dp优化1：dp优化思路

董国梁



例1： 高楼实验

- 吉吉国王发现了 `n` 个奇怪的蛋，但是这些蛋非常的坚硬，吉吉国王发现他无论怎么敲这些蛋都不会碎，因此吉吉国王想要知道这些蛋的硬度。

- 由于没有衡量硬度的方法，吉吉国王决定自己创造一个。在吉吉王国有一栋高为 `m` 的楼房，如果一个蛋在第 `i` 层楼扔下去没有碎，但是从第 `i+1` 层楼扔下去碎了，那么我们就说这个蛋的硬度是 `i` 。如果在第 `m` 层扔下去都没有碎，那么这个蛋的硬度就是 `m` ；而如果在第1层摔下去就碎了，那么硬度就是0。

- 现在吉吉国王要用这 `n` 个蛋来测试这种蛋的硬度，他想知道在最坏的情况下，能够测出多大的硬度。

- $1 \leq n, m \leq 1000$

求解

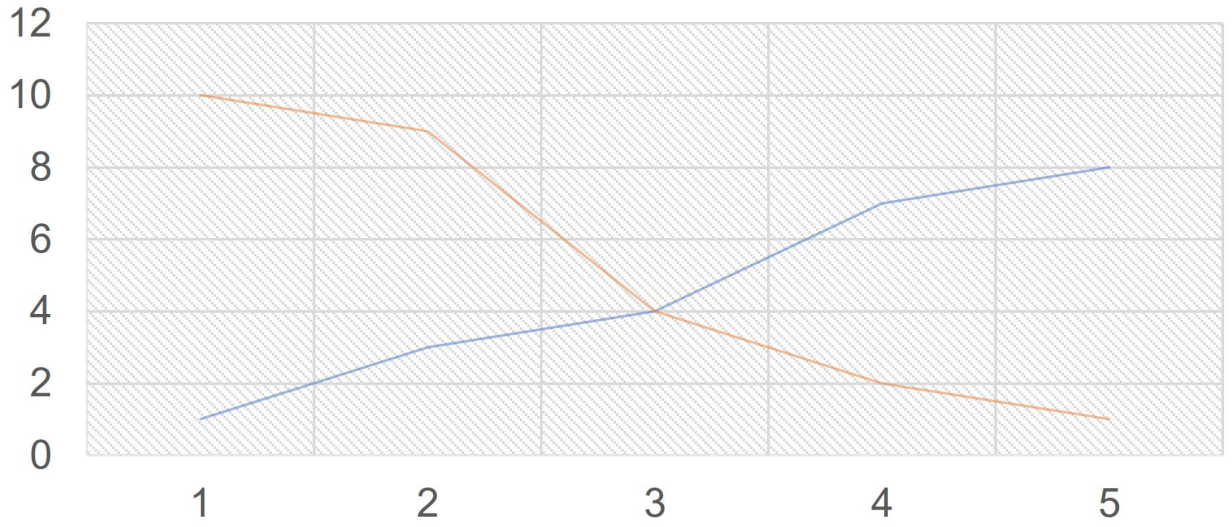
- **题目求的是 `n` 个蛋在 `m` 层上实验最坏情况下要确定硬度需要的最少次数，**
- **状态定义**：$f[i][j]$ 表示 `i` 个蛋在 `j` 层上实验最坏情况下要确定硬度需要的最少次数
- **边界条件**：
  - $f[i][0]=0$
  - $f[1][j]=j$ ——1个蛋的时候只能一层一层从下往上扔不然可能还没确定就碎了
- **状态转移**：
  - 考虑第一个蛋在哪层扔：在 `w` 扔两种结果：
  - 碎了：那么剩下的 $(i-1)$ 个蛋在 $1\sim(w-1)$ 层实验，最坏情况的最小次数为 $f[i-1,w-1]$
  - 没碎：用剩下的 `i` 个蛋在 $w+1\sim j$ 层实验——在 $w+1$ 到 `j` 层确定硬度和在1到 $j-w$ 层确定硬度用的蛋的数量是一样的
- **转移方程**：$f[i][j]=\min(\max(f[i-1][w-1],f[i][j-w])+1)$
- **时间复杂度**：$O(n^3)$

一个由“显然”引发的优化： ——观察问题本身的性质

- 如果蛋的个数无限多的话——类似猜数组对区间二分即可——$\log_2(n+1)$ 次即可完成
- 所以 `m` 大于 $\log_2(n+1)$ 时都可以按照二分查找去操作，直接输出 $\log_2(n+1)$ 即可
- **时间复杂度：** $O(n^2 \log_2 n)$

观察方程本身的性质进行优化

- $f[i][j] \geq f[i][j-1]$ 始终成立
- 所以 $f[i]$ 单调
- 再来看方程：$f[i][j] = \min(\max(f[i-1][w-1], f[i][j-w]) + 1)$
- `w` 增加 $f[i-1][w-1]$ 增加 $f[i][j-w]$ 减小
- **决策点**是 $f[i-1][w-1]$ 和 $f[i][j-w]$ 函数值的交点两边最近的整数点



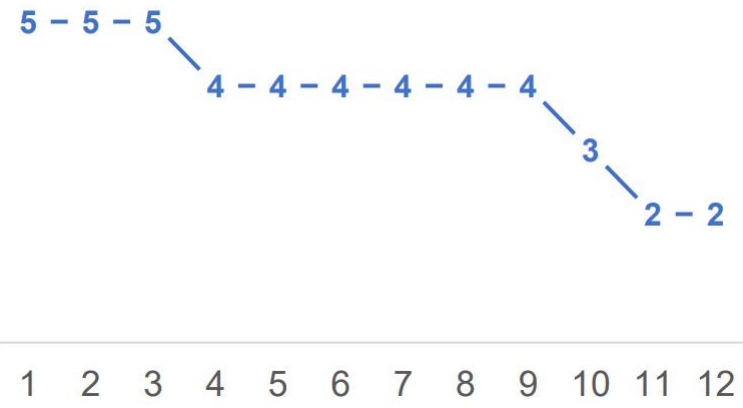
- **二分查找**
  - 找最优决策也变成了$\log n$级别——总复杂度$O(n\log_2^2 n)$

对转移进行优化

- $f[i][j] = \min(\max(f[i-1][w-1], f[i][j-w]) + 1)$
- $f[i][j] \leq \max(f[i-1][w-1], f[i][j-w]) + 1$
- $f[i][j] \leq f[i-1][w-1] + 1$，$f[i][j] < f[i][j-w] + 1$
- $f[i][j] \leq f[i][j-1] + 1$
- 又$f[i][j] \geq f[i][j-1]$
- $f[i][j-1] \leq f[i][j] \leq f[i][j-1] + 1$
- **结论**：
  - 若某一个决策能使得$f[i][j] = f[i][j-1]$，那么$f[i][j] = f[i][j-1]$
  - 否则：$f[i][j] = f[i][j-1] + 1$

- **怎么判断这样一个决策是否存在?**

  - 维护一个指针 `p`，使得 `p` 始终满足：$f[i][p] < f[i][j-1]$ 且 $f[i][p+1] = f[i][j-1]$，即 `p` 指向的是与 $f[i][j-1]$ 相等的数的最前一个的前一个
  - $f[i][j] = \min(\max(f[i-1][w-1], f[i][j-w]) + 1)$
  - $j-w$ 取 `p` 的时候，$w-1$ 为 $j-p-1$
  - 若 $f[i][p] < f[i-1][j-p-1]$ 时，$f[i][j] = f[i][j-1]$
  - 否则 $f[i][j] = f[i][j-1] + 1$



- **结果**：所以我们只需要根据 $f[i][p]$ 和 $f[i-1][j-p-1]$ 的大小就知道转移之后的结果—— $O(1)$ 的转移，总复杂度 $O(n \log_2 n)$

组合数1

交换状态里面的已知量和未知量

- **原状态定义**：$f[i][j]$ 表示 `i` 个蛋在 `j` 层上实验最坏情况下要确定硬度需要的最少次数
- **新状态定义**：$g[i][j]$ 表示 `j` 个蛋试 `i` 次最坏情况下能确定E最高层数，最后我们只需要找到一个最小的 `x` 使得 $g[x][M] > n$成立， `x` 就是答案

- **边界条件**：
  - 无论多少鹰蛋：试一次只能确定一层，即 $g[1][j] = 1$；
  - 只用一个鹰蛋试 `i` 次就可以在 `i` 层中确定E， $g[i][1] = i$
- **状态转移**：
  - 第一次在某一层扔下蛋——
    - 碎了：在之后的 $i-1$ 次里，我们用 $j-1$ 个蛋在下面的楼层确定E，我们会希望下面的层数尽量的多，下面最大层数是 $g[i-1][j-1]$
    - 没碎：在之后的 $i-1$ 次里，用 `j` 个蛋在上面的里面确定E，能确定的最大层数是 $g[i-1][j]$
- **转移方程**： $g[i][j] = g[i-1][j-1] + g[i-1][j] + 1$

算法编号	时间复杂度	空间复杂度	优化方法
算法一	$O(N^3)$	O(N)	
算法二	$O(N^2\log_2 N)$	O(N)	考察问题自身的性质，减少状态总数
算法三	$O(N\log_2^2 N)$	O(N)	研究动态规划方程，找出其中的特性，优化状态转移部分
算法四	$O(N\log_2 N)$	O(N)	进一步挖掘动态规划方程的特性，从而再次降低状态转移部分的时间复杂度
算法五	$O(\sqrt{N})$	$O(\log_2 N)$	建立新的动态规划模型，从另一个角度重新审视问题

小结

- 发掘问题本身性质
- 状态的性质（单调性）
- 转移的性质
- 状态量和求解量其实是相对的

例2

- N个点，连了M条边，问最少需要加几条边使得图中存在点数恰好为k的连通块，数据保证有解
- $N, M \leq 10^5$

- 物品数上限： 10^5
- 背包体积上限: 10^5
- 物品体积和上限： 10^5

例3

- 一共有30001个岛屿，下标为0到30000，有些岛屿藏有宝藏。你从0往下标大的方向跳，第一步跳的距离为d。如果上一步跳的距离为l，这一步就可以跳 $l-1$ 或 l 或 $l+1$（距离必须大于0）。问最多拿到多少宝藏。

- 设 $f[i][j]$ 表示到达第 i 个岛屿，跳的距离为 j 得到的最大宝藏数，那么转移方程：
  - $f[i+j][j]=\max(f[i+j][j],f[i][j]+c[i+j])$
  - $f[i+j+1][j+1]=\max(f[i+j+1][j+1],f[i][j]+c[i+j+1])$
  - $f[i+j-1][j-1]=\max(f[i+j-1][j-1],f[i][j]+c[i+j-1])$

例3

- 由于最大步长是 $1+2+3+...+a+250>30000$。所以第二维最多在第一次的 d 的基础上上下浮动250，于是第二维开500就够了。
- （第二维的值 $x=250$ 表示和 d 相等，$(x-250)+d$ 才是当前跳跃的距离）

例4

- 给出 N 个不同的数，取第 i 个数的代价为 $c[i]$，求取出若干个数使得其最大公约数为1的最小代价

- $f[i]$ 表示 $\gcd$ 为 i 的最小代价
- $f[\gcd(a,b)] = \min(f[\gcd(a,b)], f[a] + \text{cost}(b))$

```
for(int i = 0; i < n; i++)
{
    map<int, int>::iterator it;
    for(it = dp.begin(); it != dp.end(); it++)
    {
        int t = gcd(l[i], it->first);
        if(dp.count(t) != 0)
            dp[t] = min(dp[t], it->second + c[i]);
        else
            dp[t] = it->second + c[i];
    }
}
```

例5

- 吉吉国王现在想选出 m 组人来组成部队，现在他有 n 个人可以选择，每个人都有一个战力值 a_i。但是吉吉国王非常笨，因此他只能选择连续的一段区间 $[l,r]$ 中的人来组成一个部队，所以选出的这 m 组人在这1到 n 个人中没有重合的部分。吉吉国王希望这 m 组人的战斗力之和最大，你需要告诉吉吉国王最大的战斗力之和是多少。
- $1 \leq n \leq 1e5$
- $1 \leq m \leq 100$
- $-1e9 \leq a_i \leq 1e9$

例5

- $f[i][j]$ 表示前 i 个人(第 i 个人必须选)已经选了 j 个区间的最大战斗力之和
  - $f[i][j]=\max(f[i-1][j]+a[i],f[k][j-1]+a[i])$

    $=\max(f[i-1][j],f[k][j-1])+a[i] \quad 1\le k\le i-1$

例6

- 给定一排长度为 n 的瓷砖，每块瓷砖都有一个数字 a_i。现在需要对这些瓷砖进行涂色，每次涂色都可以选一段之前没有涂过的区间，涂色的代价就是这段区间种不同的数字的个数的平方，比如对于 $(1,3,3,4,2)$，选择对区间 $[2,5]$ 涂色，因为 2 到 5 有 2,3,4 三种数字，那么代价就是 9。现在需要求出把所有瓷砖都涂上颜色的最小代价

- $f[i]$ 表示涂完前 i 个所化的最小代价，显然有 $f[i]=\min\{dp[j]+\text{num}(j+1,i)^2\}$，
- 其中 $1\le j<i$, $\text{num}(j+1,i)$ 表示区间 $[j+1,i]$ 的颜色个数。

例6

- $\mathrm{num}(j+1,i)$ 没有超过 $\sqrt{n}$ 的必要——一个一个涂，那么需要花费 n，最优方案肯定不会比他大

- 如果当前计算到了第 i 位，那么即可以将 a_i 加进来，而之前如果有和 a_i 值相等的点就可以删掉了（往前枚举 j 的时候不用再遍历这个点（用指针跳过他），因为他不会再影响 j 到 i 之间不同的数的个数）保证对于任意一个 i，往前遍历 j 的时候，每种颜色只会走一次，复杂度 $O(n \times \mathrm{color})$。

- color 很多怎么办?

- 如果当前遍历到的颜色个数大于了 $\sqrt{n}$ 个那么，这个循环就可以结束了，所以最多 $O(n^2)$
- $1 \leq n \leq 5e4$

例7

- 设有 n 棵树，每个篮子的容量为 k，第 i 棵树上的红果和蓝果的数分别为 a_i 和 b_i。现在又很多个篮子，每个篮子装同一个颜色的果子，但是允许不同树上的果实，要么只能装同一个树上，颜色允许不同的果实。现在告诉你篮子的大小 k，问你最多可以装满多少个篮子。
- $1 \leq n, k \leq 500$
- $0 \leq a_i, b_i \leq 10^9$

- $f[i][j][k]$ 来表示前 i 棵树，留下了 j 个红果子，k 个蓝果子可以装满的最大篮子数。
- $O(nk^4)$

例7

- $f[i][j]$ 表示前 i 棵树装完后，剩下 j 个红果子，最多能填满的篮子的数量。
- 剩余蓝果子数量 $=$ 前i棵树果子总数 $-\ k \times f[i][j] - j$

- 从 $f[i][j]$ 推出去——枚举第 $i+1$ 棵树红果子的分配：s 个红的和 $k-s$ 个蓝的装一箱，剩下的 $a[i+1]-s$ 个红果子和之前剩下的 j 个红果子自己装满 $\frac{(a[i+1]-s+j)}{k}$ 个箱子，剩下 $(a[i+1]-s+j)\%k$ 个，$b[i+1]-k+s$ 个蓝果子和之前剩下的 $\mathrm{sum}[i]-k\times f[i][j]-j$ 个蓝果子装满 $\frac{(b[i+1]-k+s+\mathrm{sum}[i]-k\times f[i][j]-j)}{k}$ 个箱子，剩下 $(b[i+1]-k+s+\mathrm{sum}[i]-k\times f[i][j]-j)\%k$ 个

$$f[i+1][(a[i+1]-s+j)\%k] = \max(f[i][j]+1+\frac{(a[i+1]-s+j)}{k}+\frac{(b[i+1]-k+s+\mathrm{sum}[i]-k\times f[i][j]-j)}{k})$$

时间复杂度 $O(nk^2)$

谢 谢 大 家

董国梁

