# 数据结构与算法第三次作业

Shuxin Chen, 2015013229

2017 年 3 月 7 日

1. Knuth[212] 已经证明，对所有 $1 \le i < j \le n$，存在最优二叉搜索树，其根满足 $root[i,j-1] \le root[i,j] \le root[i+1,j]$。利用这一特性修改算法 OPTIMAL-BST，使得运行时间减少为 $\Theta(n^2)$。

   我们设 $l = j - i + 1$ 表示区间 $[i,j]$ 的宽度。在原算法中，我们用第一重循环从 1 到 $n$ 枚举 $l$，第二重循环枚举起始位置 $i$，并通过 $i$ 和 $l$ 算出 $j$，第三重循环枚举区间 $[i,j]$ 的可能根节点 $r$，并求出最优解。

   接下来我们通过对 $l$ 的归纳来证明, 对于任意的 $l$, 当 $i = 1, 2, \ldots, n-l+1$ 时, $root[i,j]$ 的值单调不降。

   当 $l = 1$ 时，对于任意的 $i$，显然均有 $root[i,j] = root[i,i] = i$，此时 $root[i,j]$ 单调递增，即满足单调不降。

   假设当 $l = l_0$ 时满足单调不降，则有

$$root[1,l_0] \le root[2,l_0+1] \le \cdots \le root[n-l_0+1,n]$$

   由 Knuth 的结论，我们有

$$\begin{gathered}
root[1,l_0] \le root[1,l_0+1] \le root[2,l_0+1] \\
root[2,l_0+1] \le root[2,l_0+2] \le root[3,l_0+2] \\
\cdots\cdots\cdots\cdots \\
root[n-l_0,n-1] \le root[n-l_0,n] \le root[n-l_0+1,n]
\end{gathered}$$

   将上面的式子联立，可以得到

$$root[1,l_0+1] \le root[2,l_0+2] \le \cdots \le root[n-l_0,n]$$

因此当 $l = l_0 + 1$ 时，同样满足单调不降。根据归纳假设，证明完成。我们取 $l$ 对应的第一项和最后一项，可以得到 $root[n-l+1,n] \geq root[1,l]$，并且因为它们的取值范围为 $[1,n]$，因此它们的差一定小于等于 $n-1$。

那么我们可以改变之前算法中的第三重循环，只需要枚举区间 $[root[i,j-1], root[i+1,j]]$ 即可。此时，对于任意的 $l = l_0$，第二重和第三重循环总共的运行次数为

$$
\begin{aligned}
&\sum_{i=1}^{n-l_0+1} (root[i+1, i+l_0-1] - root[1, l_0-1] + 1) \\
&= (n - l_0 + 1) + \sum_{i=1}^{n-l_0+1} (root[i+1, i+l_0-1] - root[1, l_0-1]) \\
&= (n - l_0 + 1) + (root[n-l_0+2, n] - root[1, l_0-1]) \\
&\leq (n - l_0 + 1) + (n - 1) \\
&= 2n - l_0 \\
&= \Theta(n)
\end{aligned}
$$

又因为 $l$ 的取值范围为 $[1,n]$，因此整个算法的时间复杂度为 $\Theta(n^2)$。

2. 给定一幅彩色图像，它由一个 $m \times n$ 的像素数组 $A[1 \ldots m, 1 \ldots n]$ 构成，每个像素是一个红绿蓝 (RGB) 亮度的三元组。嘉定我们希望轻度压缩这幅图像，具体地，我们希望从每一行中删除一个元素，使得图像边窄一个像素。单位了避免影响视觉效果，我们要求相邻两行中删除的像素必须位于同一列或相邻列。也就是说，删除的像素构成从顶端行到底端行的一条“接缝”(seam)，相邻像素均在垂直或对角线方向上相邻。

   a. 证明：可能的接缝的数量是 $m$ 的指数函数，假定 $n > 1$。

   b. 假定现在对每个像素 $A[i,j]$ 我们都已计算出一个实型的“破坏度”$d[i,j]$，表示删除像素 $A[i,j]$ 对图像可是效果的破坏程度。直观地，一个像素的破坏度越低，它与相邻像素的相似度越高。再假定一条接缝的破坏度定义为它包含的像素的破坏度之和。设计算法，寻找破坏度最低的接缝。分析算法的时间复杂度。

   首先，原版书中的问题 a 的表述为：Show that the number of such possible seams grows at least exponentially in $m$, assuming that $n > 1$. 这句话的正确翻译应该为：可能的接缝数量至少是 $m$ 的指数**级**函数。

   我们设 $count[i]$ 表示从顶端行到第 $i$ 行的接缝数量。显然，$count[1] = n$。我们试着寻找 $count[i-1]$ 到 $count[i]$ 的递推式。对于像素 $A[i,j]$，若它被删除，那么当它在边界

时，上一层可能删除的像素有两种取法；不在边界时，有三种取法。因此，我们得到了 $count[i]$ 的范围，即

$$count[i-1]*2 \leq count[i] < count[i]*3$$

这样递推下去，当 $i=m$ 时，有

$$n*2^{m-1} \leq count[m] < n*3^{m-1}$$

因此，可能的接缝数量至少是 $m$ 的指数级函数。

接下来，我们设计一个能够求出破坏度最低的接缝的算法。由于这个问题的输入是 $\Theta(m*n)$ 的，这是算法时间复杂度的下界，即算法的时间复杂度为 $\Omega(m*n)$。

我们用 $f[i,j]$ 表示从顶端到像素 $A[i,j]$ 的接缝的最低破坏度，那么问题的答案即为 $\max_{1\leq j\leq n}\{f[m,j]\}$。和上文求出接缝数量的方法类似，我们可以用第 $i-1$ 行已经计算出的答案来得出第 $i$ 行的答案。对于像素 $A[i,j]$，当它被删除时，第 $i-1$ 行删除的像素只能为 $j-1, j, j+1$ 列中的一个。那么 $f[i,j]$ 应该为 $f[i-1,j-1], f[i-1,j], f[i-1,j+1]$ 中的最小值加上 $d[i,j]$。考虑到边界情况，例如 $j=1$ 时，我们不能选择上一层的 $j-1$ 列，因此可以规定在边界外的 $f$ 值，即

$$\begin{cases} f[i,j]=0, & i=0 \\ f[i,j]=\infty, & i>0 \mid j=0 \mid j=n+1 \end{cases}$$

当 $1\leq i\leq m, 1\leq j\leq n$ 时，我们有

$$f[i,j]=\min\{f[i-1,j-1], f[i-1,j], f[i-1,j+1]\}+d[i,j]$$

求出了所有 $f$ 值之后，再求出上文提及的 $\max_{1\leq j\leq n}\{f[m,j]\}$，即可以得到接缝的最低破坏度。在递推的同时，我们记录每个 $f$ 状态是从上一层的哪个状态得出的，记为 $parent$ 数组。假设 $f[m,j_0]$ 处取到最低破坏度，那么我们从该处不断的取 $parent$，即 $f[m,j_0]$ 的上层状态为 $f[m-1, parent[m,j_0]]$，以此类推。当取到第 1 层时结束，此时我们就得到了最低破坏度对应的具体方案。

所有的 $f$ 值和 $parent$ 值可以通过下面的伪代码求出。

```
OPTIMAL-SEAM(M, N, d)
let f[0..M, 0..N+1], parent[1..M, 1..N] be new tables
for i = 0 to N
    f[0, i] = 0
    parent[0, i] = -1
for i = 1 to M
    f[i, 0] = ∞
    f[i, N+1] = ∞
    for j = 1 to N
        prevMin = min{f[i-1, j-1], f[i-1, j], f[i-1, j+1]}
        if prevMin == f[i-1, j-1]
            f[i, j] = f[i-1, j-1] + d[i, j]
            parent[i, j] = j - 1
        elseif prevMin == f[i-1, j]
            f[i, j] = f[i-1, j] + d[i, j]
            parent[i, j] = j
        else
            f[i, j] = f[i-1, j+1] + d[i, j]
            parent[i, j] = j + 1
return f, parent
```

由伪代码可以看出，在双重循环中的代码数量为 $\Theta(1)$，因此整个算法的时间复杂度为 $\Theta(m * n)$，达到了理论复杂度的下界。