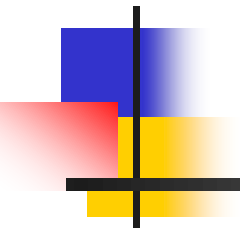


枚举思想

枚举（ Enumerate ）

- 枚举就是尝试每一种可能的情况，直到找到一个合法的解。
- 一些问题看似很复杂，但当数据量不大的时候，用枚举往往能解决。
- 枚举的关键在于准确描述解的特征和解空间的结构，然后遍历整个解空间寻找合法解。

枚举的要点

给出解空间

建立简洁的数学模型。
枚举的时候要想清楚：可能的情况是什么？要枚举哪些要素？

减少枚举的空间

枚举的范围是什么？是所有的内容都需要枚举吗？
在用枚举法解决问题的时候，一定要想清楚这两件事，否则会带来不必要的时间开销。

选择合适的枚举顺序

根据题目判断。比如题中要求的是最大的符合条件的素数，那自然是从大到小枚举比较合适。

枚举的一般步骤

1. 确定枚举范围，选取合适的枚举方式，不能遗漏任何一个真正解，同时要避免重复；
2. 看是否存在优化可能，将可能成为解的答案范围尽可能的缩小，以便提高解决问题的效率；
3. 根据问题找到合理、准确、好描述并且好编码的验证条件；
4. 枚举并判断当前的解是否符合验证条件，并保存符合条件的解；
5. 按要求输出枚举过程中留下的符合条件的解。

例题--百钱买百鸡问题

- 公鸡每只5元，母鸡每只3元，小鸡3只1元，用100块钱买100只鸡，问公鸡，母鸡，小鸡各多少只？

```c
#include <iostream>
int main()
{
    int x,y,z;//三个变量分别为公鸡，母鸡，小鸡的数量
    for(x=0;x<=20;x++)//公鸡最多20只
    {
        for(y=0;y<=33;y++)//母鸡最多33只
        {
            z=100-x-y;//小鸡的数量
            if (z%3==0 && x*5+y*3+z/3==100)//小鸡3只一元，所以小鸡数量应该是3的倍数
                printf("公鸡%d只,母鸡%d只,小鸡%d只\n",x,y,z);
        }
    }
    return 0;
}
```

例题----一元三次方程求解问题

- **求$ax^3+bx^2+cx+d=0$的解，保留2位小数，根的范围$-100 \leqslant x \leqslant 100$，且根与根之差的绝对值$\geqslant 1$，数据确保有3个实根。**

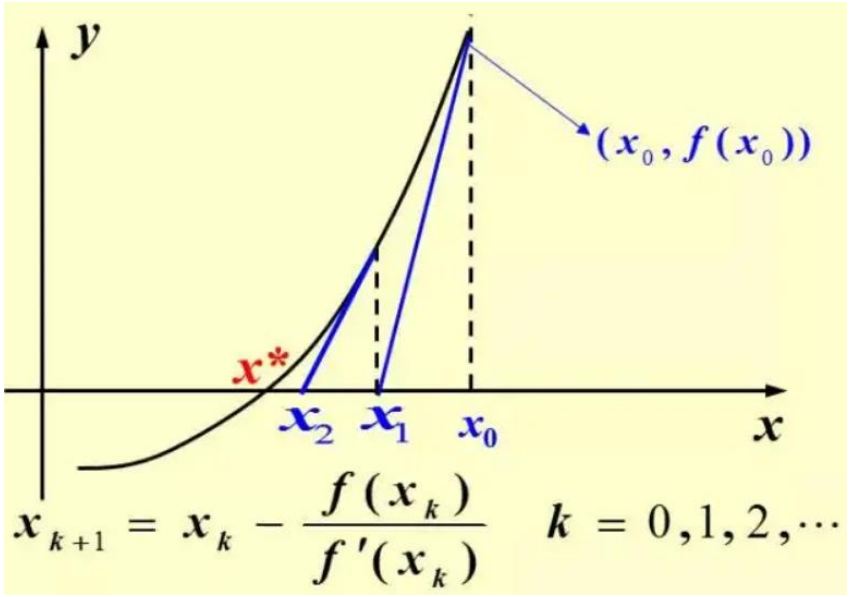
提示：记方程 f(x) = 0，若存在 2 个数 x1 和 x2，且 x1 < x2，f(x1)×f(x2)<0，则在 (x1,x2) 之间一定有一个根。(数学上的零点定理)

- 朴素想法：

```
for (double i = -100; i < 100; i += 0.001)
    判断f(i)*f(i+0.001)的正负;
```

- 优化一下：
  - 三个答案都在[-100, 100]范围内，两个根的差的绝对值≥1，保证了每一个大小为1的区间里至多有1个解，也就是说当区间的两个端点的函数值异号时区间内一定有一个解，同号时一定没有解。
  - 可以枚举互相不重叠的每一个长度为1的区间，在区间内进行二分查找。

- **另一种优化：**
  - **先对函数求导，$f'(x)=3ax^2+2bx+c$，求出函数极值点** p、q。**显然必有三个根在**$[-100, p)$，$[p, q]$，$(q, 100]$ **三个区间内。**
  - **然后用牛顿迭代法多次迭代即可。**



例题--零和对数

- **一个数组中的数互不相同，求其中和为0的数对的个数。** $0<|a_i|\leqslant 5000$

朴素想法：O(n^2)

```
for (int i = 1; i <= n; i++)
    for (int j = 1; j <= n; j++)
        if (a[i] + a[j] == 0)    ans++;
```

重复

一点点优化：O(n^2)

```
for (int i = 1; i <= n; i++)
    for (int j = 1; j < i; j++)
        if (a[i] + a[j] == 0)    ans++;
```

更多的优化：O(n)

```
1
2   bool met[MAXN * 2 +1];
3   memset(met, 0, sizeof(met));
4
5   for (int i=0;i<n; i++) {
6       met[MAXN + a[i]] =true;
7       if (met[MAXN - a[i]])
8           ans++;
9   }
10
```

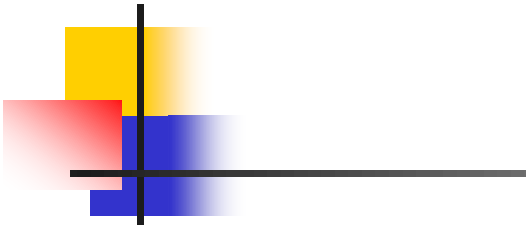
空间换时间

	-3	-2	-1	0	1	2	3
met	0	0	0	0	0	0	0
	0	1	2	3	4	5	6

a	1	-2	-1	2

例题--回文质数

- **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。写一个程序来找出范围 **[a,b]** 间所有的回文质数，**5 ≤ a < b ≤ 1e8**
- 朴素想法：
  - 暴力枚举[a, b]的每一个数，先判断是否为质数，再判断是否回文，**TLE**
  - 判断回文快，而回文数又少，所以先判断回文，再判断质数， **TLE**



- 优化一下：
  - 考虑用线性筛筛出所有质数，再判断回文，仍然TLE
- 再优化一下：
  - 除了11，偶数位的数不存在回文质数。
    4位，6位，8位，……，不存在回文质数，可以跳过

小结

- 枚举法的优点
  - 思路简单，程序编写和调试都比较方便。
- 枚举法的缺点
  - 运算量比较大，效率不高；
  - 如果枚举范围太大，在时间上可能会超时；需要设法优化。