深圳大学考试答题纸

（以论文、报告等形式考核专用）

二○一六～二○一七　学年度第一学期

课程编号 1500770006　课程名称 算法设计与分析　主讲教师 李荣华　评分

学　　号 2014150120　姓名 洪继耀　专业年级 2014 软件工程 02 班

教师评语：

题目：**资源分配问题-动态规划算法的应用**

算法演示+PPT 讲解成绩	大作业报告成绩

一、问题介绍与要求

题目

资源分配问题：某厂根据计划安排，拟将 m 台相同的设备分配给 n 个车间，各车间获得这种设备后，可以为国家提供盈利 C_{ij}(分配 i 台设备 j 号车间将得到的利润，$1 \le i \le m$，$1 \le j \le n$) 。问如何分配，才能得到最大的盈利？

题目问题的本质

将数量一定的一种或若干种资源合理分配给若干个使用者或地区，使得目标函数最优。

要求

- 设计动态规划算法求解资源分配问题，写出求得最优值的递推公式。
- 测试不同问题规模（按级数增长）的执行结果（即最优分配方案、最优分配方案的值）、运行时间。
- 能够实现的问题规模越大，成绩越高。

二、解决思路

动态规划算法

动态规划是用来解决多阶段决策过程最优化的一种数量方法。其特点在于，**它可以把一个 n 维决策问题变换为几个一维最优化问题**，从而一个一个地去解决。

资源分配问题与动态规划

资源分配问题要求把资源分配给**多个使用者**，本质上来说就是一个**多维（分阶段）决策问题**，系统所处的状态和时刻（即某个时候剩下的资源）是进行决策的重要因素，我们可以运用动态规划来解决分配问题。

此外，资源分配问题满足：

- 最优子结构性质：如果你分配给前 n 的使用者的方案是最优的，那么就能找到分配给前 n+1 个使用者的最优方案。
- 无后效性：分配给前 n 个使用者的最优方案并不会受到后续使用者的影响。
- 子问题重叠性质：每次求下一阶段的最优解的时候总是要访问前几个阶段的情况，因此由很多重叠子问题，可以填表来解决问题。

具体算法

设：

- m 为设备总量，总共有 n 个车间。按车间划分阶段，k=1，2，…，n；
- 决策变量 for_k，表示分配给第 k 个使用者的资源数量；
- 状态变量为，$residue_k$ 表示在第 k 阶段，剩余的可以分给第 k 个至第 n 个使用者的总资源数量；
- 状态转移方程：$residue_{k+1} = residue_k - for_k$，其中 $residue_1 = m$
- 允许决策集合：$D_k\ (residue_k) = \{for_k | 0 \le for_k \le residue_k\}$
- 阶段指标函数(即 G_{ij} 表)：$v_k\ (residue_k\ ,\ for_k) = g_k\ (for_k)$

表示分配给第 k 个车间 for_k 设备时的收益；

则动态规划转移方程为：

$$f_k(residue_k) = max[f_{k-1}(residue_k - for_k) + g_k(for_k)] \quad 其中 for_k \in D_k(residue_k))$$

代码化：

- 我们用状态量表示用 i 台设备分配给前 j 个车间的最大获利，那么显然有：

$f[i][j] = max\{f[k][j-1] + g[i-k][j]\}$，其中 0<=k<=i。

即最大利润分为两部分，一部分是分配 k 台给前 j-1 个车间，分配 i-k 给第 j 个车间

- 依次填表，我们可以得到整张表(程序实现时先枚举车间数，再枚举设备数，再枚举状态转移时用到的设备数，简单 3 重 for 循环语句即可完成)。
- 再用 l[i][j]表示获得最优解时第 j 号车间使用的设备数为 i-l[i][j]，于是从结果倒推往回求即可得到分配方案。

性能分析

- 时间复杂度为$O(n^2 * m)$，空间复杂度为$O(n * m)$，倘若此题只需求最大获利而不必求方案，则状态量可以减少一维，空间复杂度优化为$O(n)$。
- 题目要求尽量做大问题规模，即优化到以下，但是经过思考，由于本次实验使用的是 JAVA 语言，对地址空间控制力不足，故无法做大幅度优化。如需优化必须替换为 C 语言，利用指针来优化，循环使用数组空间。
- 本次实验经测试，在 4GB 内存的机器下，最多做到吗 n*m = 8000*8000 的规模，再多内存将不够用

三、算法的对应代码展示

随机生成利润表算法

代码

```
int[][] g = new int[N][M];      // 利润表
/**
 * 随机生成利润表
 */
for (int i = 0; i < N; i++) {
    g[i][0] = (int) (Math.random() * 30); // 单件利润范围0-30
    for (int j = 1; j < 10; j++) {
        g[i][j] = g[i][j - 1] + (int) (Math.random() * 10); // 后面的利润随机增加0-5
    }
}
```

解释

这里要保证设备增多，利润不减

初始化表

代码

```
/**
 * 初始化表
 */
for (int i = 0; i < m; i++) {
    f[i][0] = 0;                // 将0台设备分配给前i个车间 肯定不产生利润
}
for (int j = 0; j < n; j++) {
    f[0][j] = 0;                // 将j台设备分配给前0个车间 肯定不产生利润
}
```

解释

对 0 的情况进行初始化，防止 JVM 没有给数组初始化为 0 的情况。

填充动态规划状态表 f

代码

```
/**
 * 填表
 */
for (int i = 1; i < m; i++) {
    for (int j = 1; j < n; j++) {
        // 情况1 不分配给i号车间，此时的最大值为将j台设备分配给前i-1个车间的最大值 f[i][j] = f[i-1][j];
        int max = f[i - 1][j];
        l[i][j] = 0;
        // 情况2 分配k台设备给i号车间得g[i-1][k-1]，剩下的j-k台设备分配给前i-1个车间得f[i-1][j-k]
        // f[i][j] = g[i-1][k-1] + f[i-1][j-k]，其中 1≤ k ≤j。
        for (int k = 1; k <= j; k++) {
            if (max < g[i - 1][k - 1] + f[i - 1][j - k]) {
                max = g[i - 1][k - 1] + f[i - 1][j - k];
                l[i][j] = k;
            }
        }
        f[i][j] = max;
    }
}
```

解释

g[i][j] 表示将 j+1 台设备分配给 i+1 号车间得到的利润（为了方便数组索引从 1 开始）

f[i][j] 表示将 j 台设备分配给前 i 个车间的最大利润 （即题目所求的是 p[m][n]）

则 f[i][j]是以下两种情况的较大值：

(1)不分配给当前的 i 号车间，此时的最大值为将 j 台设备分配给前 i-1 个车间的最大值，也就是 f[i][j] = f[i-1][j]；

(2)分配 k 台设备给 i 号车间，剩下的 j-k 台设备分配给前 i-1 个车间，

此时 f[i][j] = g[i-1][k-1] + f[i-1][j-k]，其中 1≤ k ≤j。

也就是递推方程：f[i][j] = max{f[i-1][j], g[i-1][k-1] + f[i-1][j-k]}(1<= k <=j)

初始化 p[0][j] = 0（将 j 台设备分配给前 0 个车间），p[i][0] = 0（将 0 台设备分配给前 i 个车间）

从 i = 0 开始自底向上构造表即可计算出 p[m][n]

而 l[i][j]代表分配给车间 i 的设备

查表求结果

代码

```
// 查表计算解决方案
StringBuilder result = new StringBuilder();
// 从最后一个车间、满设备开始逆推
for (int i = m - 1, j = n - 1; i > 0 && j > 0; ) {
    if (l[i][j] != 0) { // l[i][j]==k表示这个车间分配到k台设备，此时 i--, j-=k
        result.insert(0, "\n" + "车间" + i + " 得到" + l[i][j] + "件设备");
        j -= l[i][j];
        i--;
    } else { // l[i][j]==0表示这个车间没有分配到设备，此时i--
        i--;
    }
}
result.insert(0, "最大利润: " + f[m - 1][n - 1]);
return result.toString();
```

解释

l[N+1][M+1]代表了最后一个车间的分配情况，根据 l 表就可以逆推回去其他车间的分配情况

四、算法举例

假设利润表如下：

	A	B	C	D	E	F	G	H	I	J	K
	利润表g	1	2	3	4	5	6	7	8	9	10
	1	13	21	30	32	34	41	41	47	54	54
	2	26	31	31	34	37	45	54	58	58	65
	3	25	25	32	36	43	49	49	49	55	56

则生成的 f 表如下：

状态表f											
i\j	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	13	21	30	32	34	41	41	47	54	54
2	0	26	39	47	56	61	63	67	72	75	84
3	0	26	51	64	72	81	86	88	93	99	105

举第 2 行第 3 个数 47 来说，它是这么得到的：

2 个车间，3 台设备

首先不分配给车间 2 只分配给 1 的话，能得到 30 利润；

分配给车间 2 号 1 台设备得 26，那么车间 1 得到 2 台设备的最大利润为 21，加起来为 47

分配给车间 2 号 2 台设备得 31，那么车间 1 得到 1 台设备的最大利润为 13，加起来为 44

分配给车间 2 号 3 台设备得 31，那么车间 1 得到 0 台设备的最大利润为 0，加起来为 31

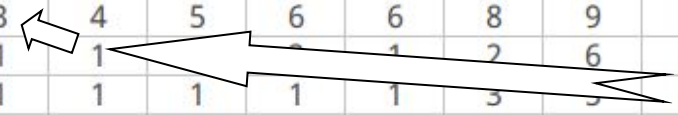
也就是最大利润为 47，此时应该给 I 表的 I[2][3]填 1

生成的 1 表如下：

设备表I											
i\j	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	6	8	9	9
2	0	1	1	1	1	2	2	1	2	6	7
3	0	0	1	1	1	1	1	1	3	5	6

解决方案查表过程：

设备表I											
i\j	0	1	2	3	4	5	6	7	8	9	10
0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	6	8	9	9
2	0	1	1	1	1				2	6	7
3	0	0	1	1	1	1	1	1	3	5	6



得到 1 号车间 3 台，2 号车间 1 台，3 号车间 6 台。

五、代码运行截图与时间测试

```
/usr/lib/jvm/java-8-oracle/bin/java ...
利润表如下：
车间1：[8, 8, 16, 25, 32, 32, 34, 39, 45, 51]
车间2：[25, 32, 34, 35, 38, 42, 51, 54, 58, 61]
车间3：[18, 18, 27, 33, 36, 38, 38, 42, 43, 46]
最大利润方案如下：
最大利润：91
车间1 得到5件设备
车间2 得到2件设备
车间3 得到3件设备
```

问题规模	时间单位：ms				
设备数	100	1000	2000	5000	8000
车间数					
100	47	454	1575	6734	20699
1000	63	3697	10666	50167	138990
2000	95	5642	20387	109258	330397
5000	239	11481	90997	501518	1000299
8000	292	22181	229501	813467	20120218

六、实验结论

算法真奇妙，动态规划真厉害。

通过这次实验，我体会到了实际生产中，算法能帮助我们获取最大化的利益，因此我们要学好算法。

另外一个感悟是，像是 JAVA 这种语言，太高级了，在有些问题上表现不如 C，因为它无法掌控内存，无法从语言底层进行优化，不过好处是写起来方便。