思考题讲解

思考题 1：小鸡啄米

（1）

- 输入：一个 m*n 的矩阵 A，矩阵的每一个元素都是一个非负整数，代表该位置的米数；
- 有一只小鸡从左上角 A[1][1]出发，每次往右或者往下走一步，走到右下角 A[m][n]停止；小鸡会将途中经过的所有米粒都吃掉；
- 设计一个算法，计算出小鸡应该如何走保证吃到的米粒最多。
- 分析算法时间复杂度。

思路：动态规划。
动态规划的设计技巧：阶段的划分、状态的表示、存储的设计
以斐波那契数列(Fibonacci sequence)为例，递归的方法定义：F(0)=0，F(1)=1, F(n)=F(n-1)+F(n-2)（n>=2，n∈N*）
阶段：F(i)的值；状态的表示：F(n)=F(n-1)+F(n-2)

回到问题上，我们定义阶段 F[i][j]为该小鸡从 A[1][1]走到 A[i][j]能吃掉的最多米粒数，那么状态的表示：F[i][j] = max(F[i-1][j], F[i][j-1]) + A[i][j];
时间复杂度 O(m*n)

（2）

- 假设有两只小鸡从左上角 A[1][1]出发，都要走到右下角 A[m][n]，都只能向下或者向右移动；
- 两只小鸡移动的先后顺序不做限制，但是先到某个位置的小鸡会将该位置米粒吃完；
- 问如何安排两只小鸡的移动顺序与轨迹，使得两只小鸡吃到的米粒数之和最多？

用相同的思路：

定义阶段 F(i1,j1,i2,j2)表示第一只小鸡走到 A[i1][j1],第 2 只小鸡走到 A[i2][j2]时，吃的最多的米粒数，此时空间为 O（m^2*n^2）。考虑到若在 A[i][j]发生了冲突，则两只小鸡都走了 i-1 + j-1 步，所以可以把这个问题看成两个小鸡每阶段都往前走 1 步，走了 m-1+n-1 步以后，求最多的米粒之和。

那我们就可以得到等式 i1 + j1 = i2 + j2。将复杂度从 4 维降到 3 维。

重新定义阶段 F(i1,i2,x)表示第一只小鸡走到第 i1 行，第二只小鸡走到第 i2 行，并且它们走了 x 步以后能吃到的最多米粒数，这样易得 j1 = x – i1 + 2, j2 = x – i2 + 2。

此时状态的表示即：F(i1,i2,x) = max(x-1 时四种状态走到该状态的米粒和)，注意要判断两只小鸡新走的这一步是否重合了来决定加多少米粒数。

时间复杂度 O(m*n*(m+n))

（3）

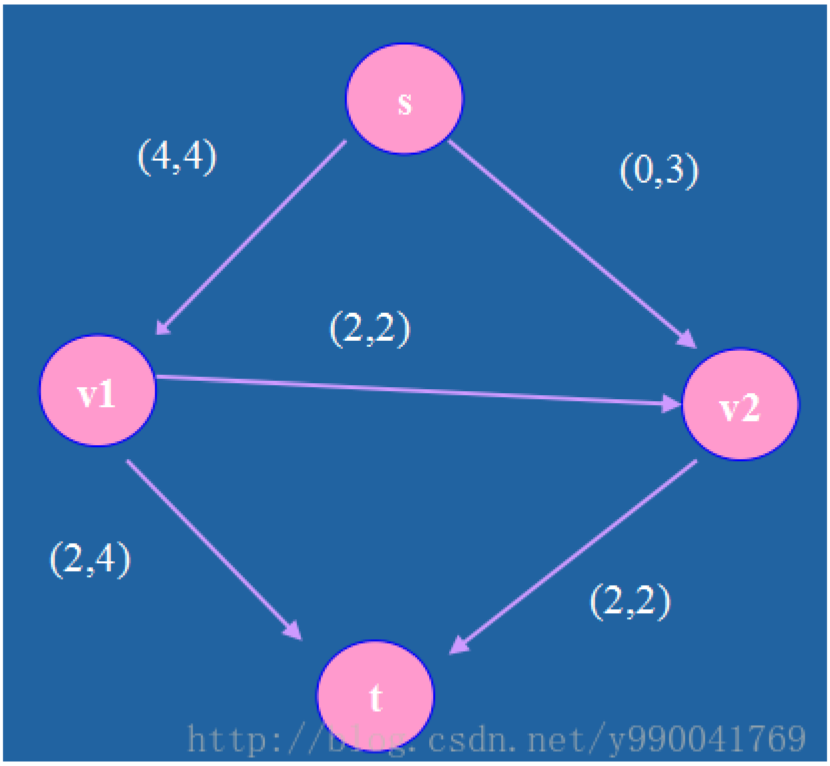
- 如果有 k 只小鸡呢？

如果还用动态规划，同理，可以构建一个 K+1 维的阶段，即走了 x 步，第 h 只小鸡在（ih）行的最大米粒和数来解。但这里可能在实现上有空间复杂度高，判断条件复杂的情况。

提出另一种解法：

最大流方法

介绍：

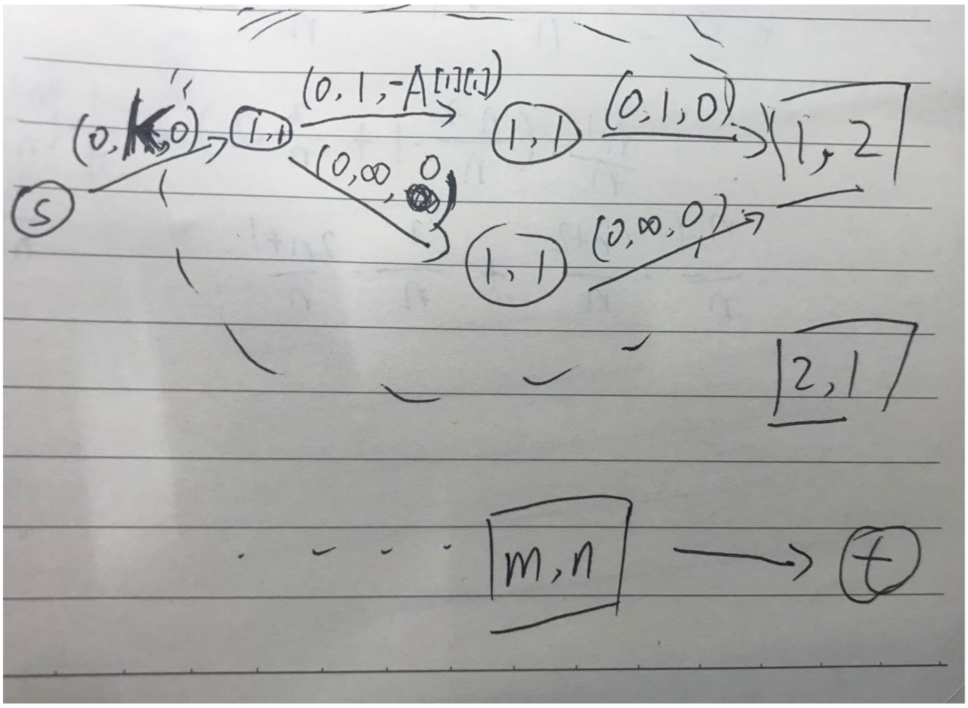


每个边两个值（f,c），f 代表当前的流量，c 代表这条边的最大容量。

最大流求从 s 出发到 t 能达到的最大流量

最小费用最大流：达到最大流的前提下，每个边还有个费用 Cost，使得 Cost 的总和最小

本题求从 1,1 出发到 m,n 的最大米粒和即可转化为最小费用最大流问题，但需要构造一个图来用边表示这道题的条件：



按照图中所示构造图以后，求解最小费用最大流问题即可

思考题 2

Majority Problem(多数问题)

定义：给定一个大小为 n 的数组，找出其中出现次数超过 n/2 的元素

思考题：

设计 O(n)时间，O(1)空间的算法解决多数问题

解法：用一个空间 A 表示记下的元素，用另一个空间 C 表示该元素当前的 count。
遍历所有元素 x：
（1） 当 A 为空时，A = x, C = 1;
（2） 当 A == x 时， C++;
（3） 当 A != x 时， C--; 如果 C == 0, 则 A = NULL
这样结束以后，如果存在一个次数超过 n/2 的元素，那么 A 的值就是该元素。

证明：对每个元素来看，该算法使得其真正的频数减少至多为 n/2，因为 n 个元素最多发生 n/2 次减法。所以如果有元素 x 的频数大于 n/2，则最后它的频数一定大于 0，所以一定在 A 中。

思考题 3

1：投 b 个球到 b 个盒子，最大的盒子里包含多少个球？O(log b)
2：投 b 个球到 b/log b 个盒子里，最大的盒子里包含多少个球？O(log b)
3：投 b 个球到 b 个盒子，每个球随机选两个盒子，并放入其中球 数较少的盒子里。最短的盒子里包含多少个球？O(loglog b), Power of 2-choices

1.
参考 Chernoff bound： https://en.wikipedia.org/wiki/Chernoff_bound
定义随机变量 Xij 表示第 i 个球投到第 j 个盒子的指示器变量，则 E[Xij] = 1 / b
Xj = sum(X1j,…Xbj)，则根据 Chernoff bound:

$$\Pr(X \geq (1+\delta)\mu) \leq e^{-\frac{\delta\mu}{3}}, \qquad 1 < \delta,$$

此时 μ = 1，
将 6*logb 代入 δ，可得 Pr（X >= 6 *logb + 1） <= 1 / b^2
则任意有一个 Xj >= 6 * logb + 1 的概率是 b * 1 / b^2 = 1 / b
则当 b -> 正无穷时，存在 c > 6, Pr(最大的盒子超过 c*logb) -> 0

2.
同理：
E[Xij] = logb / b;
此时 μ = logb，
将 6 代入 δ，得 Pr(X >= 7 * logb) <= 1 / b^2，同样得证

3.
题目错了，应该是最大的盒子里包含多少球。
设 Ni 表示所有在第 i 层的球的个数的比例，则 Ni <= 1 / i
Ni = (Ni-1)^2，则 LogNi = 2logNi-1
又因为知道初始值 N4 <= 1 / 4，则 logNi< = -2^(i-3)
则当 i = loglogn + 3 时，Ni <= 1 / logn，则当 n —> 正无穷时，Ni->0.

习题：
9.3-6 对一个包含 n 个元素的集合来说，k 分位数是指能把有序集合分成 k 个等大小集合的第 $k-1$ 个顺序统计量。给出一个能找出某一集合的 k 分位数的 $O(n\lg k)$ 时间的算法。
思路：
每个子集合的元素个数为 t = n / k，A[j]是数组 A 中下标为 j 的元素，A(j)是数组是第 j 大的元素
则所求的 k 分位数是指 A(t)，A(2t)，A(3t)，……，A((k-1)t)
常规思路是按照顺序求第 t 小，第 2t 小。。。。的元素
按顺序依次求这 k-1 个数的运行时(k-1)*n
要使运行时间为 O(nlgk)，改进方法是不要依次寻找这 k-1 个数，而是借用二分的方法来找。
先找第 k/2 个分位数，再以这个分位数为主元把数组分为两段，分别对这两段来找分位

数，这样的时间复杂度降为了 $n + 2 * n/2 + 4 * n / 4 + \ldots + (c*\log K * n /(c* \log K)) = O(n\log K)$