**1. 修改MEMOIZED-CUT-ROD，使之不仅返回最优收益值，还返回切割方案。**

|  |
| --- |
| MEMORIZRED-CUT-ROD-SOLUTION(p,n)  let r[0..n], tag[0..n] be new arrays  for i = 0 to n  r[i] = -∞  tags[i] = -∞  (q, tags) = MEMOIZED-CUT-ROD-AUX(p, n, r, tags)  print “max value is ”+q  i = n  while i > 0  print tags[i]  i = i – tags[i]  MEMORIZED-CUT-ROD-AUX(p, n, r, tags)  if r[n]≥0  return (r[n], tags)  if n = 0  q=0  tags[n]=0  else  q = -∞  for i = 1 to n  (temp, tags) = MEMORIZED-CUT-ROD-AUX(p, n-i, r, tags)  if q < p[i] + temp  q = p[i] + temp  tags[n] = i  r[n] = q  return (q, tags) |

**2. 在求解矩阵链乘法问题时我们总是可以在求解子问题之前选定AiAi+1…Aj的划分位置Ak（选定的k使得pi-1pkpj最小）。请找出一个反例，证明这个贪心方法可能生成次优解。**

假设A1，A2，A3，A4对应的p0，p1，p2，p3，p4，p5分别为10，5，2，1，10。根据Capulet教授的建议，在求解子问题之前可以选定A3作为划分位置（因为10\*1\*10最小），而在A1，A2，A3中选择A2作为划分位置，（因为10\*2\*1最小），因此划分结果为:

（（（A1A2）A3）A4）

需要的乘法次数为

而求解子问题得到的划分结果为

（（A1（A2A3））A4）

需要的乘法次数为

因此这个贪心方法可能生成次优解。

**3. 基于seam carving的图像压缩算法。**

以按行进行seam carving为例：

（1） 计算每一个像素点的破坏度

（2） 计算第一行每个像素的破坏度和，也就是每个像素的破坏度本身，标记第一行像素选取的都是自身。

（3） 从第二行开始，计算每个像素的破坏度和的最小值，每个像素的破坏度和最小值是像素本身的破坏度与上一行同一列或相邻列破坏度之和的最小值。记录标记选取上一行哪一列的破坏和。

（4）找出最后一行破坏值之和最小的一个像素，根据标记向上查找删除的一条接缝。

算法伪代码为：

|  |
| --- |
| SEAM\_CARVING (Pic, Energy, Result)  for i = 0 to width  last[i] = Energy[0][i]  tags[0][i] = 0  for i = 1 to height  for j = 0 to width  if j-1 >= 0 && j+1<=width-1  if last[j-1] < last[j]  current[j] = Energy[i][j] + last[j-1]  tags[i][j] = -1  else if last[j+1] < last[j]  current[j] = Energy[i][j] + last[j+1]  tags[i][j] = 1  else  current[j] = Energy[i][j] + last[j]  tags[i][j] = 0  //讨论边界情况，有边界则不比较该边  else if j-1 < 0  ……  else  ……  last = current  minIndex = 0  minValue = INFINITE  for i = 0 to width  if last[i] < minValue  minValue = last[i]  minIndex = i  for i = 0 to height  Result[i] = minIndex  minIndex = minIndex + tags[i][minIndex]  return Result//返回每一行需要删除的列坐标 |