中国科学技术大学
University of Science and Technology of China

算法习题课-第三章

容圣海

3.1最长单调递增子序列$O(n^2)$

中国科学技术大学 University of Science and Technology of China

- 问题输入： 由n个数组成的序列
- 问题的输出：序列中最长单调递增子序列
- 问题的规模：序列长度n

解法一：转化为求最长公共子序列

（求最长递增子序列问题 转化为 求最长公共子序列的问题）

将序列a 按非递减顺序排列，形成新序列b，问题就转变成求解a 和b的最长公共子序列。假设使用快速排序，则排序过程的时间复杂度为$O(n\log n)$， 而求两个序列的最长公共子序列的时间复杂度为$O(n^2)$（a,b长度相等，都为n），该解法的时间复杂度为$O(n^2)$。

3.1最长单调递增子序列$O(n^2)$

解法二：

用数组$b[0:n-1]$记录以$a[i]$ $i\in[0,n-1]$为结尾元素的最长单调递增子序列的长度。（先不求解最长的单调递增子序列，而是先求其长度）

$b[i]$表示当以$a[i]$为单调递增子序列最后一个元素时，所得最长单调递增子序列的长度。然后找出数组b中的最大元素，就是序列a的最长单调递增子序列。我们可以得到递推式：

$$b[i]=\begin{cases} 1 & i=0 \\ \max\limits_{0\le k<i}(b[k]+1) & a[k]\le a[i] \end{cases}$$

降规模的体现：若找到某一个元素是单调递增子序列中的结尾元素，则其后的元素便不再考虑。

最优子结构验证：$b[i]$表示当以$a[i]$为单调递增子序列最后一个元素时，所得最长单调递增子序列的长度。很明显，满足最优子结构性质。

复杂度分析：遍历n个元素，计算b值；遍历n次，找出b的最大值。

算法复杂度$O(n^2)$。

3.2最长单调递增子序列O(nlogn)

中国科学技术大学 University of Science and Technology of China

分析：

假设序列a，其最长单调递增子序列的长度为m

考虑其长度为i的单调子列$(1 <= i <= m)$，这样的子列可能有多个，选取这些子列的

结尾元素的最小值，用L_i表示。易知：$L_1 <= L_2 <= L_3 <= ... <= L_m$。

现在，需要寻找序列a 对应的L 序列，如果找到的最大的L_i是L_m，那么m就是最

大单调子列的长度。

从左至右扫描a，对于每一个a_i

1) $a_i < L_1, L_1 = a_i$

2) $a_i > L_k$,则 $L_{\mathrm{k+1}} = a_i$, k = k+1

3) $L_s \leq a_i \leq L_{s+1}$, 则$L_{s+1}=a_i$

扫描完成后，就得到了最长递增子序列的长度。

从上述方法可知，对于每一个元素，我们需要对L 进行查找操作，由于L是有

序的，使用二分查找，复杂度为$\log n$，于是总的复杂度为$O(n \log n)$。

3.1 任务调度问题

- 输入：需要处理的作业总数，A，B处理第i个作业的时间
- 输出：完成n个作业的最短时间
- 问题分析：当完成了k个作业，设机器A共花费了时间x，机器B花费时间的最小值是x的一个函数。设T[x][k]是机器B花费时间的最小值，有

$$T[x][k]=\begin{cases}\sum_{i=1}^{k} b[i] & x=0\\ \min\{T[x-a[k]][k-1],T[x][k-1]+b[k]\} & x\geq 0\end{cases}$$

- 综上：完成n个作业的总时间是max(x, T[x][n])
- 最优子结构：因为计算时间是加法计算，显然子问题的最优解也是问题的最优解
- 复杂度： $n\times\min\{\sum a_i,\sum b_i\}$

3.5 乘法表

设m(i, j, x)表示字符串加括号后值为x的方式数，求m(1,n,a)则有

$$m(i,j,a)=\sum_{k=i}^{j-1}[m(i,k,a)m(k+1,j,c)+m(i,k,b)m(k+1,j,a)+m(i,k,c)m(k+1,j,a)]$$

$$m(i,j,b)=\sum_{k=i}^{j-1}[m(i,k,a)m(k+1,j,a)+m(i,k,a)m(k+1,j,b)+m(i,k,b)m(k+1,j,b)]$$

$$m(i,j,c)=\sum_{k=i}^{j-1}[m(i,k,b)m(k+1,j,a)+m(i,k,c)m(k+1,j,b)+m(i,k,c)m(k+1,j,c)]$$

最优子结构：因为原问题最优值由子问题最优值相加和相乘得到，所以满足最优子结构。
复杂度：$O(\boldsymbol{n}^3)$

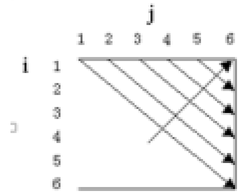
	a	b	c
a	b	b	a
b	c	b	a
c	a	c	c

One more thing

矩阵连乘问题(最优三角剖分是类似的)

$$m[i,j]=\begin{cases} 0 & i=j \\ \min\limits_{i\le k<j}\{m[i,k]+m[k+1,j]+p_{i-1}p_kp_j\} & i<j \end{cases}$$

$$m[2][5]=\min\begin{cases} m[2][2]+m[3][5]+p_1p_2p_5=0+2500+35\times15\times20=13000 \\ m[2][3]+m[4][5]+p_1p_3p_5=2625+1000+35\times5\times20=7125 \\ m[2][4]+m[5][5]+p_1p_4p_5=4375+0+35\times10\times20=11375 \end{cases}$$



(a) 计算次序

i \ j	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

(b) m[i][j]

流水作业调度

流水作业调度问题的Johnson算法

(1) 令 $N_1=\{i\,|\,a_i<b_i\}, N_2=\{i\,|\,a_i\ge b_i\}$;

(2) 将 N_1 中作业依 a_i 的非减序排序；将 N_2 中作业依 b_i 的非增序排序；

(3) N_1 中作业接 N_2 中作业构成满足 Johnson 法则的最优调度。

One more thing

❖ 0-1背包问题$O(nc)$

$$m(i,j)=\begin{cases}\max\{m(i+1,j),m(i+1,j-w_i)+v_i\} & j\ge w_i\\ m(i+1,j) & 0\le j<w_i\end{cases}$$

$$m(n,j)=\begin{cases}v_n & j\ge w_n\\ 0 & 0\le j<w_n\end{cases}$$

❖ 0-1背包问题的算法改进

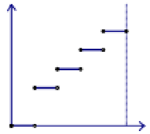
$n=5$，$c=10$，$w=\{2，2，6，5，4\}$，$v=\{6，3，5，4，6\}$

初始时$p[6]=\{(0,0)\}$，$(w5,v5)=(4,6)$。因此，$q[6]=p[6]\oplus(w5,v5)=\{(4,6)\}$；

$p[5]=\{(0,0),(4,6)\}$；

$q[5]=p[5]\oplus(w4,v4)=\{(5,4),(9,10)\}$。从跳跃点集$p[5]$与$q[5]$的并集

$p[5]\cup q[5]=\{(0,0),(4,6),(5,4),(9,10)\}$中看到跳跃点$(5,4)$受控于跳跃点$(4,6)$。将受控跳跃点$(5,4)$清除后，得到$p[4]=\{(0,0),(4,6),(9,10)\}$



$q[4]=p[4]\oplus(6，5)=\{(6，5)，(10，11)\}$

$p[3]=\{(0，0)，(4，6)，(9，10)，(10，11)\}$

$q[3]=p[3]\oplus(2，3)=\{(2，3)，(6，9)\}$

$p[2]=\{(0，0)，(2，3)，(4，6)，(6，9)，(9，10)，(10，11)\}$

$q[2]=p[2]\oplus(2，6)=\{(2，6)，(4，9)，(6，12)，(8，15)\}$

$p[1]=\{(0，0)，(2，6)，(4，9)，(6，12)，(8，15)\}$

$p[1]$的最后的那个跳跃点$(8,15)$给出所求的最优值为$m(1,c)=15$