递推、高精度加法

递推

定义：可以通过若干步可重复运算来描述复杂问题的方法，一般应用在序列计算上。

找出递推关系条件：存在子结构性质。

递推主要思想：通过子问题的解来构造（大问题的）答案。

递推关系式如何找出来?

（积累的题感、题目中描述中可以找出）

例题：数楼梯

$f[i]$: 存储走到 i 台阶的方法总数。

> 一般状态的定义与题目的问有关。

分析：走到台阶 i 有哪几种情况?

例如：走到台阶 4，可以从哪些台阶一步走上来。

分别可以从台阶 2 和台阶 3。

式子中表示出来：$f[4]=f[2]+f[3]$。

递推过程中很重要的一环：最小问题直接初始化。

$f[1]=1, f[2]=2;$

由此推出递推关系式：$f[i]=f[i-1]+f[i-2]$。

从台阶 3 开始进行递推到台阶 n。

时间复杂度：$O(n)$，每个台阶遍历一次，每次遍历只计算一次。

高精度

因为超过了计算机能够表示的整数范围，`long long` 能够存储最大的大小 2^{64}，大约是 18 位。

高精度加法

原理与数学中竖式加法一样进行计算。

1. 处理最低位->最高位，先处理相加。
2. 处理最低位->最高位，后处理进位。
3. 注意输出的时候，去除前导 0。

```cpp
const int N = 1e5+10; // 可能的最高位数
// 和，加数，加数
void add(int c[],int a[],int b[]){
  // 先做加法，从低位开始
  for(int i=0;i<N;i++)
      c[i] = a[i] + b[i];
  // 后处理进位，从低位开始
  for(int i=0;i<N;i++)
      if( c[i] > 9)
          c[i+1]++, c[i]-=10;
  // 输出，去除前导 0
  bool falg = false;
  for(int i=N-1;i>=0;i--){
      if( c[i]!=0 )
          falg = true;
      if( falg )
          cout << c[i];
  }
}
```

例题2：蜜蜂路线

巧妙的设计，通过将起点 m 转移到从蜂房 1 开始，将终点 n 转移到从蜂房 $n-m+1$ 开始，避免了高精度减法运算。

状态表示，$f[i]$：从起点 m 蜂房到 i 蜂房合法的路线数量。

递推关系式：$f[i]=f[i-1]+f[i-2]$。

初始化最小问题：$f[1]=f[2]=1$。

做法与数楼梯基本一致。
