基础动态规划

动态规划算法通常用于求解具有某种最优性质的问题，动态规划算法与分治法类似，其基本思想也是将待求解问题分解成若干个子问题，先求解子问题，然后从这些子问题问题的解得到原问题的解。动态规划的特点是决策满足最优化原理和最优子结构，状态转移无后效性。

求解动态规划问题的一般步骤，设计状态转移模型 -> 寻找状态转移方程 -> 求解

阶梯问题

1. 一个楼梯，共n层，每次可向上跨一步或两步，问走到第n层共有多少走法

```
状态模型：f(i) 为走到第 i 层的种类数
转移方程：f(i) = f(i-1) + f(i-2)
```

2. 一个楼梯，共n层，每次可向上跨二或三步，问走到第n层共有多少走法

```
转移方程：f[i] = f(i-2) + f(i-3)
```

3. 在问题 2 的基础上，每层台阶上都有一块蛋糕，每块蛋糕都有对应的美味值a[i]，走到相应的台阶能获得对应的美味值，问怎么走使得美味值最大

```
状态模型：dp(i) 为走到第 i 层能获得最大的美味值
转移方程：dp(i) = max{dp(i-2) , dp(i-3)} + a[i]
```

0-1 背包问题

1. 有 n 个物品，数量为1，他们分别有各自的重量 wi 和价值 vi ，每个物品只能选择取或不取，你有一个容量为W的背包，问最大能取多少总价值的商品

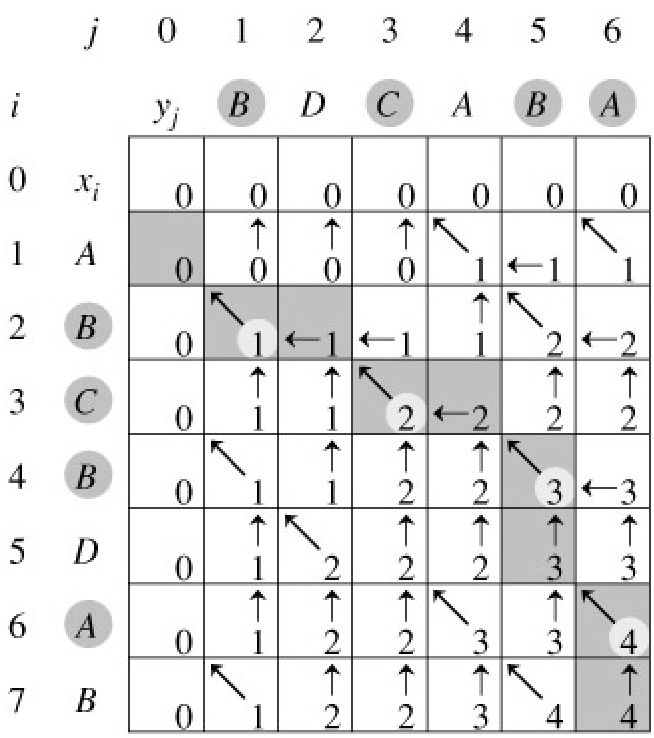
```
状态模型：f(i, w) 代表前 i 种物品容量为 w 时最多能装多少价值
转移方程：f(i, w) = max{f(i-1, w), f(i-1, w-vi) + vi}
```

最长公共子序列

1. 有两个字符串 s 和 t，求两个串最长的公共子序列的长度

```
input :
abcbdab
bdcaba

output:
bcba
```



```
状态模型：c(i, j) 代表 s 的前i个字符和 t 前j个字符最长公共子序列
转移方程：c(i, j) =
        0                          (i == 0 || j == 0)
        c(i-1, j-1) + 1            (s[i] == t[j])
        max{c(i-1, j), c(i, j-1)}  (s[i] != t[j])
```

基础数论

模运算

给定一个正整数p，任意一个整数n，一定存在等式n = kp + r； 其中k、r是整数，且满足0 <= r < p，称k为n除以p的商， r为n除以p的余数，表示成n % p = r

```
对于正整数和整数a, b, 定义如下运算:
取模运算: a % p (a mod p) , 表示a除以p的余数。
模p加法: (a + b) % p = (a%p + b%p) % p
模p减法: (a - b) % p = (a%p - b%p) % p
模p乘法: (a * b) % p = ((a % p)*(b % p)) % p
幂模p:   (a ^ b) % p = ((a % p)^b) % p
模运算满足结合律、交换律和分配律。
a≡b (mod n) 表示a和b模n同余，即a和b除以n的余数相等。
```

欧几里得算法(辗转相除法)

定理：gcd(a, b) = gcd(b, a % b) 证明：a = kb + r = kb + a%b，则a % b = a - kb。令d为a和b的公约数，则d|a且d|b 根据整除的组合性原则，有d|(a-kb)，即d|(a%b)。 这就说明如果d是a和b的公约数，那么d也一定是b和a%b的公约数，即两者的公约数是一样的，所以最大公约数也必定相等。 这个定理可以直接用递归实现，代码如下：

```
int gcd(int a, int b) {
    return b ? gcd(b, a%b) : a;
}
```

快速幂

计算 a ^ n % p, 1<a<1e9, 1<n<1e9, p为一个素数

```
解法一：
def solve(a, n, p):
    ans = 1
    for i = 1 to n:
        ans = ans * n
    return ans % p
```

```
解法二：(快速幂)
当已知 a^i为多少时，可以很轻易的知道 a^(2i) 的大小
相比于解法一，节省将近一倍的时间，我们将 a^n 的问题，不断向下分解成 a^(n/2), a^(n/2/2), a^(n/2/2/2)...
就能在log级时间内计算答案
def pow_mod(a, n, p):
    if n == 0: return 1
    res = pow_mod(a, n/2, p)
    res = res * res % p
    if n % 2 == 1: res = res * a % p
    return res
时间复杂度为O(logn)
```

素数筛

求 1 - n之间所有的素数，1<n<5e6

```
解法一:
def isprime(x):
    for i = 2 to sqrt(x):
        if x%i==0: return false
    return true

def solve():
    a = list
    for i = 2 to n:
        if isprime(i):
            i -> a
    return a
```

```
解法二:(素数筛)
def solve():
    a = list
    isprime = list //初始化2-n为true
    for i = 2 to n:
        if isprime(i):
            i -> list
            for j = i to n/i:
                isprime[j*i] = false
```