题，尝试解决简单问题。

例如，固定其中一个顶点为(x, y)，计算长/正方形个数。



思路 1：以左图中的点 (3, 4)为例（左下角为原点）。

位于同一对角线（图中虚线）上所有点均可构成正方形。

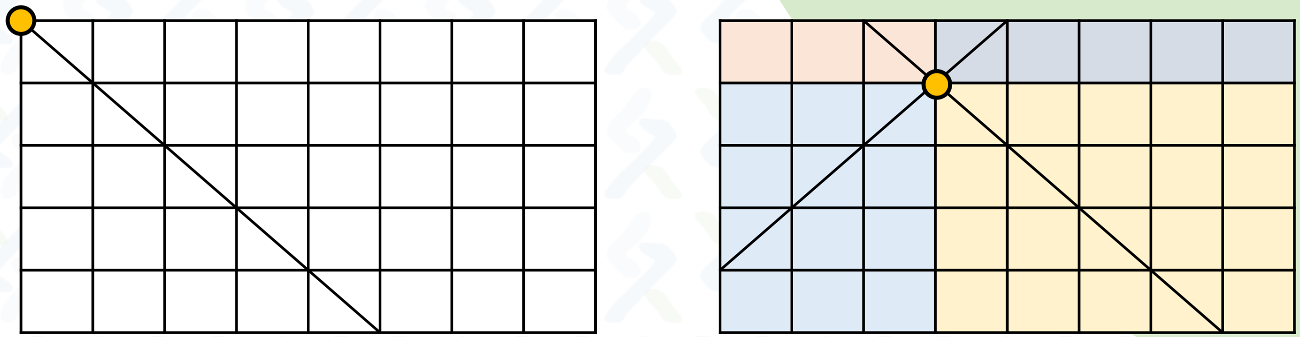
除自身所在行列外，所有其它点均可与其构成长方形；故直接得长方形数为 nm – 正方形数。

从右图可以看出，每一个长/正方形均被其 4 个顶点各计算一次。因此，最终答案需要除以 4。

统计方形加强版

斜线上的格点个数为多少呢？

若顶点在长方形顶点，格点个数为长方形的短边长，即 min(n,m)。



否则，以顶点为界分为4份。每份都这样计算，得到正方形个数：

$$\min(x,y)+\min(n-x,y)+\min(x,m-y)+\min(n-x,m-y)$$

因此，枚举(x, y)后，可在 $O(1)$ 时间内计算答案，求和。

总时间复杂度 $O(nm)$，直接优化掉一个 $O(nm)$！

统计方形加强版

还能不能更快呢？

思路 2：每一个长方形重复了4次。能否不重复呢？

结合思路 0 可知，只需考虑右上角为 (x, y) 的情况，因此计算斜线上的顶点时，只需要向左下角一个方向拓展！

（先算 4 个方向，再除以 4，可谓是画蛇添足啊……）

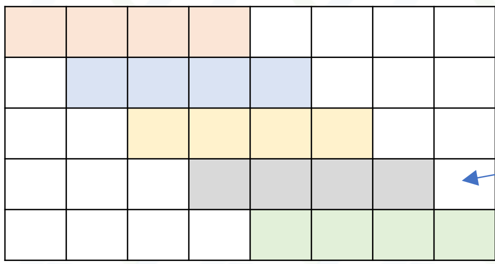
```
for (LL x = 0; x <= n; x++)
  for (LL y = 0; y <= m; y++) {
    LL tmp = min(x, y);
    squ += tmp;
    rec += x * y - tmp;
  }
```

当然，这里选择固定其它角也是等价的，但是固定右上角最简单。

注意：这里的原点是在左下角，列是 x，行是 y。如果选择左上角作为原点，那么枚举的就是长方形右下角。

统计方形加强版

思路 3：枚举边长 (a, b)。题目变为在 n×m 的长方形中能放置多少个 a×b 的方形。注意 a、b 有序，a×b 和 b×a 不等价。



8 列选
连续 4 列
8-4+1=5
种可能

```
for (LL a = 1; a <= n; a++)
  for (LL b = 1; b <= m; b++)
    if (a == b)
        squ += (n - a + 1) * (m - b + 1);
    else
        rec += (n - a + 1) * (m - b + 1);
```

- n 列中选连续 a 列：[1,a],[2,a+1],⋯,[n-a+1,n]，共 n-a+1 种可能。
- m 行中选连续 b 行构成方形，同理有 m-b+1 种情况。

所以 n×m 的长方形中可容纳 (n-a+1)×(m-b+1) 个 a×b 的矩形。

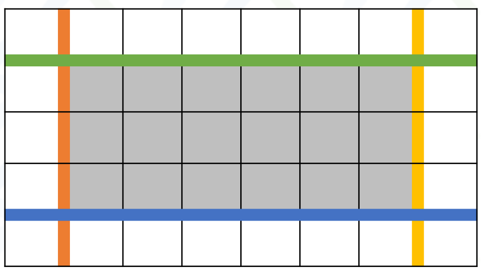
对于边长 k，只有长等于宽，才能构成正方形；其余均为长方形。

如果 a=b，就计入正方形。使用循环枚举 a、b，累加求和即可

统计方形加强版

思路 4：沿用刚才乘法原理的思想，枚举量还可进一步减少？

在 n×m 的长方形中的矩形个数，等价于在 m+1 条行线中选首尾 2 行、在 n+1 条列线中选首尾 2 列所围成的方形数目！



9 列线中选 2 列线 $\frac{1}{2}\times 9(9+1)$ 种可能

```
for (LL a = 1; a <= min(m, n); a++)
    squ += (n-a+1) * (m-a+1);

rec = n * (n+1) * m * (m+1) / 4 - squ;
```

在 n+1 列中选 2 列线围长方形，左列 n+1 种情况；右线列不能和左列线重复，只有 n 种情况，将他们乘起来。

由于重复统计（左右和右左），要除以2，有 $\frac{1}{2}n(n+1)$ 种可能。

同理，行有 $\frac{1}{2}m(m+1)$ 种可能。一共$\frac{1}{2}m(m+1)\ \frac{1}{2}n(n+1)$矩形。

借助思路 3，去掉其中的正方形。时间复杂度降到 $O(\min(m,n))$。

烤鸡

例 10.2 （洛谷 P2089）

烤鸡时需要加入 10 种配料，每种配料可放 1 到 3 克。

美味程度是所有配料质量之和。给出一个美味程度 $n(n \leq 5000)$，按照字典序输出配料搭配的方案数量和所有的搭配方案。

如果 n 超过 30 请输出0。

样例输入：

```
11
```

样例输出：

```
10
1 1 1 1 1 1 1 1 1 2
1 1 1 1 1 1 1 1 2 1
1 1 1 1 1 1 1 2 1 1
1 1 1 1 1 1 2 1 1 1
1 1 1 1 1 2 1 1 1 1
1 1 1 1 2 1 1 1 1 1
1 1 1 2 1 1 1 1 1 1
1 1 2 1 1 1 1 1 1 1
1 2 1 1 1 1 1 1 1 1
2 1 1 1 1 1 1 1 1 1
```

烤鸡

思路 1：

由于这题要求输出所有的具体方案，所以公式不管用了。

那就硬着头皮写暴力枚举吧。

首先，每一种调料至少为 1，所以必有 n≥10。

同样，每一种调料最多为 3，所以必有 n≤30。

一共有 10 种调料，所以 10 重循环 即可解决。

```
#define rep(i, a, b) for (int i = a; i <= b; i++)

rep(a, 1, 3) rep(b, 1, 3) rep(c, 1, 3) rep(d, 1, 3) rep(e, 1, 3)
    rep(f, 1, 3) rep(g, 1, 3) rep(h, 1, 3) rep(i, 1, 3) rep(j, 1, 3)
        if (a + b + c + d + e + f + g + h + i + j == n)
            ans++;
printf("%d\n", ans);
rep(a, 1, 3) rep(b, 1, 3) rep(c, 1, 3) rep(d, 1, 3) rep(e, 1, 3)
    rep(f, 1, 3) rep(g, 1, 3) rep(h, 1, 3) rep(i, 1, 3) rep(j, 1, 3)
        if (a + b + c + d + e + f + g + h + i + j == n)
            printf("%d %d %d %d %d %d %d %d %d %d\n", a,b,c,d,e,f,g,h,i,j);
```

宏定义
构造语句
简化编码

烤鸡

不过，不能上公式，不代表不能用其他方法优化了。

思路 2：

如果 a+b+c+d+e 已经超过n，那么 f、g、h、i 无论如何取值，都不可能使得答案再等于 n 了。

因为 f、g、h、i 至少取 1，所以 a+b+c+d+e 达到 n-3 或以上的时候，就已经没有任何的可能性了。

基于上述事实，可以构造出剪枝算法：限定每一个数的范围。

烤鸡

以 e 作为例子。枚举 e 时，a、b、c、d的值已经确定了。

由于 e、f、g、h、i 至少为1，所以e的最大值是n-1×5-(a+b+c+d)。

由于 e、f、g、h、i 至多为3，所以e的最小值是n-3×5-(a+b+c+d)。

故：n-15-a-b-c-d≤e≤n-5-a-b-c-d

```
#define rep(i, a, b) for (int i = max(1, a); i <= min(3, b); i++)
// 注意这里rep的定义有所变化！
rep(a, n-27, n-9)
rep(b, n-24-a, n-8-a)
rep(c, n-21-a-b, n-7-a-b)
rep(d, n-18-a-b-c, n-6-a-b-c)
rep(e, n-15-a-b-c-d, n-5-a-b-c-d)
rep(f, n-12-a-b-c-d-e, n-4-a-b-c-d-e)
rep(g, n-9-a-b-c-d-e-f, n-3-a-b-c-d-e-f)
rep(h, n-6-a-b-c-d-e-f-g, n-2-a-b-c-d-e-f-g)
rep(i, n-3-a-b-c-d-e-f-g-h, n-1-a-b-c-d-e-f-g-h)
rep(j, n-a-b-c-d-e-f-g-h-i, n-a-b-c-d-e-f-g-h-i) {
    li[ans][0]=a;li[ans][1]=b;li[ans][2]=c;li[ans][3]=d;
    li[ans][4]=e;li[ans][5]=f;li[ans][6]=g;li[ans][7]=h;
    li[ans][8]=i;li[ans][9]=j;
    ans++;
}
```

由于题目要求先输出方案数再输出答案，所以要用数组记录答案，避免重复枚举。

每种调料只有最多 3 种可能，所以答案不会超过 $3^{10}=59049$。

三连击升级版

例 10.3 （洛谷 P1618）

将 1，2，…，9 共 9 个数分成三组，分别组成三个三位数，且使这三个三位数的比例是 $A:B:C(A<B<C<10^9)$，试求出所有满足条件的三个三位数，若无解，输出 No!!! 。

样例输入：

```
1 2 3
```

样例输出：

```
192 384 576
219 438 657
273 546 819
327 654 981
```

三连击升级版

只要枚举第一个三位数，从 123 枚举到 987。

就可以根据比例确定另两个（不一定存在）

最后检验得到的结果是否总共同时具有 9 个不同数位。

已知 $x:y:z=A:B:C$，和 x 的值

y 和 z 是 多少呢？

x=192	y=384	z=576
A:B:C=1:2:3		

b[0]	b[1]	b[2]	b[3]	b[4]	b[5]	b[6]	b[7]	b[8]	b[9]
1	1	1	1	1	1	1	1	1	1

```
int b[10];
void go(int x) { // 分解三位数到桶里
  b[x % 10] = 1;
  b[x / 10 % 10] = 1;
  b[x / 100] = 1;
}
bool check(int x, int y, int z) {
  memset(b, 0, sizeof(b));
  if (y > 999 || z > 999) return 0;
  go(x), go(y), go(z);
  for (int i = 1; i <= 9; i++)
    if (!b[i]) return 0;
  return 1;
}
```

```
int main() {
  long long A, B, C, x, y, z, cnt = 0;
  cin >> A >> B >> C;
  for (x = 123; x <= 987; x++) {
    if (x * B % A || x * C % A) continue;
    y = x * B / A, z = x * C / A;
    if (check(x, y, z)) {
      printf("%lld %lld %lld\n",x,y,z);
      cnt++;
    }
  }
  if (!cnt) puts("No!!!");
  return 0;
}
```

子集枚举

To be or not to be, that is the question. 每一个元素都可以选择或者不选，所以选不选呢？

请翻至课本 P147

子集枚举

有时候问题可以归纳为这样一个情况：

在若干元素当中，选择其中一些以达成特定条件。

我们可以像《烤鸡》一题中那样，将每一个元素视为有两种情况：选或不选。然后使用多重循环解决问题。

#1	#2	#3
三个元素 可用三重循环		

#1	#2	#3	#4	#5
五个元素 可用五重循环				

#1	#2	……	#9	#10
10 个元素 难道要写 10 重循环？				

对于这种特殊的问题，可以使用子集枚举。

子集枚举

注意到每一个元素只有两种状态。联想到二进制。

用第 i 个二进制位来表达第 i 个元素的取舍。习惯上，1表示选取，0表示丢弃。那么一个 n 位二进制整数就可以表达 n 个元素的取舍，即：

25 =

1	1	0	0	1
取4	取3	舍2	舍1	取0

= {0, 3, 4}

为什么二进制的最右边的一位，代表第一个元素呢？

可用 C++ 自带的位运算符，轻松实现集合的各项运算。

交	x & y	{0,3,4} ∩ {1,2,3} = {3}	25 & 14 = 8	11001 & 01110 = 01000
并	x | y	{0,3,4} ∪ {1,2,3} = {0,1,2,3,4}	25 | 14 = 31	11001 & 01110 = 11111
对差	x ^ y	{0,3,4} ⊕ {1,2,3} = {0,1,2,4}	25 ^ 14 = 23	11001 ^ 01110 = 10111
补	((1<<n) - 1) ^ x	{0,1,2,3,4} – {0,3,4} = {1,2}	31 ^ 25 = 6	11111 ^ 11001 = 00110
差	x ^ (x & y)	{0,3,4} – {1,2,3} = {0,4}	25 ^ 8 = 17	11001 ^ 01000 = 10001
判定	(1<<a) & x	3 ∈ {0,3,4}	(1<<3)&25 != 0	01000 & 11001 = 01000

子集枚举

这种二进制表示法又称为“状态压缩”。将来介绍状态压缩动态规划的时候，会再用到。

0	0	0	0	0
五个元素全 0，S=0				

1	1	1	1	1
五个元素全 1，S=31				

有了二进制表示后，就可以通过 for 循环遍历 0（全0，即空集）到 2^n-1（全 1，即全集）的所有整数，得到所有可能的子集。

```
int U = 1 << n; // U = 2^n, U-1即为全集
for (int S = 0; S < U; S++)  // 枚举所有子集[0,U)
if (/*满足题设条件*/) {
    // 进行统计
}
```

优点：非常简单方便，集合运算可以基于整数运算 $O(1)$ 实现。

限制：元素个数不能太多。一般不超过 20 个。

选数

例10.4 （洛谷 P1036，NOIP2002 普及组）

从 n $(n \le 20)$ 个整数中任选 k 个整数相加，求有多少种选择情况可以使和为质数。

样例输入：

```
4 3
3 7 12 19
```

样例输出：

```
1
```

选 3+7+19=29，一共只有一种。

选数

进行子集枚举之后，求和做质数判定。

题目要求选k个整数，且和为质数。所以判定条件就是：

1. 二进制表示中恰好包含 k 个 1。
2. 将对应元素取出后，使用质数判定算法发现是质数。

#0 3	#1 7	#2 12	#3 19
取	取	不取	取
以上情况对应二进制 1011 此时 S 的值就是 11			

```
bool eval(int S) {
    int sum = 0;
    for (int i = 0; i < n; i++)
        if (S & (1 << i)) sum += a[i];
        // 如果第i个元素在S中，求和
    return is_prime(sum);  // 质数判定
}

int U = 1 << n; // U = 2^n, U-1即为全集
for (int S = 0; S < U; S++)  // 枚举所有子集[0,U)
//        恰好包含k个1                通过质数判定
if (__builtin_popcount(S) == k && eval(S))
    ans++;  // 统计
```

排列枚举

排列枚举，顾名思义，就是要求枚举所有元素排列。如果想知道一些元素所有的排序方法，就需要枚举排列。

请翻至课本 P149

全排列问题

例 10.6 （洛谷 P1706）

输出自然数 1 到 n 所有不重复的排列，即 n 的全排列。

所谓全排列，指将1~n所有数字不重复、不遗漏地构成数列的所有可能情况。要求按字典序输出。

样例输入：

```
3
```

样例输出：

```
1 2 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

全排列问题

第一个排列当然是 1 2 3 4 … n。

如何生成第二个排列呢？

我知道是 1 2 3 … n n-1
接下来的讲解有一点难
实在搞不懂原理的话可跳过

我们通常使用的十进制数，可以认为是一种特殊的排列：允许各位数字可以相同。在十进制中，生成下一个排列等价于 +1。

十进制数如何处理+1呢？可以近似写作如下算法：

1. 令最低位为当前位。
2. 令当前位+1。
3. 如果结果为10，则产生进位，当前位前移，转(2)。否则结束。

全排列问题

再回到全排列。全排列要求每一项都必须不同，所以可以理解为每一次“+1”都会产生进位。于是问题就转变为，需要进多少位？

n	n-1	…	2	1
最后一个排列				

1	n	n-1	…	2
以 1 开头的最后一个排列				

共同特点：有一个单调递减后缀！

猜想：每次“+1”需要进位到最长单调递减后缀的前一项。

以 1 3 2 5 4 为例，单调递减后缀为5 4。所以需要进位到2。
1 3 已经被使用，所以2的下一项为4。得前三项为1 3 4。
还剩下2和5，依次排列。综上，下一个排列是1 3 4 2 5。

全排列问题

进位之后，高位仍然是可以保证不变的。

而进位则需要将当前位p替换为后缀中大于p的最小值q。如下图：

Before		p	单调递减	q	单调递减

After		q	单调递增	p	单调递增

其中，p是大于q的最小元素。于是得到代码：

```
bool next_permutation(int a[], int n) {
    int p, q;
    for (p = n-1; p >= 1 && a[p] > a[p+1]; p--);  // 寻找单减后缀
    if (p <= 0) return false;  // 已经是最后一个排列了
    for (int i = p+1, j = n; i <= j; i++, j--)
        swap(a[i], a[j]);  // 翻转单减后缀为单增
    for (int i = p+1; i <= n; i++)
        if (a[i] > a[p]) {q = i; break; } // 找到最小大于p的值
    swap(a[p], a[q]);
    return true;  // 生成了新的排列
}
```

全排列问题

其实万能的STL里面也是有这个算法的，头文件和函数：

```
#include<algorithm> //头文件
bool next_permutation(begin, end); //函数原型
```

和大多数 STL 函数一样，begin 指数组首地址，end 指末地址+1。

```
do {
    for (int i = 1; i <= n; i++)
        printf("%5d", a[i]);
    puts("");
} while (next_permutation(a + 1, a + n + 1));
```

生成 next_permutation 的时间复杂度 O(n)。

一共有 O(n!) 个可能的排列，总复杂度为O(n×n!)。

对于本题而言，这个复杂度是可以接受的。

对于其他问题而言，一定要根据题目需要，设计合适的剪枝算法！

三连击升级版（重现）

例 10.3 （洛谷 P1618）

将 1，2，…，9 共 9 个数分成三组，分别组成三个三位数，且使这三个三位数的比例是 $A:B:C(A < B < C < 10^9)$，试求出所有满足条件的三个三位数，若无解，输出 No!!! 。

另外一种解法：直接枚举三个三位数再检验！每次变排列。

b[1]	b[2]	b[3]	b[4]	b[5]
1	2	3	4	5
b[6]	b[7]	b[8]	b[9]	
6	7	8	9	

b[1]	b[2]	b[3]	b[4]	b[5]
1	2	3	4	5
b[6]	b[7]	b[8]	b[9]	
6	7	9	8	

```
for (int i = 1; i <= 9; i++)
  a[i] = i;
do {
  x = a[1] * 100 + a[2] * 10 + a[3];
  y = a[4] * 100 + a[5] * 10 + a[6];
  z = a[7] * 100 + a[8] * 10 + a[9];
  if (x * B == y * A && y * C == z * B)
  //避免浮点误差和爆long long的小技巧
    printf("%lld %lld %lld\n", x, y, z), cnt++;
} while (next_permutation(a + 1, a + 10));
if (!cnt)puts("No!!!");
```

课后习题与实验

学而时习之，不亦说乎。学而不思则罔，思而不学则殆。——孔子

请翻至课本 P112

复习

求解问题 在若干种选项中，寻找可行解/最优解的问题。

暴力枚举 遍历所有选项解决求解问题的手段。

循环枚举 数学优化，剪枝优化，打表法

子集枚举 二进制状态压缩，__builtin_popcount

排列枚举 排列生成算法，next_permutation

作业

习题 10.1 涂国旗（洛谷 P3392）

一个由 $N \times M$ 小方块组成的旗帜，且满足以下条件是俄罗斯国旗。

1. 从最上方若干行（不小于 1）的格子全部是白色的。
2. 接下来若干行（不小于 1）的格子全部是蓝色的。
3. 剩下的行（不小于 1）全部是红色的。

有一个 $N \times M$ 的矩阵，每个格子是蓝、白、红色之一。希望改动最少数量的格子，使其变成俄罗斯国旗，至少要修改多少个格子？

习题 10.2 First Step（洛谷 P3654）

现有 $N \times M (N, M \leq 100)$ 小方格组成的篮球场，每个小方格可能是 .（空地）或者是 #（障碍）。

现在有一列 $1 \times K$ 的队员，每人占用一小格，排成一排站在空地上（可横可竖），求站位方案数量。

作业

习题 10.3 回文质数（洛谷 P1217，UASCO Training）

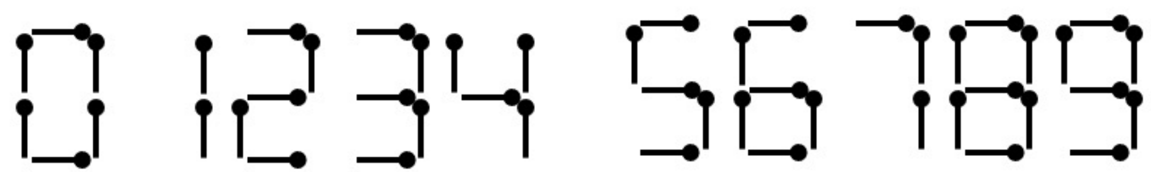
写一个程序来找出范围 $[a,b](5 \le a < b \le 10^8)$ 间的所有回文质数。

例如 151 既是一个质数又是一个回文数(从左到右和从右到左是看一样的)，所以 151 是回文质数。

习题 10.4 火柴棒等式 （洛谷 P1149，NOIP 2008）

给你 n 根火柴棍，你可以拼出多少个形如 “A+B=C” 的等式？

等式中的 A、B、C 是用火柴棍拼出的整数（若该数非零，则最高位不能是 0），问能拼成多少种不同的等式。



用火柴棍拼数字 0-9 的拼法如图所示。

作业

习题 10.5 妖梦拼木棒（洛谷 P3799）

现有 $n(n \leqslant 100000)$ 根长度不超过 5000 的木棒，从中选取 4 根组成一个等边三角形，请问有几种选法？

习题 10.6 kkksc03 考前临时抱佛脚（洛谷 P2392）

有四个科目的作业，每个科目有不超过20题，解决每道题都需要一定的时间。kkk 可以同时处理同一科目的两道不同的题，求他完成所有题目所需要的时间。

参考阅读材料

以下的内容限于课件篇幅未能详细阐述。如果学有余力，可自行翻阅课本作为扩展学习。

- P149 例 10.5：按照字典序来枚举子集
- P151 例 10.7：熟练使用 next_permutation()
- 习题 17.7， 17.8