算法分析与设计

教材:

[1][王] 王晓东,计算机算法设计与分析(第4版),电子工业.

[2][S] 唐常杰等译, Sipser著, 计算理论导引, 机械工业.

参考资料:

[3][C] 潘金贵等译, Cormen等著, 算法导论, 机械工业.

[4][M] 黄林鹏等译, Manber著, 算法引论-一种创造性方法, 电子.

[5][刘] 刘汝佳等, 算法艺术与信息学竞赛, 清华大学.

[6][L] Lewis等著, 计算理论基础, 清华大学.

第一章

算法分析题 1-1, 1-2, 1-4

第1章 概论

1-1 求下列函数的渐近表达式:

$3n^2+10n$; $n^2/10+2^n$; $21+1/n$; $\log n^3$; $10\log 3^n$.

解: $3n^2+10n = \Theta(n^2)$; $n^2/10+2^n = \Theta(2^n)$;

$21+1/n = \Theta(1)$; $\log n^3 = \Theta(\log n)$;

$10\log 3^n = \Theta(n)$;

1-2 试论O(1)与O(2)的区别.

答: 没有区别, 因为根据定义 $1 = O(2)$, $2 = O(1)$

第1章 概论

1-4 (1) 假设某算法在输入规模为n时的计算时间为$T(n)=3\times 2^n$. 在某台计算机上实现并完成该算法的时间为t秒. 现有另一台计算机, 其运行速度是第一台的64倍, 那么在这台新机器上用同一算法在t秒内能解输入规模为多大的问题?

(2) 若上述算法的计算时间改进为$T(n)=n^2$, 其余条件不变, 则在新机器上用t秒时间能解输入规模为多大的问题?

(3) 若上述算法的计算时间改进为T(n)=8, 其余条件不变, 那么在新机器上用t秒时间能解输入规模为多大的问题?

解: 设机器1上t秒能解的问题规模为 n1,

机器2上t秒能解的问题规模为 n2.

(1) 由 $t = 3\times 2^{n1} = 3\times 2^{n2}/64$ 知, n2=n1+6, 所以规模增大6.

(2) 由 $t = n1^2 = n2^2/64$ 知, n2 = 8 n1, 所以规模增大7倍.

(3) 若 $t \geq 8$ 则n1可以任意大, 若$t \geq 1/8$则n2可以任意大.

第2章 分治

2-8 设n个不同的整数排好序后存于T[1:n]中. 若存在一个下标i, $1 \le i \le n$, 使得T[i]=i. 设计一个有效算法找到这个下标. 要求算法在最坏情况下的计算时间$O(\log n)$.

解: 不同整数意味着要么严格递增, 要么严格递减

若T[1:n]严格递减, 则

T[i]<i蕴含$\forall j>i$(T[j]<j), T[i]>i蕴含$\forall j<i$(T[j]>j)

满足二分法条件, 可用二分搜索

若T[1:n]严格递增,

T[i]>i蕴含$\forall j>i$(T[j]>j), T[i]<i<0蕴含$\forall j<i$(T[j]<j)

也满足二分法条件, 可用二分搜索

满足二分法条件也意味着至多有一个解.

算法略

第2章 分治

2.9 设T[0:n-1]是n个元素的数组. 对任一元素x, 设S(x)={ i | T[i]=x}.当|S(x)|>n/2时, 称x为主元素. 设计一个线性时间算法, 确定T[0:n-1]是否有一个主元素.

算法1: 性质: 若数列有主元素, 则中位数必为主元素.

先找中位数a, 即第⌊(n+1)/2⌋大的数, 在计数a出现次数.

若a出现次数大于n/2, 则a即为主元素; 否则无主元素.

找中位数时间O(n), 计数a出现次数时间O(n).

算法2: 性质: 若数列有主元素, 则去掉两个不同数, 主元素不变.

1. p=T[0], ct=1, i=1, //p记可能主元素, ct为计数器,

2. 当i<n, 若T[i]=p,则(ct++,i++);否则(ct- -,i++;若ct==0, p=T[i],i++)

3. 判断p是否为主元素.

注1: map对应平衡二叉树每次插入删除搜索时间是O(logn)

注2: 有人用计数的方法, 当知道数组T的取值范围时是可行的.

第3章 动态规划

1. 考虑下面的整数线性规划问题 .

即给定 序列$a_1, a_2,\ldots,a_n$, 求

$$\max c_1x_1+ c_2x_2+\ldots+ c_nx_n,$$

满足 $a_1x_1+ a_2x_2+\ldots+ a_nx_n \le b$, x_i为非负整数

解: 动态规划, 子结构[1:i], OSP

设f[i][k]为用X[1:i]组合出重量k的最大价值

则 f[i][k]=max{f[i-1][k],f[i][k-x[i]]+c[i]}

第3章 动态规划

1. 考虑下面的整数线性规划问题 .

即给定 序列$a_1, a_2,\ldots,a_n$, 求

$$\max c_1x_1+ c_2x_2+\ldots+ c_nx_n,$$

满足 $a_1x_1+ a_2x_2+\ldots+ a_nx_n \le b$, x_i为非负整数

1. 初始f[1:n]=0, f[0]=0
2. 对i=1:k, 对s=1:n,
3. 若s≥X[i] 且 c[i]+f[s-X[i]]>f[s], 则f[s]+=f[s-X[i]]+c[i]
4. 输出f[n]

第3章 动态规划

2. 石子合并问题

问题描述: 在一个圆形操场的四周摆放着n堆石子. 现在要将石子有次序地合并成一堆. 规定每次只能选相邻的2堆石子合并成一堆, 并将新的一堆石子数记为该次合并的得分. 试设计一个算法, 计算出将n堆石子合并成一堆的最小得分和最大得分.

算法设计: 对于给定n堆石子, 计算合并成一堆的最小得分和最大得分.

数据输入: 由文件input.txt提供输入数据. 文件的第1行是正整数n, 1≤n≤100, 表示有n堆石子. 第2行有n个数, 分别表示n堆石子的个数.

结果输出: 将计算结果输出到文件output.txt, 文件第1行是最小得分, 第2行是最大得分.

输入文件示例

input.txt

4

4 4 5 9

输出文件示例

output.txt

43

54

第3章 动态规划

先讨论直线上石子合并问题的算法

- 动规, 子结构[i:j], OSP, 类似于矩阵连乘问题
- 定义m[i,j]为从第i堆到第j堆的石子合并能得到的最少分数, 那么

  $m[i,j] = \min \{ m[i,k]+m[k+1,j]+ sum[i:j] \mid i \le k < j \}$

  其中sum[i:j]是第i堆到第j堆石子总数
- 修改矩阵连乘公式可以得到下面的算法(其中s[i,j]是最佳分断点)

```
1. 对 i = 1 到 n,  m[i,i]=0,
2. 对 r = 1 到 n-1
3.      对i = 1 到 n-r
4.           j = i + r; s[i,j] = i;
5.           m[i,j] = m[i,i]+m[i+1,j]+ sum[i:j];
6.           对 k = i + 1 到 j-1
7.                t =m[i,k]+m[k+1,j]+ sum[i:j],
8.                若m[i,j]>t, 则m[i,j]=t; s[i,j]=k;
```

输出m[1,n], 合并次序

```
Traceback(i, j, s)
1.   若i = = j, 打印 a[i]
2.   否则   打印 “(”
3.        Traceback(i, s[i,j], s)
4.        打印 “+”
4.        Traceback(s[i,j]+1,j,s)
5.        打印 “)”
```

第3章 动态规划

再讨论圆周上的石子合并问题, 子结构[i:j]稍作修改

- 定义m[i][len]为合并第i堆到第i+len-1堆石子能得到的最少分数
- 当i+len-1>n时, 指跨过第n堆到第(i+len-1)%n堆,
  仅sum函数需要修改
- m[i][len]=min{ m[i][k]+m[i+k][len-k]+ sum[i:i+len-1] | 0 ≤ k < len}
- s[i][len]记从i到i+len-1最佳分断点

```
1. 对 i = 1 到 n,  m[i][1]=0, s[i][1]=i
2. 对 len = 2 到 n
3.    对i = 1 到 n
4.       s[i][len] = i; j=i+len-1;
5.       m[i][len] = m[i][1]+m[i+1][len-1]+ sum[i:j];
6.       对 k = 2 到 len-1
7.          t=m[i][k]+m[i+k][len-k]+ sum[i:j],
8.          若m[i][len]>t, 则m[i][len]=t; s[i][len]=k;
```

输出 min { m[i][n] |1≤i≤n }
类似可以

- 打印合并次序
- 由加速原理加速

第3章 动态规划

3. 数字三角形问题

问题描述: 给定一个有n行数字组成的数字三角形, 如下图所示. 试设计一个算法, 计算出从三角形的顶至底的一条路径, 使该路径经过的数字和最大.

算法设计: 对于给定的n行数字组成的三角形, 计算从三角形顶至底的路径经过的数字和的最大值.

数据输入: 由文件input.txt提供输入数据. 文件的第1行数字三角形的行数n, 1≤n≤100. 接下来n行是数字三角形各行中的数字. 所有数字在0~99之间.

结果输出: 将计算结果输出到文件output.txt, 文件第1行中的数是计算出的最大值.

```
   7
  3   8
 8   1   0
2  7   4   4
4  5  2   6   5
```

数字三角形

输入文件示例

input.txt

```
5
7
3 8
8 1 0
2 7 4 4
4 5 2 6 5
```

输出文件示例

output.txt

```
30
```

第3章 动态规划

动规, 两种方式, 自顶向下, 自底向上

• 自顶向下

定义m[i,j]为从第1行到第i行第j列能得到的最大分数, 那么

m[i,j] = a[i,j] + max { m[i-1,j], m[i-1,j-1]}, 当j≤i; =0, 当j>i或j=0.

```
1. m[2:n]=0, m[1]=a[1,1], m[0]=0,
2. 对 i = 2 : n
3.      对j = i : 1
4.           若m[j-1]>m[j], 则m[j]=m[j-1]
5.           m[j]+=a[i,j]
6. 输出 max { m[j] | 1≤j≤n }
```

第3章 动态规划

动规, 两种方式, 自顶向下, 自底向上

• 自底向上

定义m[i,j]为从第1行到第i行第j列能得到的最大分数, 那么

m[i,j] = a[i,j] + max { m[i+1,j], m[i+1,j+1]}, 当j≤i

```
1. 对j=1:n, m[j]=a[n,j],
2. 对 i = n-1 到 1
3.      对j = 1 到 i
4.           若m[j+1]>m[j], 则m[j]=m[j+1]
5.           m[j]+=a[i,j],
6. 输出m[1]
```

电子人的基因：
输入两个A～Z组成的字符串（长度均不超过30），找一个最短的串，使得输入的两个串均是它的子序列（不一定连续出现）。
你的程序还应统计长度最短的串的个数。

例如，ABAAXGF和AABXFGA的最优解之一为AABAAXGFGA，一共有9个解。

分析：

参考LCS问题的思路。记输入的两个字符串为S1和S2，定义pa(i,j)为S1[1…i]和S2[1…j]的公共父串的最短长度。则pa的转移方程如下：

- 当S1[i]!=S2[j],对应的父串的最后一位是使用S1[i] 还是S2[j] ——

  pa[i,j]=min(pa(i-1,j)+1,pa(I,j-1)+1)

- 当S1[i] =S2[j],对应的父串的最后一位是确定的

  pa[i,j]=pa(i-1,j-1)+1)

边界条件是当i=0或j=0时，pa[i.j]=max(i,j),则父串是S1或S2其中一个。

- 串折叠问题

给出一个由大写字母组成的长度为n（$1 \le n \le 100$）的串，“折叠”成一个尽量短的串。

例如，AAAAAAAAAAABABABCCD折叠成9(A)3(AB)CCD。

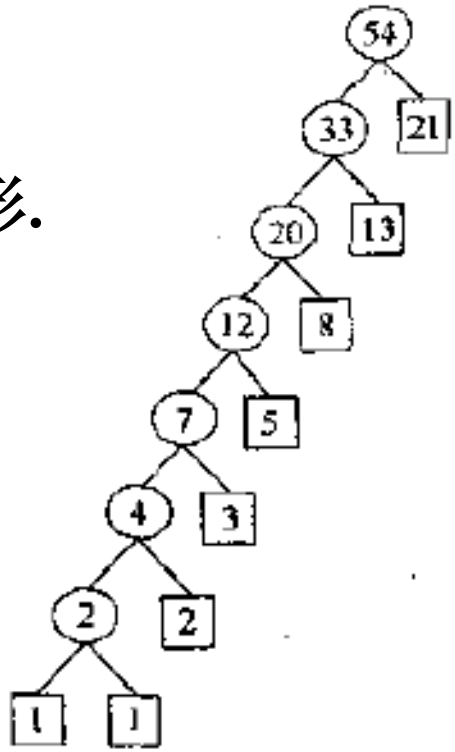
折叠是可以嵌套的，例如，NEERCYESYESYESNEERCYESYESYES可以折叠成2(NEERC3(YES))。多解时可以输出任意解。

- 参考最有矩阵连乘问题的思路，进行区间的划分。记输入串为S，DP(L,R)表示这个区间的最优折叠结果的字串表示，用STL的string来存储。
- 状态转移方程如下：
- 边界条件：当L=R时， DP(L,R)=“S[L]”

- 当L<R时，则考虑两种策略，去长度最短的方案：
  - 把区间切分成两个部分，分别折叠然后拼接，遍历所有的区间切分方案[L,K]和[k+1，R],求出令DP[L,K]和DP[K+1,P]长度之和最小的k值kMin，则有， DPL,R]=DP[L,Kmin]+DP[Kmin+1,R]
  - 如果S[L,R]是周期串，首先求出最小的重复串的长度，记为rep=(L-R+1)/cycle，也就是重复次数。这样就把区间变成DP[L,L+cycle-1]的rep次折叠。例如，ABABABAB变成4（AB）

第四章 贪心

1. 字符a~h出现的频率恰好是前8个Fibonacci数,
它们的Huffman编码是什么? 将结果推广到
n个字符的频率恰好是前n个Fibonacci数的情形.



解:根据a~h的频率, 画出Huffman编码树如右图
所以各字符编码为:
h:1, g:01, f:001, e:0001, d:00001, c:000001,
b:0000001, a:0000000,
推广到n个符号的情形. 记第i个符号为i,
则f[i]=f[i-1]+f[i-2]
由数学归纳法易证明 $sum_{i=1}^{k} f[i] < f[k+2]$
从而也以类似右图的偏二叉树为其Huffman编码树
于是对i=2:n, i的编码为$0^{n-i}1$, 1的编码是0^{n-1}.

第四章 贪心

2. 若在0-1背包问题中, 各物品依重量递增排列时, 其价值恰好降序排列, 对这个特殊的0-1背包问题, 设计一个有效算法找出最优解, 并说明算法的正确性.

解: 设物品1:n按照重量w[1:n]依次递增, c为容量

贪心选择性质: 最优解一定包含物品1

证明: 反证法, 若不包含, 则可用物品1替换任一物品得到更优解.

最优子结构性质:

从最优解中去掉物品1,

它仍是物品2:n和容量c-w[1]的最优解

证明: 反证法, 否则可以替换2:n的选择得到更优解.

算法: 按重量递增排序(O(nlogn)), 依次放入背包, 直到超重(O(n))

第四章 贪心

3. 将最优装载问题的贪心算法推广到2艘船的情形

贪心算法还能产生最优解吗?

解: 不行.

最优装载要求装载件数最多.

其贪心算法是每次选择最轻的物品.

设有物品分别重1,2,3,4,5, 船1容量7, 船2容量8.

若按照最优装载的贪心算法,

船1装1,2,3, 船2装4, 只能装4件物品.

最优解是船1装1,2,4, 船2装3,5.

贪心法

- 勇者斗恶龙

你的王国里有一条n个头的恶龙，你希望雇佣一些骑士把它杀死（即砍掉所有的头）。村里有m个骑士可以雇佣，一个能力值为x的骑士可以砍掉恶龙一个直径不超过x的头，且需要支付x个金币。如果雇佣骑士才能砍掉恶龙的所有的头，且需要支付的金币最少？

注意：一个骑士只能砍一个头（且不能雇佣两次）

【输入格式】

输入包含多组数据。每组数据的第一行为正整数n和m（1≤n,m≤20 000）；以下n行每行为一个整数，即恶龙每个头的直径；以下m行每行为一个整数，即每个骑士的能力。输入结束标志为n=m=0。

【输出格式】

对于每组数据，输出最少花费。如果无解，输出“Loowater isdoomed!”。

【样例输入】

```
2 3
5
4
7
8
4
2 1
5
5
10
0 0
```

【样例输出】

```
11
Loowater is doomed!
```

- 能力强的骑士开价高是合理的，但如果被你派去砍一个很弱的头，就是浪费人才了。
- 因此，可以把雇佣来的骑士按照能力从小到大排序，所有头按照直径从小到大排序，一个一个砍就可以了。
- 当然，不能砍掉“当前需要砍的头”的骑士就不要雇佣了
- 证明;　从资金最少考虑 显然正确。若不这样做可能反而会砍不掉所有头。a[i]能砍 a[i+1]能砍 显然用a[i]，并且a[i+1]可能以后还有更大的发挥空间.　从能将所有头砍掉的角度来看 若a[i]刚好砍掉 那么[1..i]的被舍弃骑士显然也不能砍掉 所以这个角度也是正确的

第五章 回溯

运动员最佳配对问题

问题描述: 羽毛球队有男女运动员各n人. 给定2个n×n矩阵P和Q. P[i][j]是男运动员i与女运动员j配混合双打的男运动员竞赛优势; Q[i][j]是女运动员i与男运动员j配混合双打的女运动员竞赛优势. 由于技术配合和心理状态等各种因素影响, P[i][j]不一定等于Q[j][i]. 男运动员i和女运动员j配对的竞赛优势是P[i][j]*Q[j][i]. 设计一个算法, 计算男女运动员最佳配对法, 使得各组男女双方竞赛优势的总和达到最大.

数据输入: input.txt, 第1行有一个正整数n(1≤n≤20),接下来2n行是P和Q

结果输出: 最佳配对的各组男女双方竞赛优势总和

输入:

```
3
10 2 3
2 3 4
3 4 5
2 2 2
3 5 3
4 5 1
```

输出
52

第五章 回溯

解: 男运动员位置不动, 女运动员全排列, 回溯搜索最优值

解空间是n的全排列, 所以选择排列树作为解空间结构.

变量设计: 当前得分cs, 最佳得分bests, x[1:n]女运动员的排列

定义函数 $f(i,m,x) = \max_{j=m+1}^{n} P[i][x[j]]*Q[x[j]][i]$, 其中i>m,

是在前m位男运动员已配对的情况下, 男运动员i配对其她女运动员的上界

定义函数 Upb(m,x) = f(m+1,m,x)+f(m+2,m,x)+…+f(n,m,x).

当前m位男运动员已配对的情况下, cs+Upb(m,x)是余下情况配对的上界,

由此可以设计剪枝(限制)条件 cs+Upb(m,x) > bests

注1: 有的同学没有设计剪枝条件, 这不能体现回溯的优势.

注2: 有同学使用 cs < bests作为剪枝条件, 这是错误的.

因为可能当前还有很多没有配对, 当所有配对完成后会有更优值.

注3: 也可以设计其它的剪枝条件.

第五章 回溯

初始: 当前得分cs=0, 最佳得分bests=0,

对i=1:n, x[i]=i, 是女运动员的初始排列

backtrack(i)

1. 若 i > n, 返回
2. 对 j = i : n
3. | 交换x[i],x[j], cs+=P[i][x[i]]*Q[x[i]][i],
4. | 若 cs+Upb(m,x) > bests,
5. | | 若cs>bests, 则 bests=cs,
6. | | backtrace(i+1)
6. | cs-=P[i][x[i]]*Q[x[i]][i], 交换x[i],x[j],

主程序执行backtrack(1)即可

	女1	女2	女2
男1	20	6	12
男2	4	15	20
男3	6	12	5

第五章 回溯

- 素数环

输入正整数n，把整数1, 2, 3,…, n组成一个环，使得相邻两个整数之和均为素数。输出

时从整数1开始逆时针排列。同一个环应恰好输出一次。$n \leq 16$。

样例输入：

6

样例输出：

1 4 3 2 5 6

1 6 5 2 3 4

- 由模型不难得到：每个环对应于1～n的一个排列，但排列总数高达16!=2*1013

第六章 分支限界

在解最大团问题的优先队列式分支限界法中, 当前扩展节点满足cn+n-i≥bestn的右儿子节点被插入到优先队列中. 如果将这个条件改为满足cn+n-i>bestn右儿子节点插入优先队列仍能满足算法正确性吗? 为什么?

答: 会影响算法正确性.

改成大于号会漏掉一些最优解, 但是不会降低最优值.

所以如果在回溯时要搜索所有最优解, 则不能改成大于.

如果在回溯时只希望得到最优值和一个最优解可以改成大于.

但对分支限界,如果改成大于会造成不能插入第n+1层节点而不能正确结束.

注: 本教材作者的解答有错误.

这个语句出现在当前扩展节点右分支处, 此时团顶点数上界是cn+n-i.

因为当前n个节点团顶点数是cn, 还有n-i个节点没有考虑.

作者认为此时上界是cn+n-i+1是错误的.

例如,当程序中第一次进循环时, 到这里的团顶点数上界就是n-1.