## 实验1. 贪心法求解会场安排问题&基于分治法的循环日程表算法

### 实验内容

本实验要求基于算法设计与分析的一般过程（即待求解问题的描述、算法设计、算法描述、算法正确性证明、算法分析、算法实现与测试），基于贪心法求解会场安排问题以及针对循环日程表算法从实践中理解分治法的思想、求解策略及步骤。

### 实验目的

* 理解贪心法的核心思想以及分治法求解过程；
* 理解分治法的核心思想以及分治法求解过程。

### 环境要求

本实验使用Java语言编写，使用eclipse开发工具。

### 实验结果

1. 贪心法求解会场安排问题

**1．问题的描述**

设有n个会议的集合C={1,2,…,n}，其中每个会议都要求使用同一个资源（如会议室），而在同一时间内只能有一个会议使用该资源。

每个会议i都有要求使用该资源的起始时间bi和结束时间ei，且bi < ei 。如果选择了会议i使用会议室，则其在半开区间[bi, ei)内占用该资源。

如[bi, ei)与[bj, ej)不相交，则称会议i与j是相容的。

会场安排问题要求：在所给的会议集合中选出最大的相容活动子集，亦即尽可能多的选择会议来使用资源。

**2．算法设计**

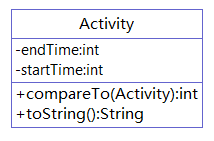
（1）策略

选择最早结束时间且不与已安排会议重叠的会议：

如果选择开始时间最早且使用时间最短的会议，较为理想。而此即结束时间最早的会议。

（2）数据结构

a.会议类



b.排序类

**3．描述算法**

伪代码如下：

Activityselector(actity [] activities)//输入为会议数组activities

SORT (activities) //对activities按照endtime（结束时间）非减排序

A.add(1);//A 存放被安排的会议，首先选择会议1

activities [1].setFlag(true);//设置会议属性flag被选中

int k = 1; // 记录已被选择会议集合中最晚结束的会议

for (int i = 2 i < =n; i++)

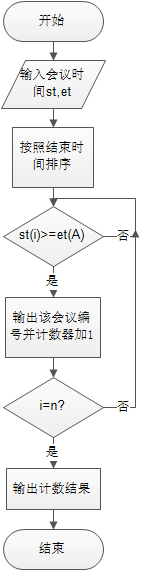
if (activities [i].getStartTime() > activities [k].getEndTime()) // 会议m的开始时间不小于会议k的结束时间

activities [i].setFlag(true);

A.add(i);// 将会议m加入到集合A中

k = i;// 此时集合A中最晚结束的会议为m

算法分析如图所示：



**4．算法的正确性证明。**

（1）最优子结构性质的证明 – 反证法：

令Sij表示在ai结束之后开始，且在aj开始之前结束的那些活动的集合。而Aij为Sij的最大相容子集，且包含活动ak。由于最优解包含ak，可以得到两个子问题：

寻找Sik（ai结束后开始且在ak开始前结束）中的最优解；

寻找Skj（ak结束后开始且在aj开始前结束）中的最优解。

又令Aik = Aij∩Sik和Akj = Aij∩Skj。因此，Aij = Aik∪{ak}∪Akj，|Aij| = |Aik| + |Akj| + 1。

只需证明：Aij必然包含Aik和Akj。

（2）贪心选择性质的证明：

令Sk={ai∈S: si >= fk}为在ak结束之后开始的活动的集合，Ak为贪心选择ak后的最大相容的活动集合：

很显然a1处于原问题最优解中，且可导致原问题的最优解；

假设当ak为原问题求解过程的第j个贪心选择且可导致原问题的整体最优解。原问题的最优解由已选择的活动集Ak和子问题Sk的最优解所组成。据贪心选择策略，第j+1次选择Sk中的第一个活动（假设为am），显然am处于子问题Sk的最优解中，而且可导致原问题的整体最优解。原问题的最优解由已选择的活动集Am和子问题Sm的最优解所组成。

因此，会场安排问题的贪心选择性质得到证明。

**5．算法复杂性分析**

算法复杂性分为两部分。一部分为排序时间复杂性，另一部分费排序代码算法复杂性。排序算法采用冒泡排序算法。

时间复杂性：

(a)：O(n2)，(b)：O(n)。🡺 O(n2)

空间复杂性：

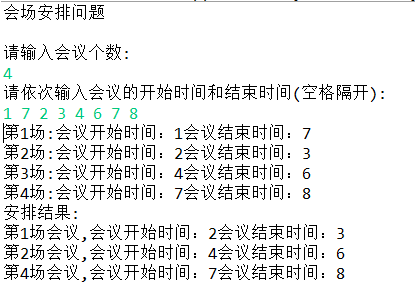
(a)：O(n)，(b)：O(1)。🡺 O(n)

**6．算法实现与测试**

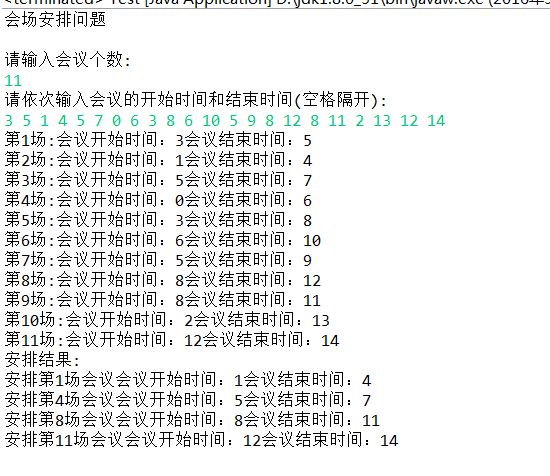
（1）代码以附件形式上交。

（2）程序运行结果如图所示：

测试1：



测试2：



1. 基于分治法的循环日程表算法

**1．问题的描述**

设有n=2k个运动员要进行羽毛球循环赛，现要设计一个满足以下要求的比赛日程表：

每个选手必须与其它n-1个选手各赛一次；

每个选手一天只能比赛一次；

循环赛一共需要进行n-1天。

需要注意的是：

由于n=2k，显然n为偶数。

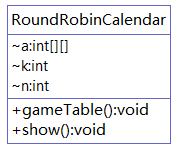
**2．算法设计**

（1）策略

将所有的选手分为两半，n个选手的比赛日程表就可通过为n/2个选手设计的比赛日程表来决定。递归进行分割，直到只剩下2个选手时，比赛日程表的制定就变得很简单。

（2）数据结构

RoundRobinCalendar类



**3．描述算法**

**Table（）**

for i=1 to n

a[1][i] = i;// 设置日程表第一行

m = 1;// 每次填充时，起始填充位置

n1 = n;

for s = 1 to k

n1 /= 2;

for i to n1

for i=m+1 to 2\*m

for j = m+1 to 2\*m

a[i][j + (t - 1) \* m \* 2] = a[i - m][j + (t - 1) \* m \* 2 - m]

a[i][j + (t - 1) \* m \* 2 - m] = a[i - m][j + (t - 1) \* m \* 2]

m \*= 2;

分析过程：

1.输入一个数字n，根据（x&（x-1））==0判断n是否等于2^k。不是则提示重新输入；是则利用换底公式k=(int)(Math.log(n)/Math.log(b))求出k.

2.用一个for循环输出日程表的第一行

for(int i=1;i<=N;i++)

a[1][i] = i；

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

3定义一个m值，m初始化为1，m用来控制每一次填充表格时i（i表示行）和j（j表示列）的起始填充位置。

4.用一个for循环将问题分成几部分，对于k=3，n=8，将问题分成3大部分，第一部分为，根据已经填充的第一行，填写第二行，第二部分为，根据已经填充好的第一部分，填写第三四行，第三部分为，根据已经填充好的前四行，填写最后四行。

for (int s=1;s<=k;s++)

N/=2;

5.用一个for循环对4中提到的每一部分进行划分

for(int t=1;t<=N;t++)

对于第一部分，将其划分为四个小的单元，即对第二行进行如下划分

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

同理，对第二部分（即三四行），划分为两部分，第三部分同理

6.最后，进行每一个单元格的填充。填充原则是：对角线填充

例:由初始化的第一行填充第二行

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | 1 | 4 | 3 | 6 | 5 | 8 | 7 |

进行第二部分的填充

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 | 1 | 4 | 3 | 6 | 5 | 8 | 7 |
| 3 | 4 | 1 | 2 | 7 | 8 | 5 | 6 |
| 4 | 3 | 2 | 1 | 8 | 7 | 6 | 5 |

最后是第三部分的填充

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 |  | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 2 |  | 1 | 4 | 3 | 6 | 5 | 8 | 7 |
| 3 |  | 4 | 1 | 2 | 7 | 8 | 5 | 6 |
| 4 |  | 3 | 2 | 1 | 8 | 7 | 6 | 5 |
| 5 |  | 6 | 7 | 8 | 1 | 2 | 3 | 4 |
| 6 |  | 5 | 8 | 7 | 2 | 1 | 4 | 3 |
| 7 |  | 8 | 5 | 6 | 3 | 4 | 1 | 2 |
| 8 |  | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

**4．算法的正确性证明。**

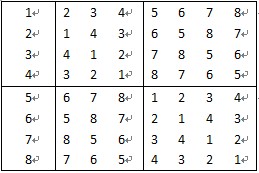
因为分治法证明不像贪心法一样要同时证明两个性质，现在用类推法来证明该算法的正确性。

（1）n=21个选手

直接进行比赛即可，且需要n-1=2-1天完成。如下图所示：

|  |  |
| --- | --- |
| 天数  编号 | 1 |
| 1 | 2 |
| 2 | 1 |

（2）请按此要求将比赛日程表设计成有n行和n-1列的一个表。在表中的第i行，第j列处填入第i个选手在第j天所遇到的选手。其中1≤i≤n，1≤j≤n-1。8个选手的比赛日程表如下图：



按分治策略，我们可以将所有的选手分为两半，则n个选手的比赛日程表可以通过n/2个选手的比赛日程表来决定。递归地用这种一分为二的策略对选手进行划分，直到只剩下两个选手时，比赛日程表的制定就变得很简单。这时只要让这两个选手进行比赛就可以了。如上图，所列出的正方形表是8个选手的比赛日程表。其中左上角与左下角的两小块分别为选手1至选手4和选手5至选手8前3天的比赛日程。据此，将左上角小块中的所有数字按其相对位置抄到右下角，又将左下角小块中的所有数字按其相对位置抄到右上角，这样我们就分别安排好了选手1至选手4和选手5至选手8在后4天的比赛日程。依此思想容易将这个比赛日程表推广到具有任意多个选手的情形。

因此，算法的正确性得证。

**5．算法复杂性分析**

假设有n=2k个参赛者。

1. K=1时，安排两个选手比赛需要常量时间，因为T(n)=O(1)。
2. 当k≥1时，计算出n值需要进行for循环，需要常量时间为O(1)，用for循环输出日程表的第一行所需时间也是O(1)。用三重for循环将其不断分解规模为n/2的子问题，所需时间为T(n/2)。

综上可以看出T(n)=T(n/2)+f(n) ，f(n)为给表格左右复制的时间。f(n)=(n/2)2。推出T(n)=T(n/2)+(n/2)2。

N规模的问题做logN次f(n)

T(n)属于O（∑O((n/(2k))^2)） 1<=k<log(n) 也就是T(n)∈O(n2)

因为过程中使用到了二维数组，所以其空间复杂性为O(n2)。

**6．算法实现与测试**

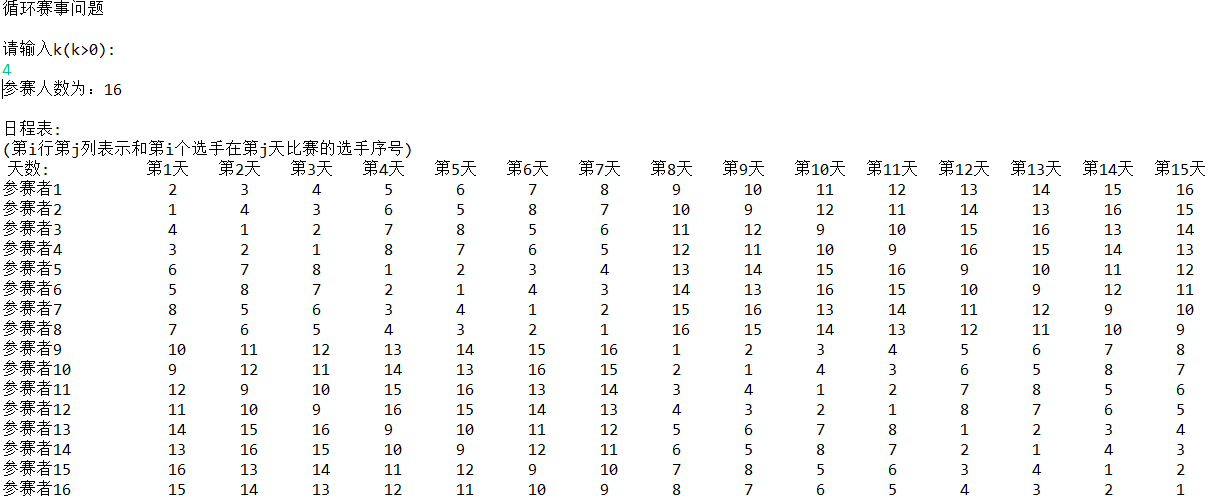
（1）代码以附件形式上交。

（2）程序运行结果如图所示：

测试1：



测试2：



### 实验总结

1. 贪心法精髓是“今朝有酒今朝醉”。每个阶段的决策一旦做出就不可更改。不允许回溯。虽不能对所有问题都得到整体最优解，但对许多问题能产生整体最优解。如单源最短路经问题，最小生成树问题等。在一些情况下，即使贪心算法不能得到整体最优解，其最终结果却是最优解的很好近似。
2. 对于一个问题能否使用贪心法解决，应该分析问题是否存在两个性质：贪心选择性质和最优子结构性质。就会场安排问题而言，它的确符合两个性质。
3. 本次实验，我采用面向对象编程的思想，创建会议类，会议包括属性开始时间和结束时间以及是否被安排的标志。会议类要实现Compare接口，然后对会议类进行排序。排序完成之后，从重排序后数组的第一个元素入手，如果之后的开始时间大于等于其结束时间，就安排该会议使用会场，然后下一个会议的比较对象就变成了最后一个安排好的会议的结束时间。
4. 分治法的基本思想是将一个难以直接解决的大问题，分割成一些规模较小的相同问题，以便各个击破，分而治之。首先将问题分解为k个子问题（与最好规模相同）并分别求解。如子问题规模仍不够小，则再划分为k个子问题。如此递归进行，直到问题规模足够小易求解为止。
5. 对于一个问题是否能用分治法解决，应该分析问题是否存在几个特征：问题的规模缩小到一定程度就可以容易解决、问题可以分解为若干个规模较小的相同子问题、问题所分解出的各个子问题是相互独立的、问题分解出的子问题的解可以合并为原问题的解。
6. 根据分治法的思想，循环赛日程表将本问题进行了由小规模到大规模的求解设计，程序设计的关键点在于如何对问题进行划分和填充公式的归纳。填充原则就是对角线填充，就是左上、左下、右上、右下的单元格的对应赋值在划分时，主要运用了两个for循环；在填充时，运用了两个for循环。通过这次程序设计，加深了对分治算法的认识。

算法设计与分析的过程：

