第六章 动态规划

6.3

分析：

可以不记录，而用两个变量f1和f2来保存源程序中的和，并在一轮循环中用f1 f2来计算和即新的f1,f2。因而可以节省2n-2个空间，在最后一轮循环后即j=n后，使之仍然能够计算出f\*，并且仍然能够构造出最快装配路线。F1,f2保存的为和。根据这两个值，就能计算出f\*。

6.5

类似于归并算法

分治法,其中s数组存放记录的是划分位置k

MatrixChainMultiply(,s,i,j)

1. if i =j then
2. return
3. else
4. q←s[i,j]
5. A1←MatrixChainMultiply(,s,i,q)
6. A2←MatrixChainMultiply(,s,q+1,j)
7. return MatrixMultiply(A1,A2)

6.7

备忘录方法是一种自顶向下的高效动态规划方法，它可能包含递归过程，并在第一次解决一个子问题时将结果记录到一个表中，在下一次遇见孩子子问题时，只需查表而无需再解决一次，因而当某个算法有重叠子问题时，能很高效地解决问题。而MergeSort算法的每个子问题都是相互独立的，不存在重叠子问题，因而使用备忘录方法并不能提高效率。

6.9

个人理解这道题的原算法本质就是通过观测b数组中记录的路径来找到LCS的来源。而B数组的得到方式是通过X[]和Y[]的比较而来，当不存在b数组时，只需要重新比较X[]和Y[]更换成条件即可。

PrintLCS(c,X,Y,i,j)

1. if i=0 or j=0 then
2. return 0
3. if X[i]=Y[j] then
4. PrintLCS(c,X,Y,i-1,j-1)
5. print x[i]
6. else jf c[i-1,j]>=c[I,j-1] then
7. PrintLCS(c,X,Y,i-1,j)
8. else
9. PrintLCS(c,X,Y,i,j-1)

6.12

动态规划法（时间复杂度O(N^2))

设长度为N的数组为{a0，a1, a2, ...an-1)，则假定以aj结尾的数组序列的最长递增子序列长度为L(j)，则L(j)={ max(L(i))+1, i<j且a[i]<a[j] }。也就是说，我们需要遍历在j之前的所有位置i(从0到j-1)，找出满足条件a[i]<a[j]的L(i)，求出max(L(i))+1即为L(j)的值。最后，我们遍历所有的L(j)（从0到N-1），找出最大值即为最大递增子序列。时间复杂度为O(N^2)。

例如给定的数组为{5，6，7，1，2，8}，则L(0)=1, L(1)=2, L(2)=3, L(3)=1, L(4)=2, L(5)=4。所以该数组最长递增子序列长度为4，序列为{5，6，7，8}。算法代码如下：

LIS(A,L)

1. for i←0 to n do
2. L[i]←1
3. for j←0 to n do
4. for i←0 to j do
5. if(A[j]>A[i]Z and L[j] < L[i]+1) then
6. L[j] =L[i]+1
7. for i←0 to n do
8. max(L[i])
9. return max

解法3：O(NlgN）算法

假设存在一个序列d[1..9] ={ 2，1 ，5 ，3 ，6，4， 8 ，9， 7}，可以看出来它的LIS长度为5。

下面一步一步试着找出它。

我们定义一个序列B，然后令 i = 1 to 9 逐个考察这个序列。

此外，我们用一个变量Len来记录现在最长算到多少了

首先，把d[1]有序地放到B里，令B[1] = 2，就是说当只有1一个数字2的时候，长度为1的LIS的最小末尾是2。这时Len=1

然后，把d[2]有序地放到B里，令B[1] = 1，就是说长度为1的LIS的最小末尾是1，d[1]=2已经没用了，很容易理解吧。这时Len=1

接着，d[3] = 5，d[3]>B[1]，所以令B[1+1]=B[2]=d[3]=5，就是说长度为2的LIS的最小末尾是5，很容易理解吧。这时候B[1..2] = 1, 5，Len＝2

再来，d[4] = 3，它正好加在1,5之间，放在1的位置显然不合适，因为1小于3，长度为1的LIS最小末尾应该是1，这样很容易推知，长度为2的LIS最小末尾是3，于是可以把5淘汰掉，这时候B[1..2] = 1, 3，Len = 2

继续，d[5] = 6，它在3后面，因为B[2] = 3, 而6在3后面，于是很容易可以推知B[3] = 6, 这时B[1..3] = 1, 3, 6，还是很容易理解吧？ Len = 3 了噢。

第6个, d[6] = 4，你看它在3和6之间，于是我们就可以把6替换掉，得到B[3] = 4。B[1..3] = 1, 3, 4， Len继续等于3

第7个, d[7] = 8，它很大，比4大，嗯。于是B[4] = 8。Len变成4了

第8个, d[8] = 9，得到B[5] = 9，嗯。Len继续增大，到5了。

最后一个, d[9] = 7，它在B[3] = 4和B[4] = 8之间，所以我们知道，最新的B[4] =7，B[1..5] = 1, 3, 4, 7, 9，Len = 5。

于是我们知道了LIS的长度为5。

注意，这个1,3,4,7,9不是LIS，它只是存储的对应长度LIS的最小末尾。有了这个末尾，我们就可以一个一个地插入数据。虽然最后一个d[9] = 7更新进去对于这组数据没有什么意义，但是如果后面再出现两个数字 8 和 9，那么就可以把8更新到d[5], 9更新到d[6]，得出LIS的长度为6。

然后应该发现一件事情了：在B中插入数据是有序的，而且是进行替换而不需要挪动——也就是说，我们可以使用二分查找，将每一个数字的插入时间优化到O(logN)~~~~~于是算法的时间复杂度就降低到了O(NlogN)～！

代码如下（代码中的数组B从位置0开始存数据）：

数组B记录LIS序列

LIS2(A)

1. len=1
2. B[0]=A[0]
3. for i←0 to n do
4. if(A[i] >B[len-1])
5. B[len] = A[i]
6. len++
7. else
8. pos = BiSearch(B,len,A[i]) //二分法查找插入的位置
9. B[pos]=A[i]
10. return len

智能科学与技术系

周雨

31520181154417