优化，再优化！

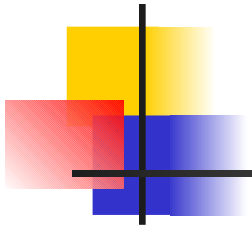
——从《鹰蛋》一题浅析对动态规划算法的优化

安徽省芜湖市第一中学 朱晨光

引言

在当今的信息学竞赛中，动态规划可以说是一种十分常用的算法。它以其高效性受到大家的青睐。然而，动态规划算法有时也会遇到时间复杂度过高的问题。因此，要想真正用好用活动态规划，对于它的优化方法也是一定要掌握的。

本文将就《鹰蛋》这道题目做较为深入的分析，并从中探讨优化动态规划的本质思想与一般方法。

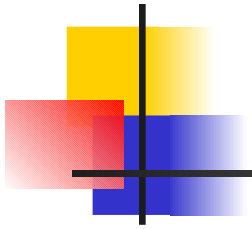


问题

有一堆共 M 个鹰蛋，一位教授想研究这些鹰蛋的坚硬度 E 。他是通过不断从一幢 N 层的楼上向下扔鹰蛋来确定 E 的。

当鹰蛋从第 E 层楼及以下楼层落下时是不会碎的，但从第（ E+1 ）层楼及以上楼层向下落时会摔碎。

如果鹰蛋未摔碎，还可以继续使用；但如果鹰蛋全碎了却仍未确定 E ，这显然是一个失败的实验。教授希望实验是成功的。



问题

例如：若鹰蛋从第 1 层楼落下即摔碎，E=0 ；若鹰蛋从第 N 层楼落下仍未碎， E=N 。

这里假设所有的鹰蛋都具有相同的坚硬度。给定鹰蛋个数 M 与楼层数 N (M,N<=1000) ， 求最坏情况下确定 E 所需要的最少次数。

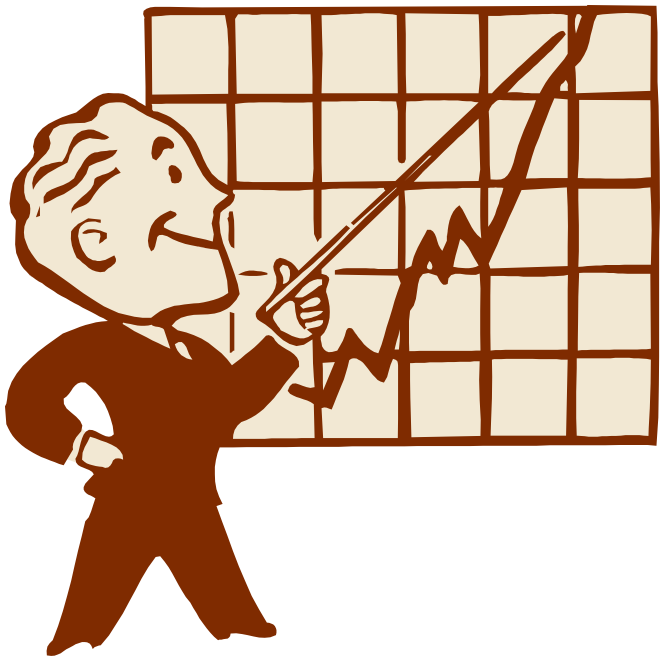
样例： M=1 ， N=10

ANS=10

（解释：只能将这个鹰蛋从下往上依次摔）

算法一

由于是求最优值，我们自然想到了使用动态规划！



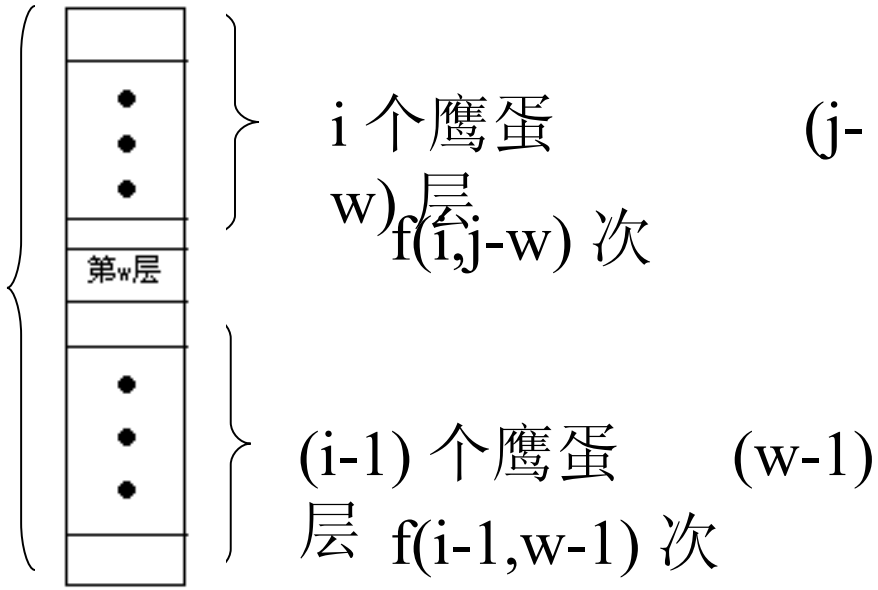
算法一

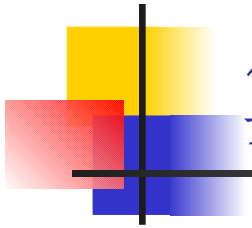
状态定义：

f(i,j): 用 i 个蛋在 j 层楼上最坏情况下确定 E 所需要的最少次数。

状态转移：

i 个鹰蛋
j 层





算法一

状态定义：

f(i,j): 用 i 个蛋在 j 层楼上最坏情况下确定 E 所需要的最少次数。

状态转移：

$$f(i,j)=\min\{\max\{f(i-1,w-1),f(i,j-w)\}+1 \mid 1<=w<=j\}$$

算法一

显然，这个算法的时间复杂度为 $O(N^3)$

太高了

!

如何才能降低它的时间复杂度呢？

算法二

经过观察，我们发现这题很类似于二分查找，只不过是对鹰蛋的个数有限制。

若是对鹰蛋的个数没有限制呢？

这题就变成求二分查找在最坏情况下的比较次数

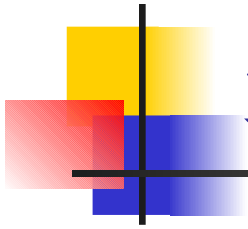
!

答案即为 $\lceil \log_2(n+1) \rceil$

算法二

因此，当 M>=$\lceil \log_2(n+1) \rceil$ 时，直接输出 $\lceil \log_2(n+1) \rceil$ 即可 .

算法的时间复杂度立即降为 $O(N^2 \log_2 N)$



算法二

这里，我们是通过减少状态总数而得到了优化的空间，从而大大提高了算法效率。这也是优化动态规划算法的一种常用方法。

然而优化还远未结束！

算法三

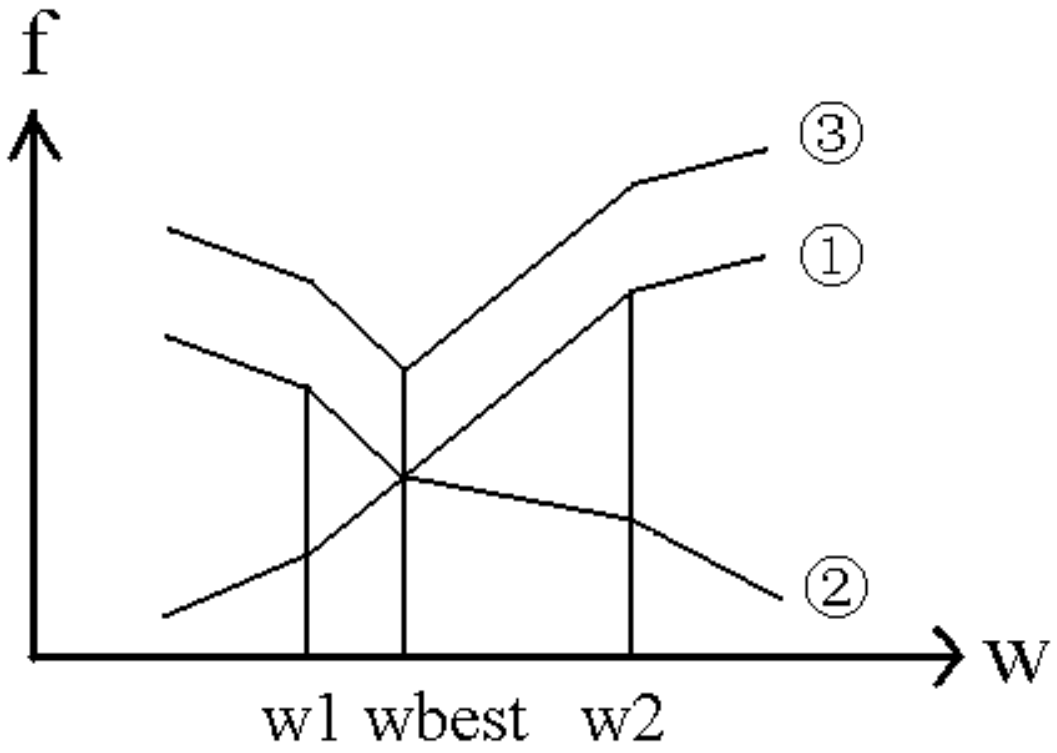
经观察发现，动态规划函数 f(i,j) 具有如下单调性：

$$f(i,j)>=f(i,j-1)\ (j>=1)$$

这条性质可以用数学归纳法进行证明，这里就从略了。

那么， f(i,j) 的单调性有什么作用呢？

算法三



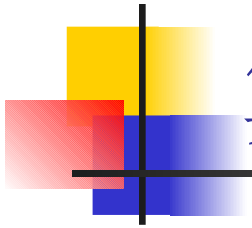
（如图，令①为 f(i-1,w-1) 的图象，②为 f(i,j-w) 的图象 ,③ 即为 max{f(i-1,w-1),f(i,j-w)}+1 的图象）

算法三

这样，我们就成功地将状态转移的时间复杂度降为 $O(\log_2 N)$ ，算法的时间复杂度也随之降为 $O(N(\log_2 N)^2)$.

在对算法三进行研究之后，我们会萌生一个想法：既然现在 f(i, j) 都需要求出，要想找到更高效的算法就只能从状态转移入手，因为这一步是 $O(\log_2 N)$ ，仍然不够理想。

因此，算法四将以状态转移为切入点，进一步探究优化的空间。



算法四

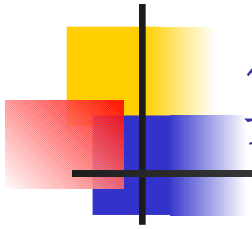
通过进一步挖掘状态转移方程，我们得到如下不等式：

$$f(i,j-1)<=f(i,j)<=f(i,j-1)+1 \quad (j>=1)$$

根据这个不等式，我们可以得到如下推理：

若存在一个决策 w 使得 f(i,j)=f(i,j-1)，则 f(i,j)=f(i,j-1)

若所有决策 w 均不能使 f(i,j)=f(i,j-1)，则 f(i,j)=f(i,j-1)+1



算法四

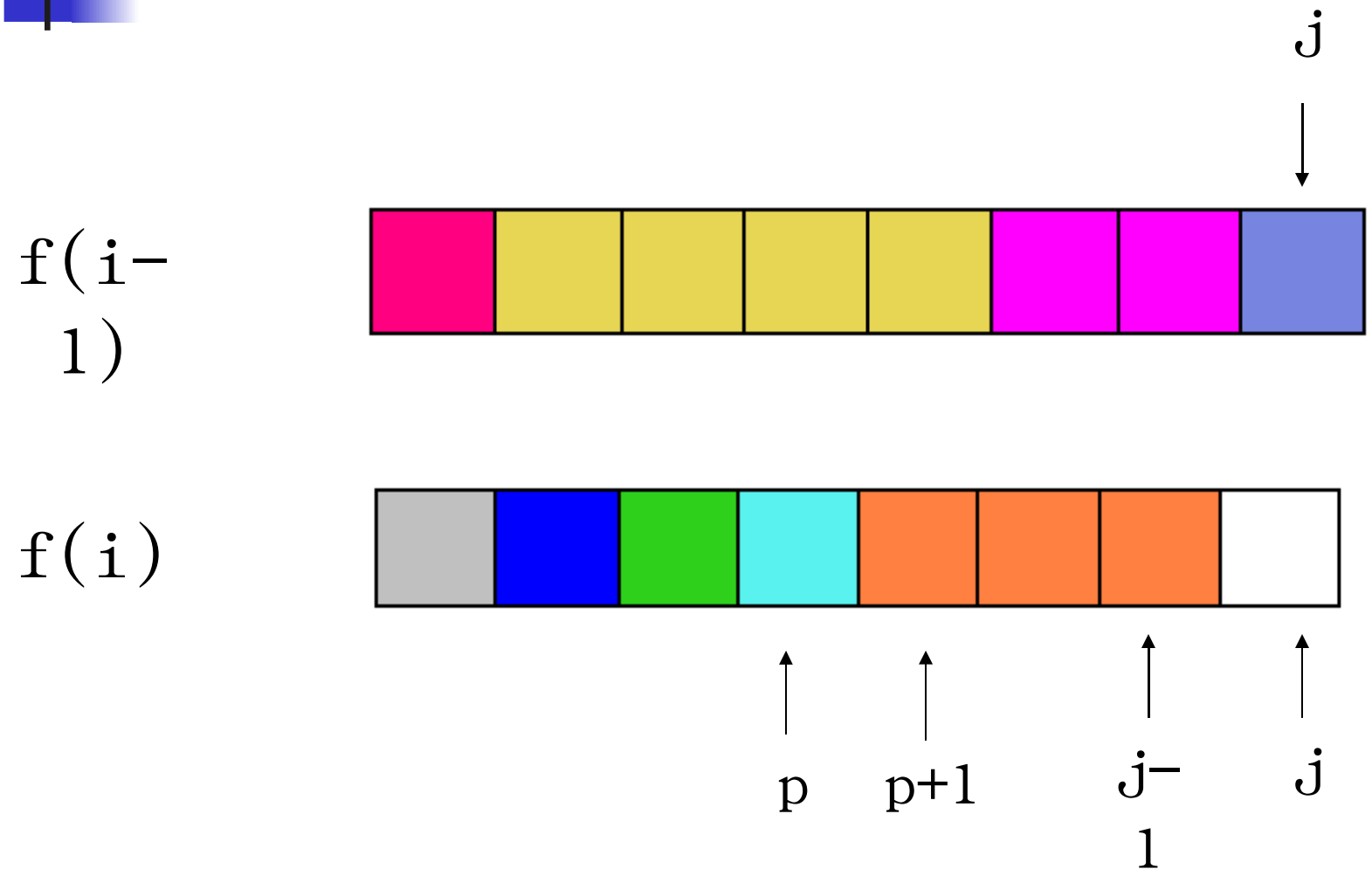
这里，我们设一指针 p ，并使 p 时刻满足：

$$f(i,p)=f(i,j-1)-1 \quad 且 \quad f(i,p+1)=f(i,j-1)$$

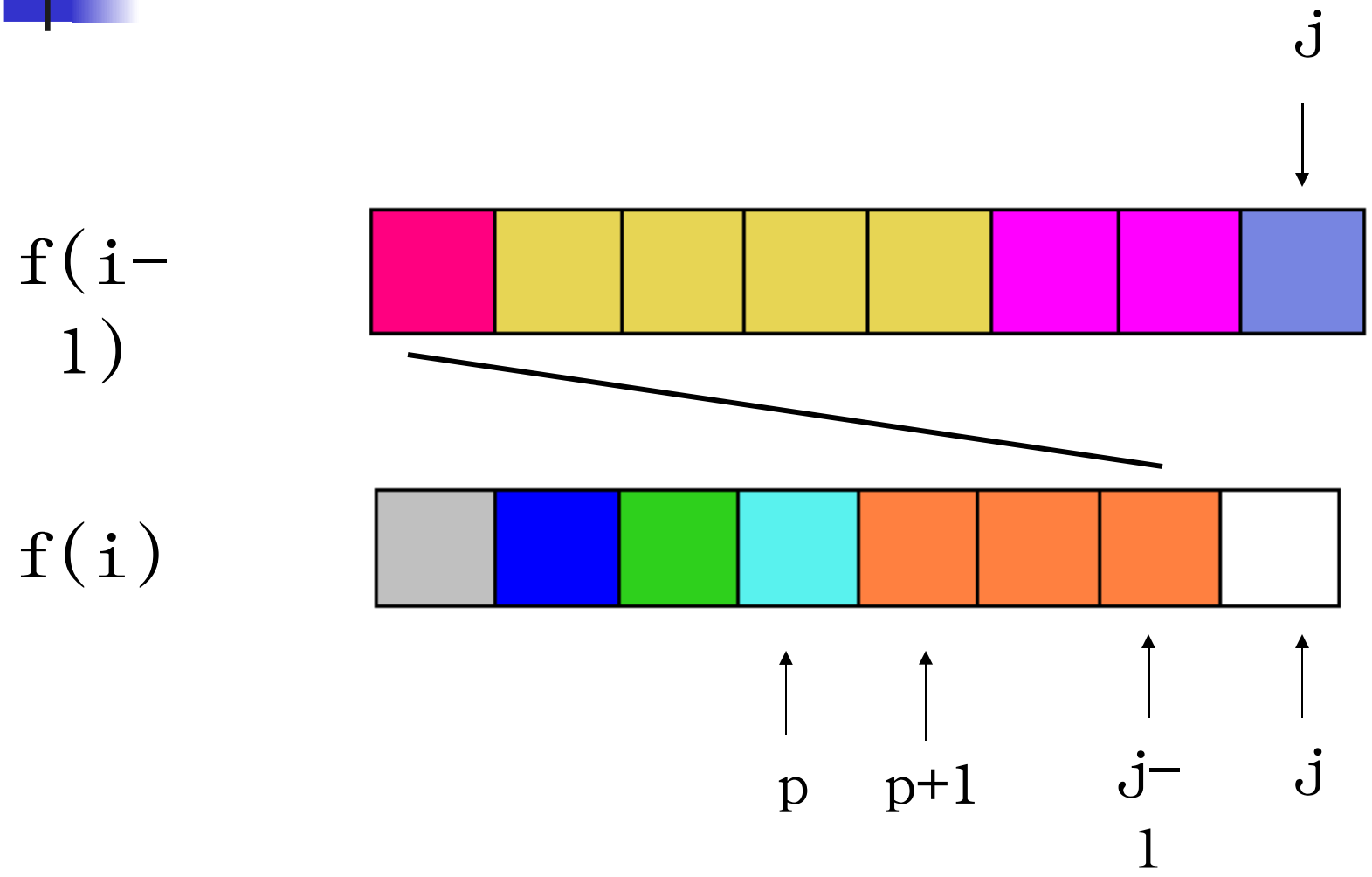
由状态转移方程可知，决策时 f(i,p) 所对应的函数值是 f(i-1,j-p-1).

下面，我们将证明只需通过判断 f(i,p) 与 f(i-1,j-p-1) 的大小关系便可以决定 f(i,j) 的取值。

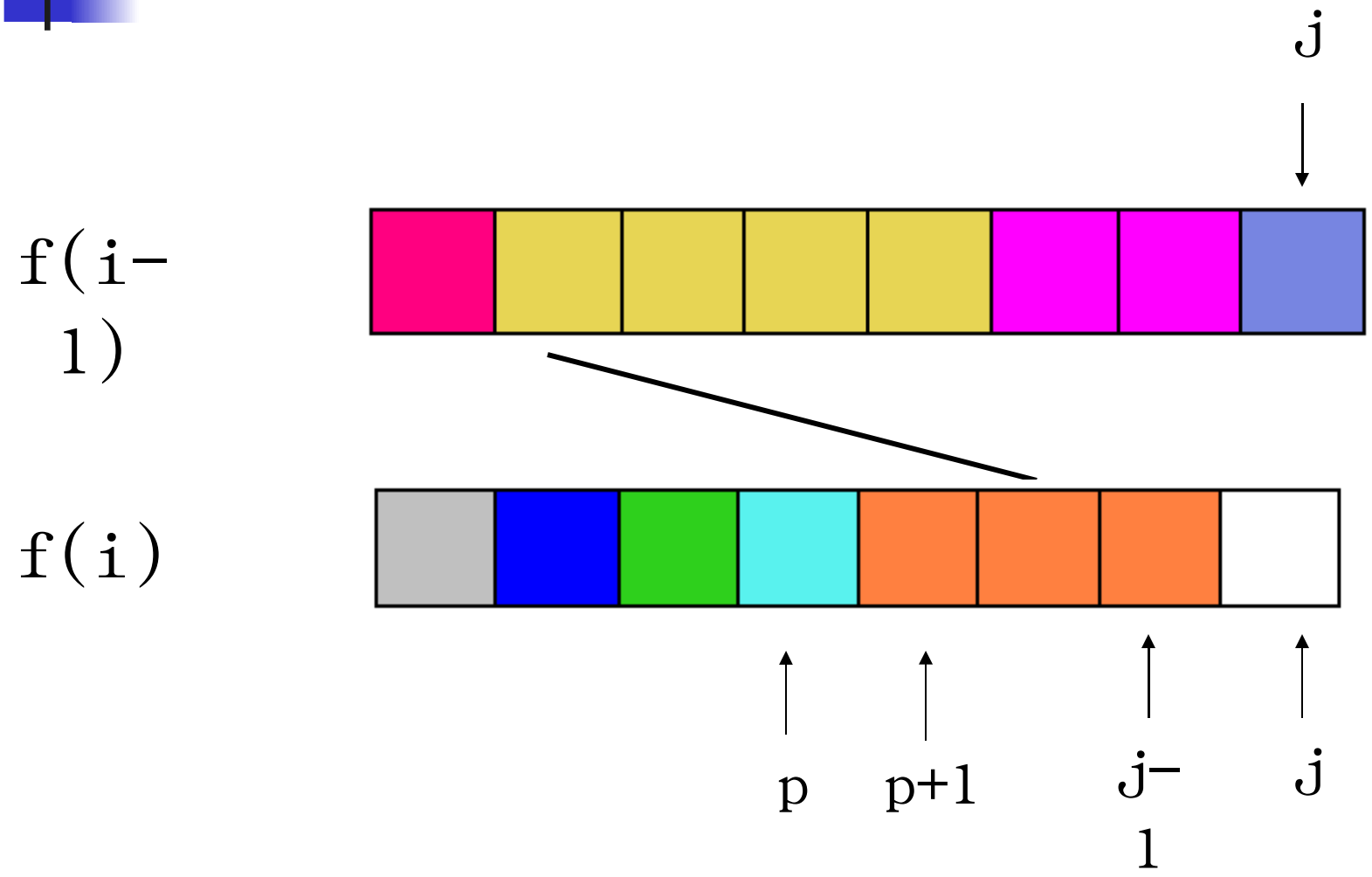
算法四



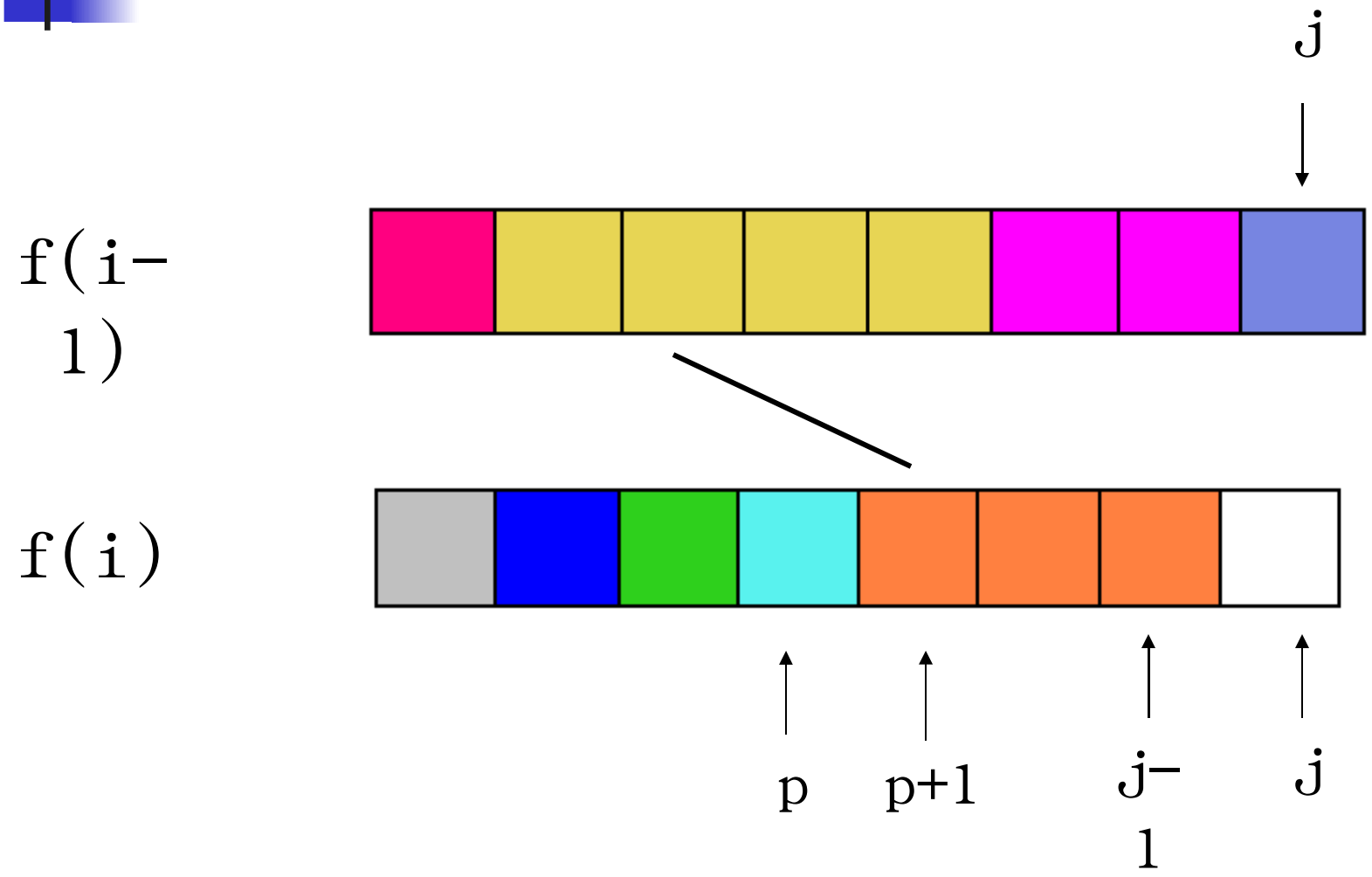
算法四



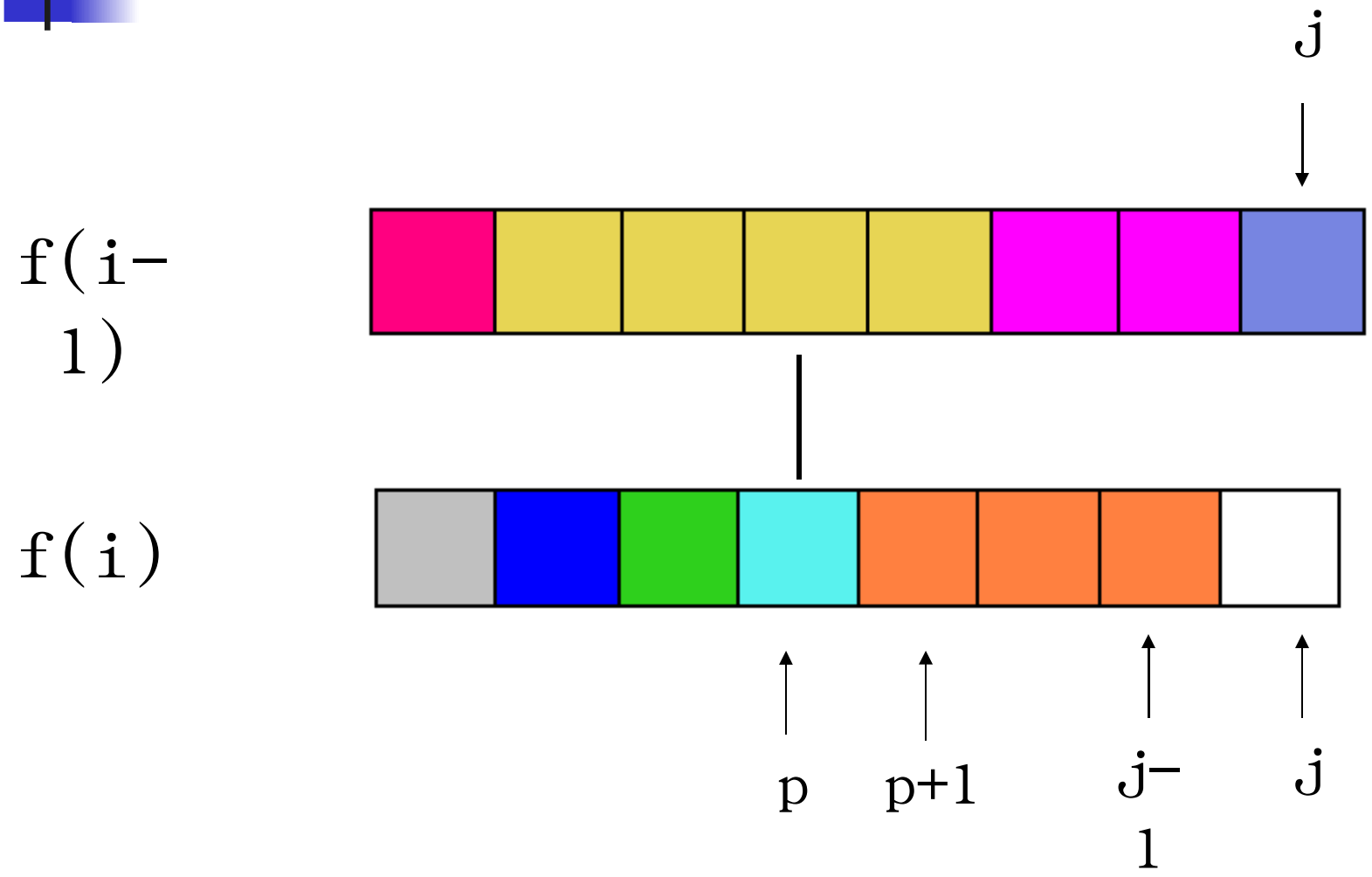
算法四



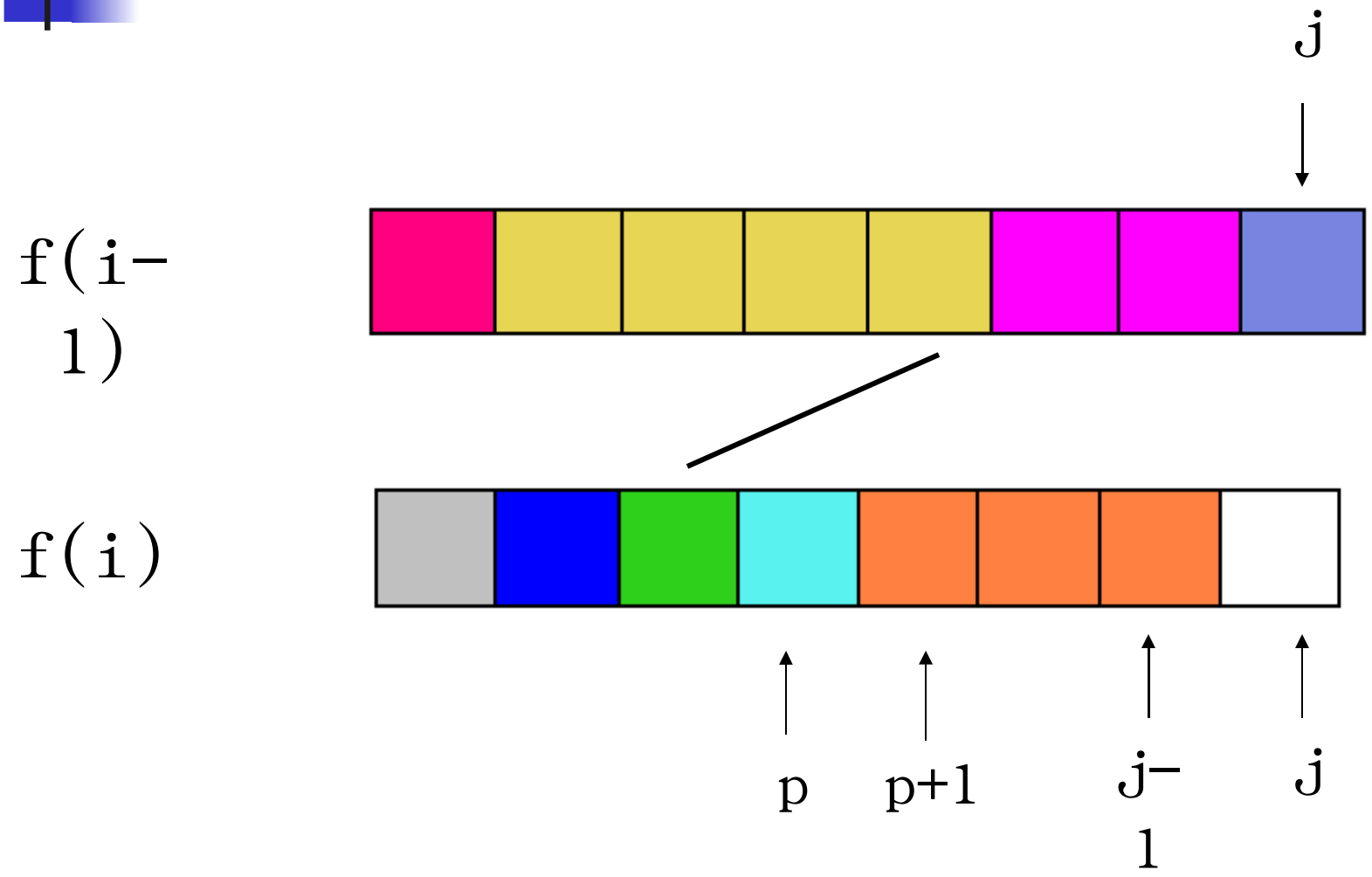
算法四



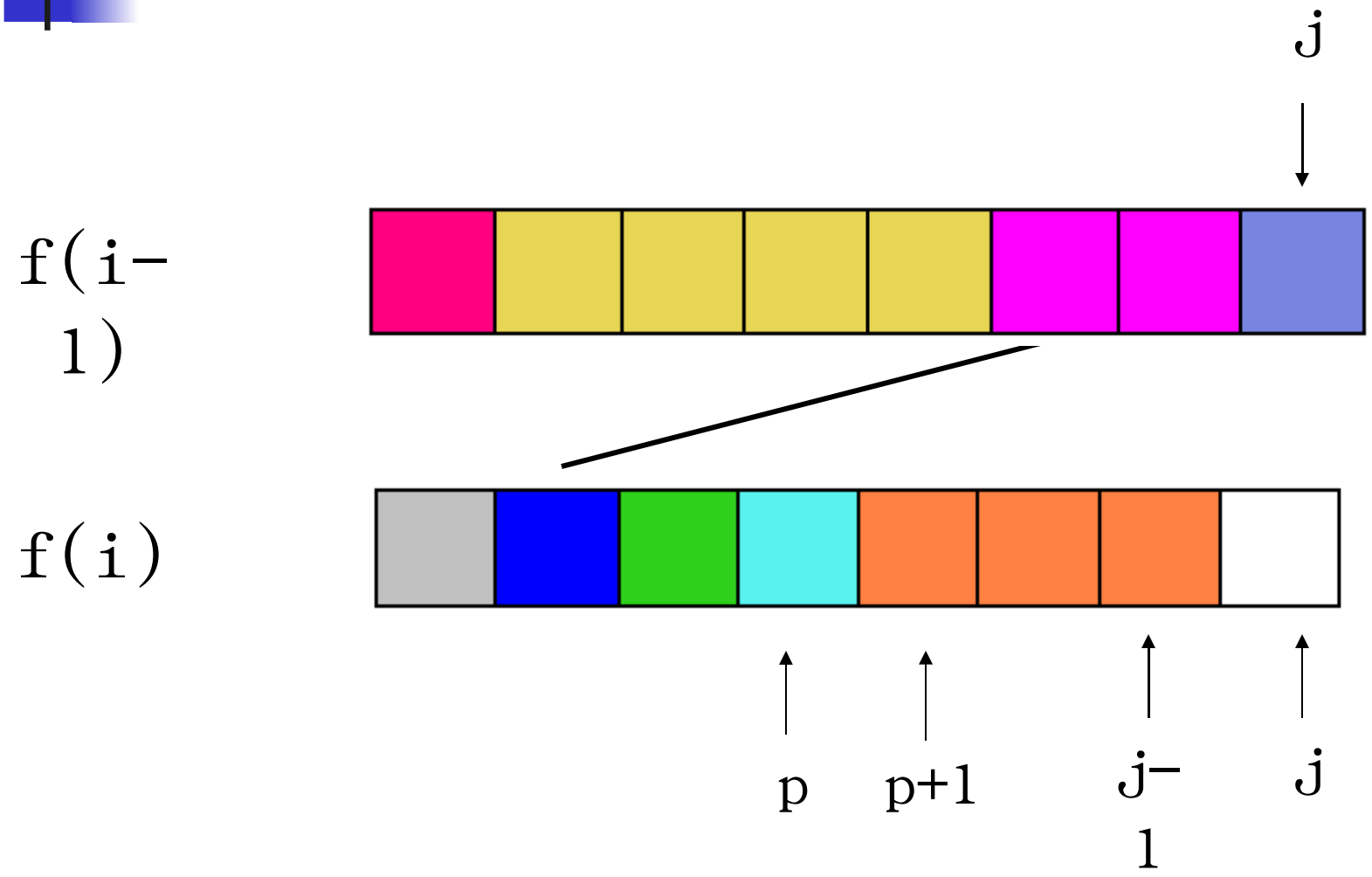
算法四



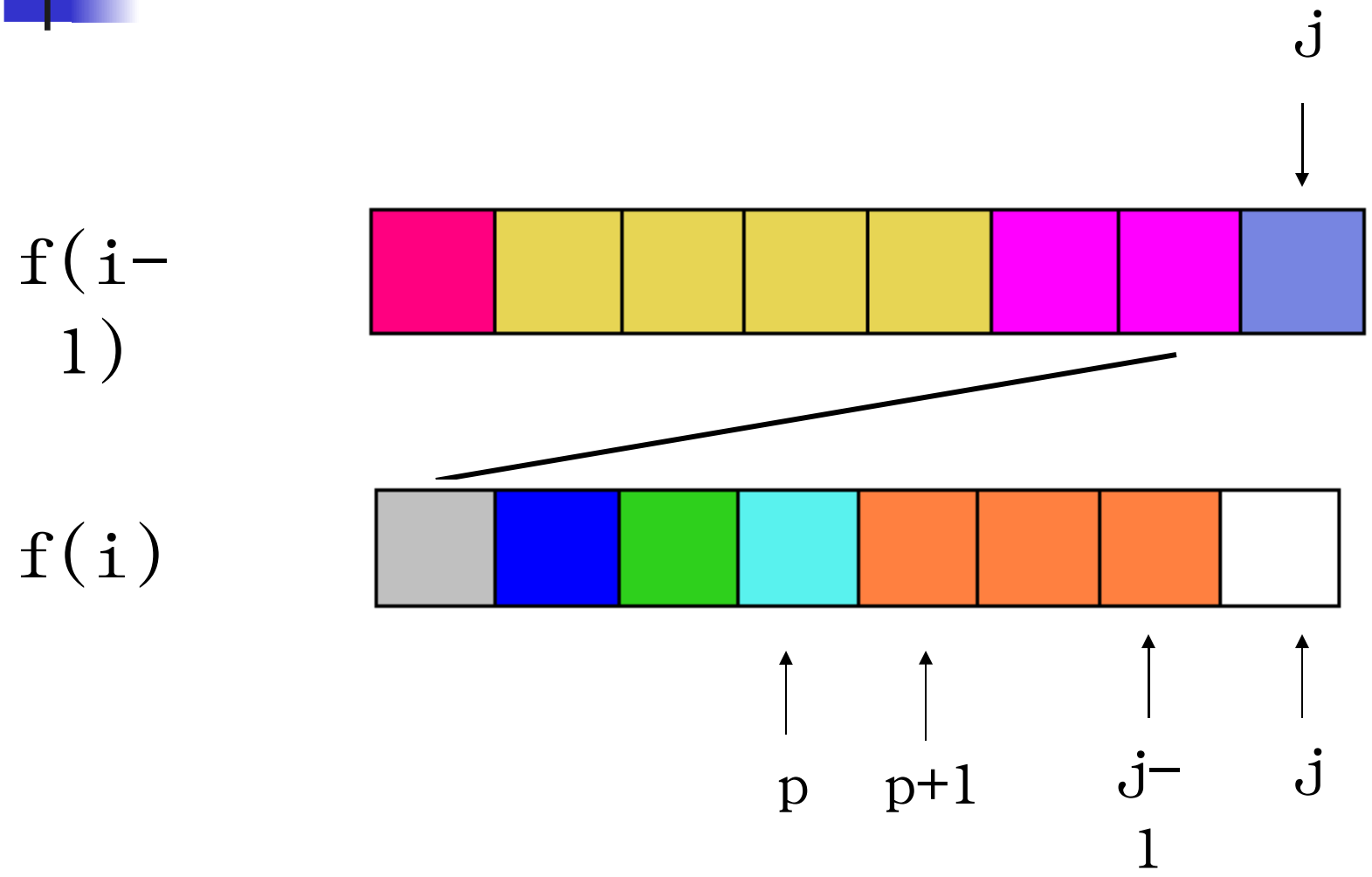
算法四



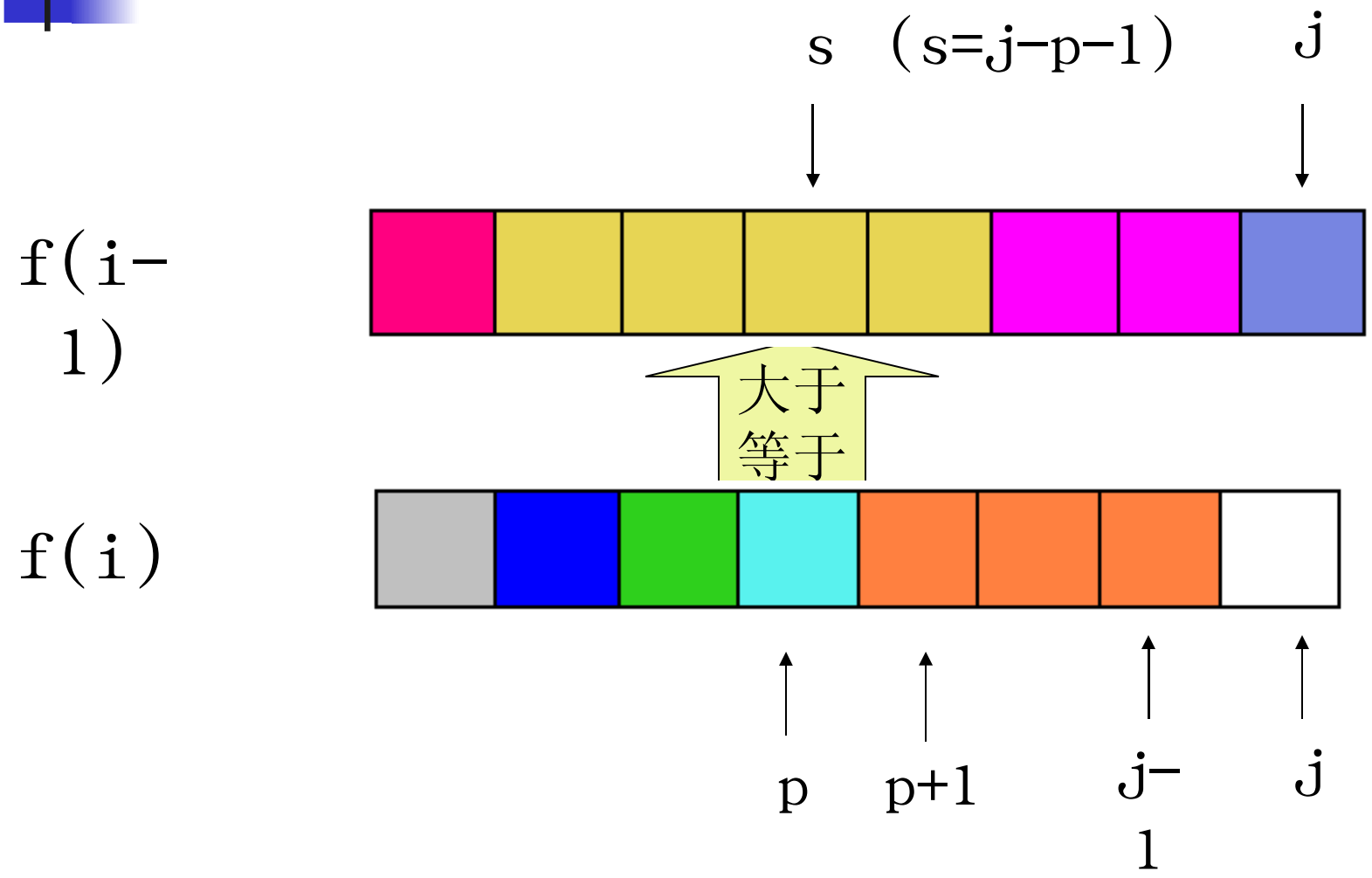
算法四



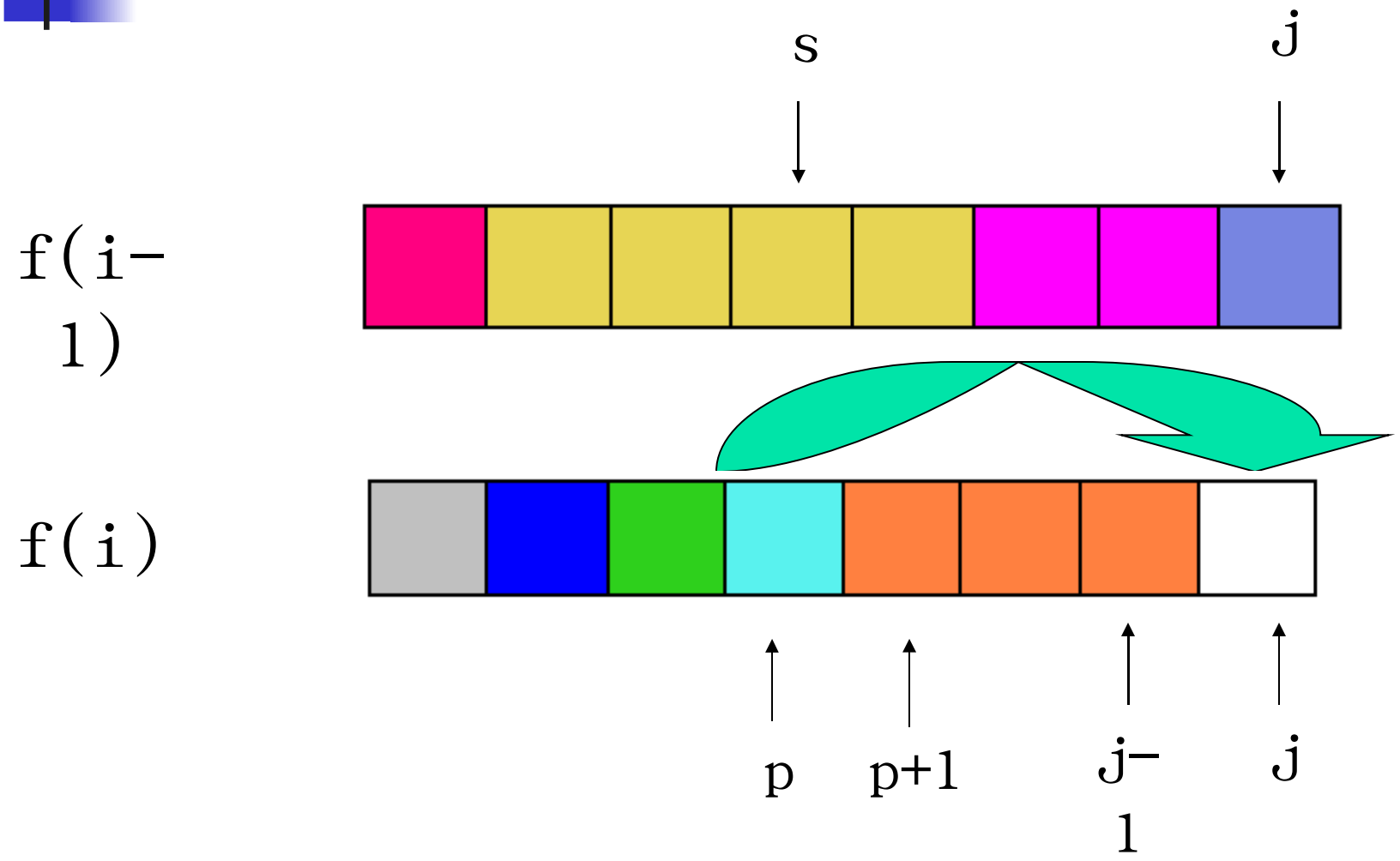
算法四



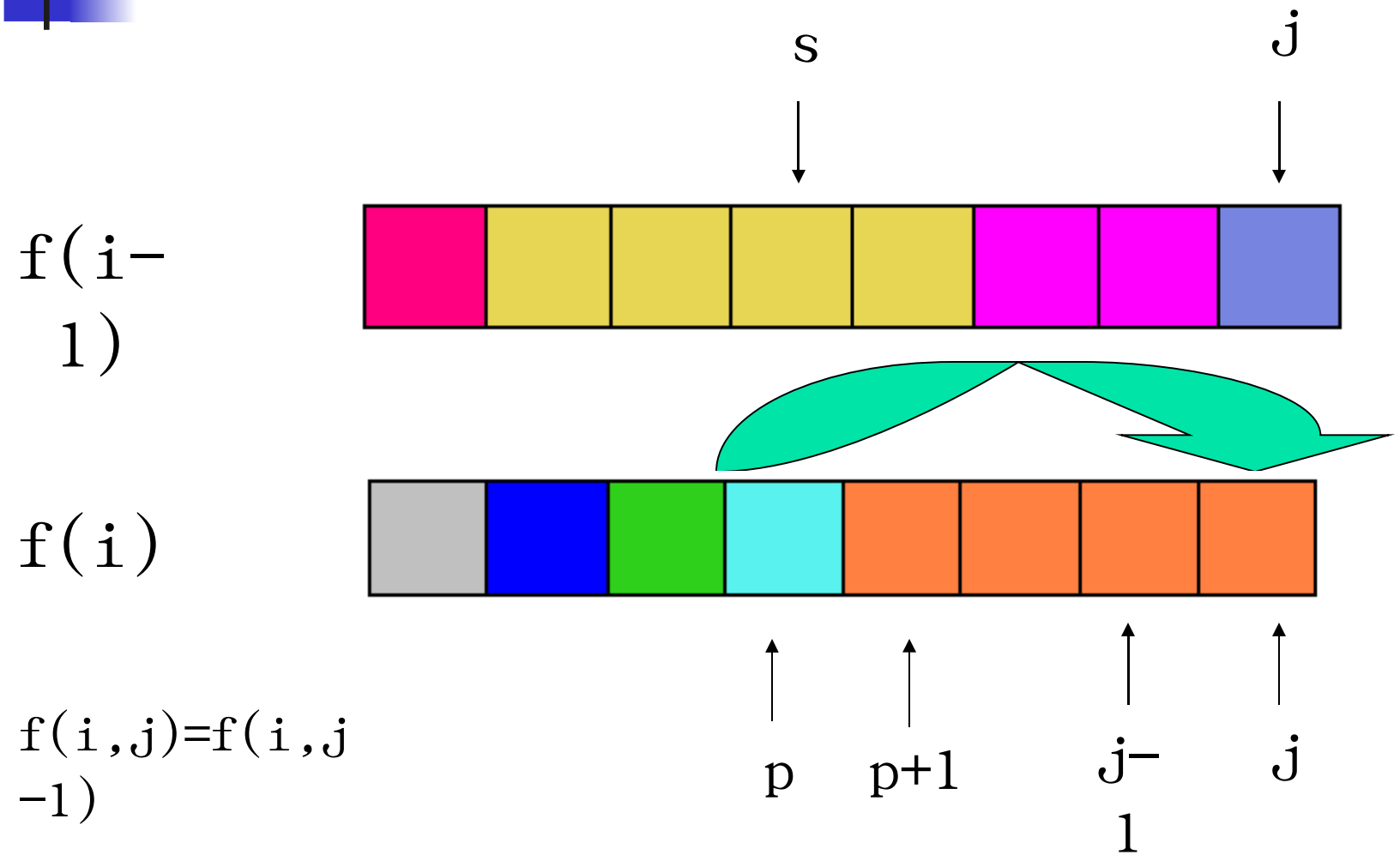
算法四



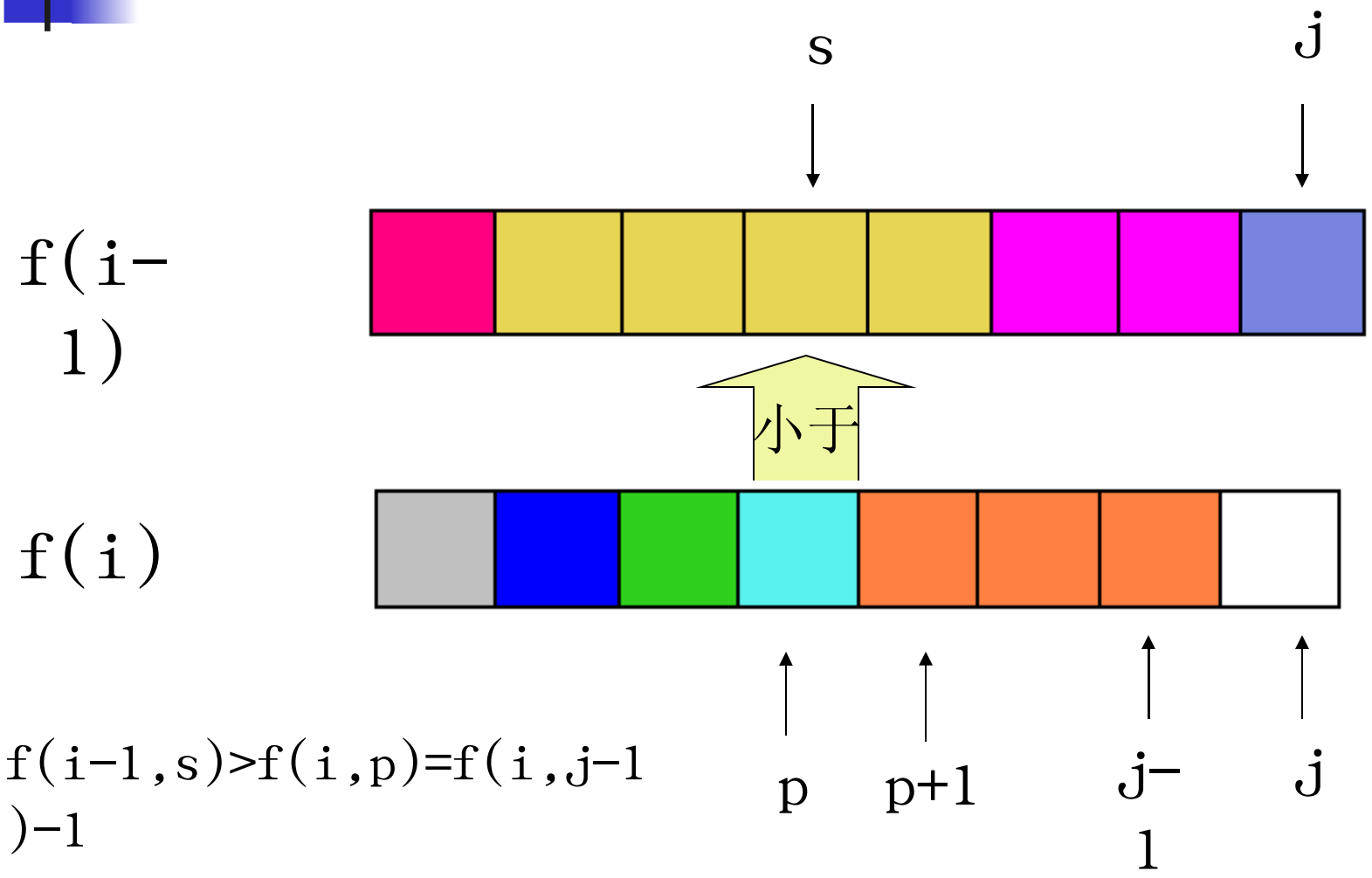
算法四



算法四



算法四



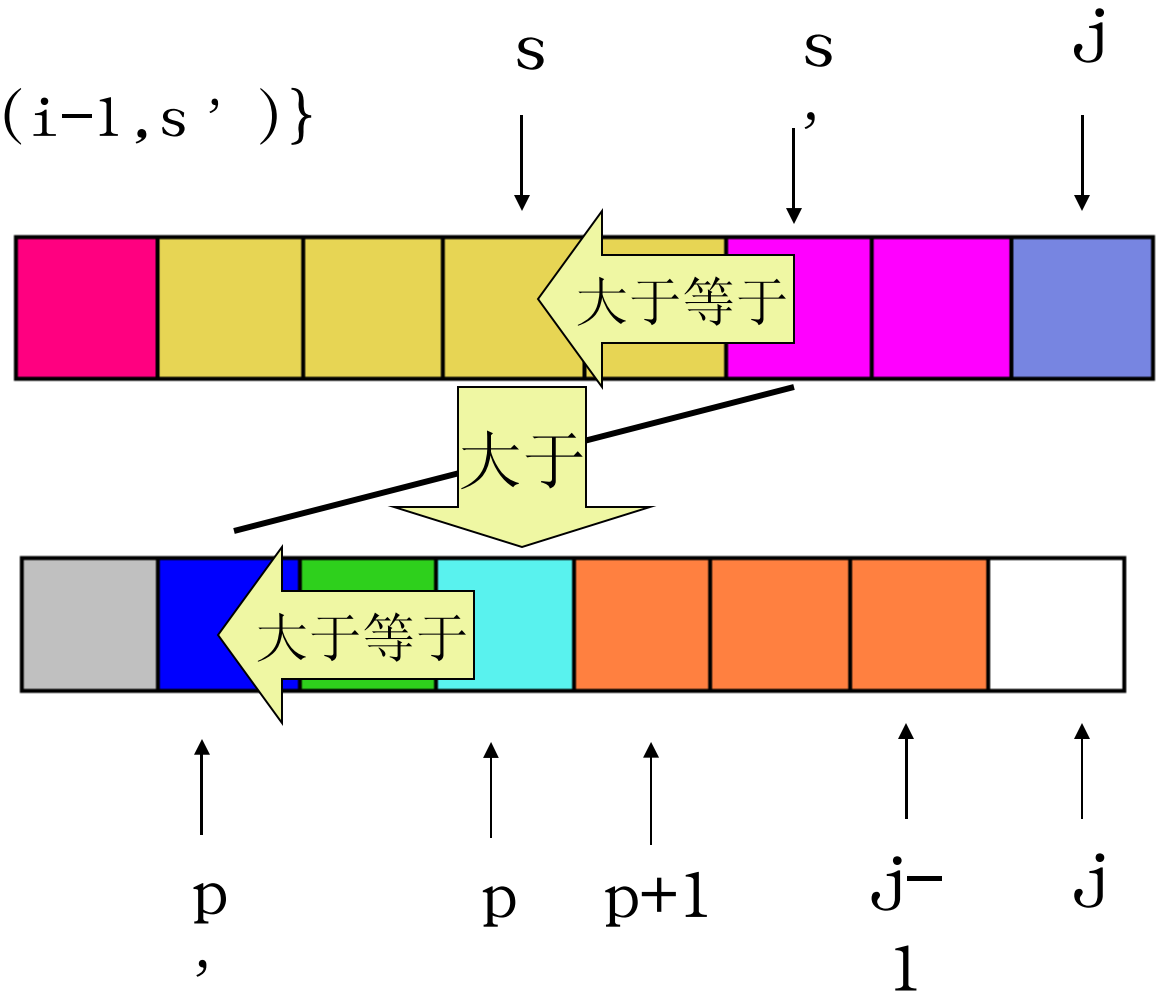
f(i-1,s)>f(i,p)=f(i,j-1)-1

情况一（p’<p)

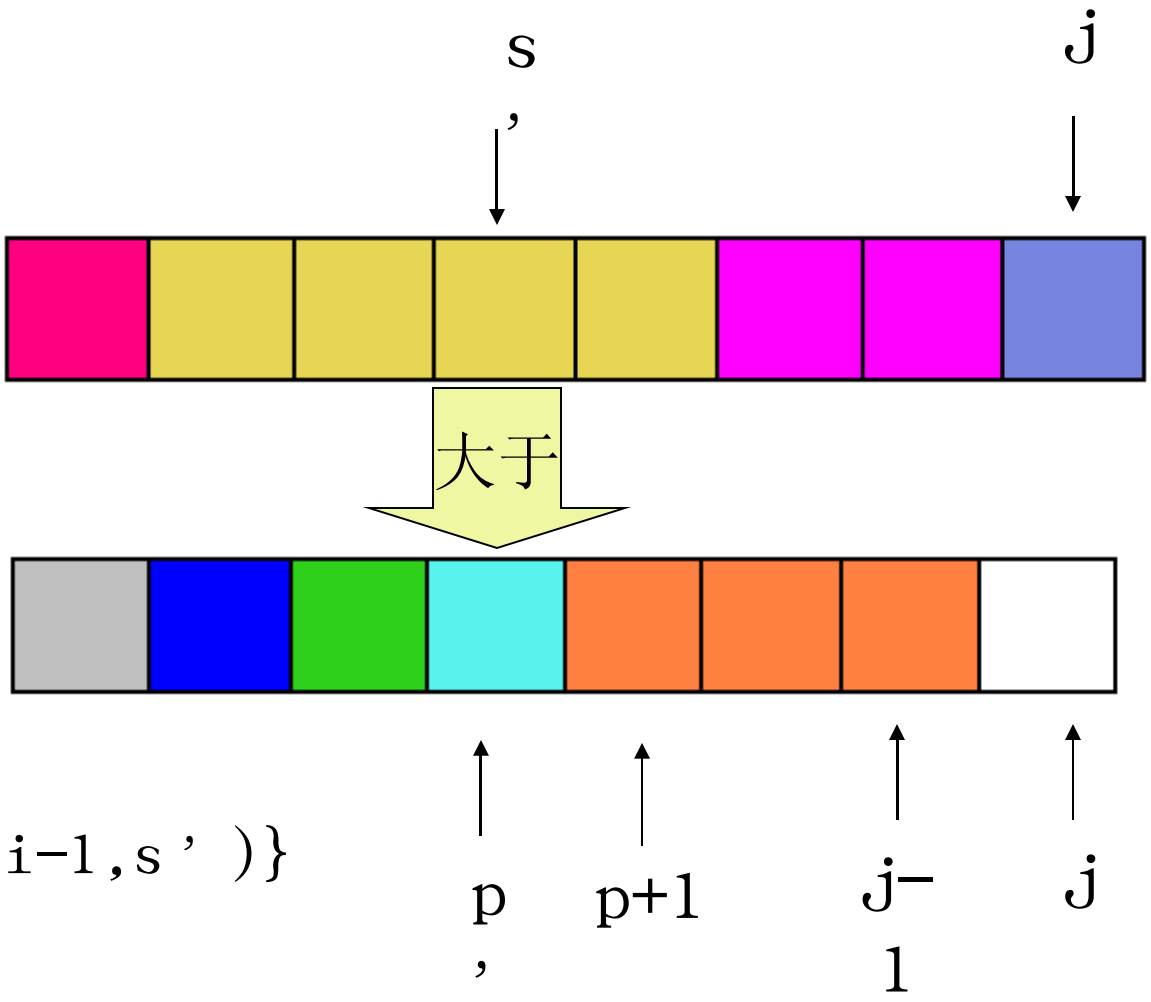
max{f(i,p’),f(i-1,s’)}+1>f(i,j-1)

f(i-1)

f(i)

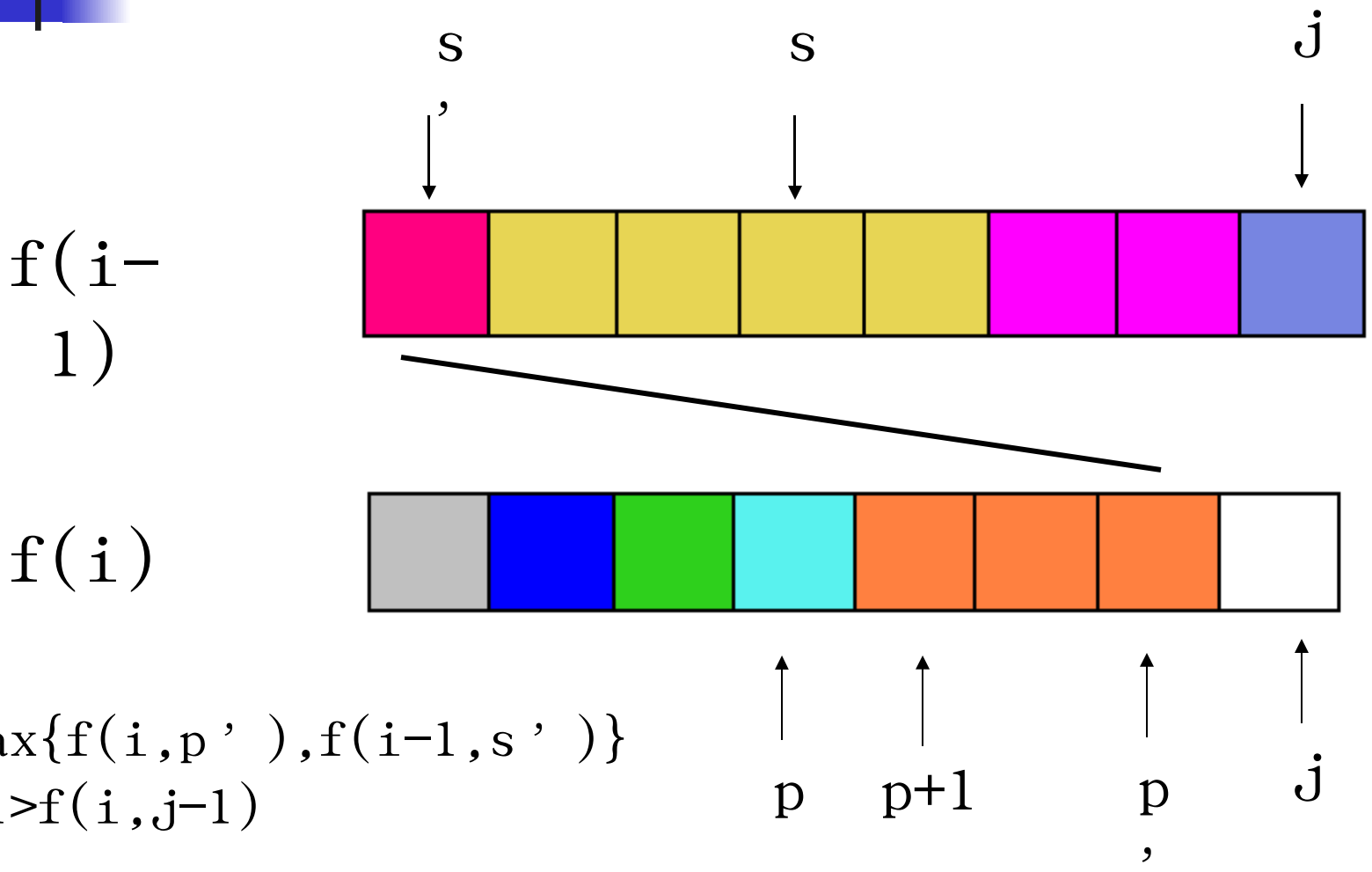


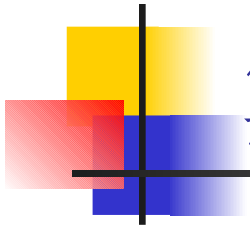
情况二（p’=p)



max{f(i,p’),f(i-1,s’)}
+1>f(i,j-1)

情况三（p'>p)





算法四

综上所述，

当 f(i,p)>=f(i-1,j-p-1) 时，可以直接得出 f(i,j)=f(i,j-1);

当 f(i,p)<f(i-1,j-p-1) 时 , 无论任何决策都不能使 f(i,j)=f(i,j-1), 所以此时 f(i,j)=f(i,j-1)+1.

因此，我们只需根据 f(i,p) 与 f(i-1,j-p-1) 的大小关系便可直接确定 f(i,j) 的取值，从而使状态转移成功地降为 O(1), 算法的时间复杂度降为 $O(N\log_2 N)$

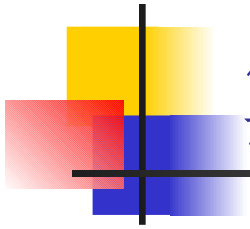
小结

这时我们会发现，经过了数次优化的动态规划模型已经不可能再有所改进了，对这题的讨论似乎可以到此为止了。但是，直到现在为止，我们还只是对一个动态规划模型进行优化。事实上，对于一道动态规划题目，往往可以建立多种模型。因此，我们不妨继续思考，来找到更高效的模型。

算法五

经过不懈努力，我们终于又找到了一种模型，在这种模型下的算法五，可以将时间复杂度降为$O(\sqrt{N})$。让我们来看一看算法五的精彩表现吧！

这里，我们需要定义一个新的动态规划函数 g(i,j)，它表示用 j 个蛋尝试 i 次在最坏情况下能确定 E 的最高楼层数。



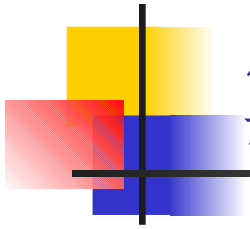
算法五

很显然，无论有多少鹰蛋，若只试 1 次就只能确定一层楼，即 g(1,j)=1 (j>=1)

而且只用 1 个鹰蛋试 i 次在最坏情况下可在 i 层楼中确定 E，即 g(i,1)=i (i>=1)

状态转移也十分简单。

g(i,j)=g(i-1,j-1)+g(i-1,j)+1　　(i,j>1)



算法五

我们的目标便是找到一个 x，使 x 满足 g(x-1,M)<N 且 g(x,M)>=N ，答案即为 x.

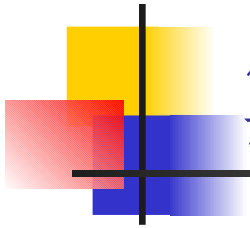
这个算法乍一看是 $O(N\log_2 N)$ 的，但实际情况却并非如此。

经过观察，我们很快会发现，函数 g(i,j) 与组合函数 C(i,j) 有着惊人的相似，而且可以很容易证明对于任意 i,j (i,j>=1)，总有 g(i,j)>= C(i,j).

算法五

这样，我们可以得到 C(x-1,M)<=g(x-1,M)<N 。

根据这个式子，我们可以证明运算量 (即 $\sqrt{xM}$) 与 $\sqrt{N}$ 同阶， 这里证明从略。因此，我们若在 M=1 时作特殊判断，就可以使运算量最差与 $\sqrt{N}$ 同阶。



算法五

在新的动态规划模型之下，我们找到了一个比前几种算法都优秀得多的方法。这就提醒我们不要总是拘泥于旧的思路。换个角度来审视问题，往往能收到奇效。倘若我们仅满足于算法四，就不能打开思路，找到更高效的解题方法。可见多角度地看问题对于动态规划的优化也是十分重要的。

总结

本文就《鹰蛋》一题谈了五种性能各异的算法，这里做一比较：

算法编号	时间复杂度	空间复杂度
算法一	$O(N^3)$	$O(N)$
算法二	$O(N^2\log_2 N)$	$O(N)$
算法三	$O(N(\log_2 N)^2)$	$O(N)$
算法四	$O(N\log_2 N)$	$O(N)$
算法五	$O(\sqrt{N})$	$O(\log_2 N)$

总结

从这张表格中，我们可以很明显地看出优化能显著提高动态规划算法的效率。并且，优化的方法也是多种多样的。这就要求我们在研究问题时必须做到：

深入探讨　　大胆创新

永不满足　　不断改进

总结

在实际问题中，尽管优化手段千变万化，但万变不离其宗，其本质思想都是：

一、找到动态规划算法中仍不够完美的部分，进行进一步改进；

二、另辟蹊径，建立新的模型，从而得到更高效的算法。

总结

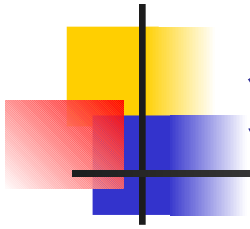
而在具体的优化过程中，需要我们做到以下几点：

减少状态总数

挖掘动态规划方程的特性

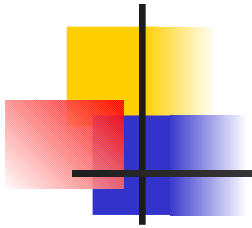
优化状态转移部分

建立新的动态规划模型



结束语

优化，再优化，让我们做得更好！



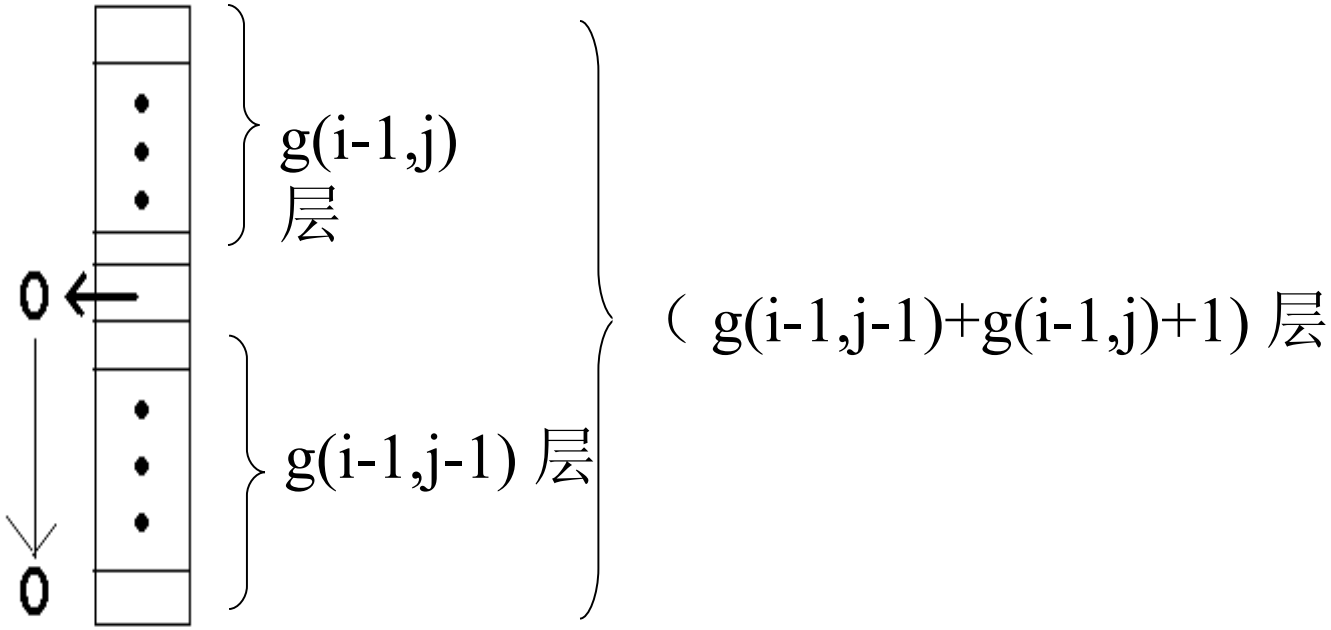
谢谢大家！

算法五

那么时间复杂度是怎样变为 O($\sqrt{N}$) 的呢？

观察 C(x-1,M)<N，这可以大致得出当 M 不太大时，x 与 $\sqrt[M]{N}$ 是同阶的。在实际情况中，可以发现只有当 M=1 时，xM=N；当 M>1 时，xM 立即降至 ($\sqrt{N}$) 的级别。因此，只需要在 M=1 时特殊判断一下就可以使算法的时间复杂度降为 O($\sqrt{N}$)，空间复杂度可用滚动数组降为 O(M), 即 O($\log_2 N$) 。

算法五



因此，有如下等式成立

：

$$g(i,j)=g(i-1,j-1)+g(i-1,j)+1 \quad (i,j>1)$$