运筹学通论I

胡晓东

应用数学研究所

中国科学院数学与系统科学研究院

http://www.amt.ac.cn/member/huxiaodong/index.html

5. 组合优化 - 算法设计技巧

精确算法

分而治之 (搜索、排大小序、旅行商)
动态规划 (最短路、三角剖分、背包)
分支定界 (整数线性规划、旅行商、工件排序)

近似算法或启发式算法

贪婪策略 (最小生成树、最大可满足、背包、
顶点覆盖、独立集、旅行商)
局部搜索(最大匹配、旅行商、最大割)
序贯法 (工件排序、装箱、顶点着色)
整数规划法 (顶点覆盖、最大可满足、最大割)
随机方法 (最小割、最大可满足、顶点覆盖)
在线算法 (页面调度、***k***-服务器、工件排序、装箱)
不可近似 (最大团、背包、旅行商、装箱、连通控制集等)

5.1 分而治之

算法设计与分析中的一个简单而重要的方法是**分而治之**技巧。

5.1 分而治之（续一）

分而治之方法主要包含以下三个步骤：

- 将问题分解成若干子问题
- (用递归的方法)分别求解所产生的子问题
- 将求得的各个子问题的解拼成原问题的解

引理 设 a, b, 和 c 是非负实数, $n = c^k$， k是一个自然数。则满足递归等式：

$$\begin{cases} T(n) = b, & n = 1; \\ T(n) = aT(n/c) + bn, & n > 1, \end{cases}$$

的解为

$$T(n)=\begin{cases} O(n), & \text{当 } a < c, \\ O(n\log n), & \text{当 } a = c, \\ O(n^{\log_c a}), & \text{当 } a > c. \end{cases}$$

5.1 分而治之 - 求最大与最小

我参加了北京奥运会的志愿者工作。竞赛部交给我的第一项任务是设计一个计算机程序，用它来在有些体育比赛项目(比如跳水、体操)评分时，统计一个运动员/队的得分：需要把 ***n*** 个裁判给出的最高分和最低分去掉，然后把剩余的分数相加求平均值。

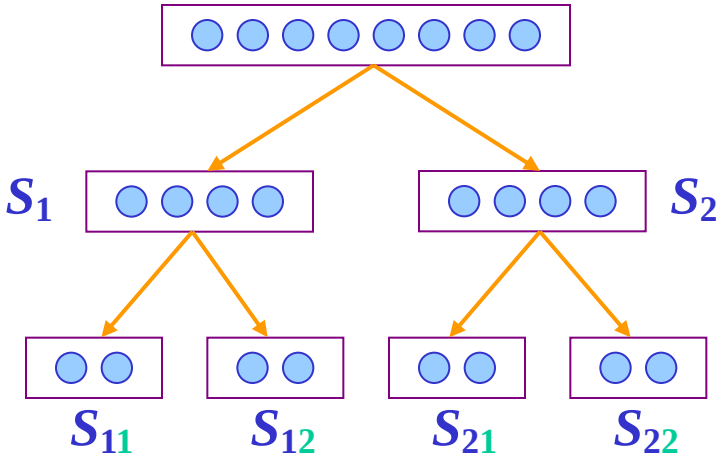
我可以先在 ***n*** 个分数中，用**两两比较**的方法，找出**最大**的分数；然后再在剩下的 ***n*** **– 1**个数中，用**两两比较**的方法，找出**最小**的分数。这样我总共用**2*n* – 3** 次比较就可以找到最大和最小的分数了。

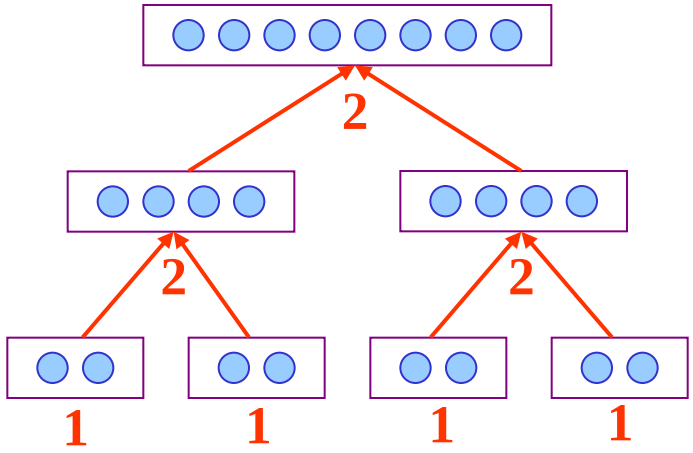
我可不可以不用这么多次呢？可以！

5.1 分而治之 - 求最大与最小（续一）

我们可以先把 n 个数分成(尽量)相当的两部分S_1 和 S_2，比如，每部分包含 $n/2$ 个数。如果我们已经知道了S_1 和 S_2 中最大和最小的数了，那么我们就可以很容易的确定出n 个数中最大和最小的数了：比较一下两个集合中最大的数，再比较一下两个集合中最小的数，即可。

设计算法时自上而下想





执行算法时自下而上算

5.1 分而治之 - 求最大与最小（续二）

定理 1 任意给 n 个数，算法 **MaxMin** 用 $3n/2-2$ 次比较就可以找到最大和最小的数。

证明： 假设算法需要$T(n)$ 次比较。显然，$T(2)=1$。当 $n>2$时，我们有递推公式：$T(n)=2T(n/2)+2$。用数学归纳法，即可证明$T(n)=3n/2-2$。

定理 2 任意给 n 个数，任意一个算法都需要用至少 $3n/2-2$ 次比较才可以找到最大和最小的数。

证明： 略(比较复杂)。

5.1 分而治之 - 依大小排序

奥组委交给我的第二项任务是设计一个计算机程序，用它来排金牌榜：需要把 n 个国家的金牌数，$a_1, a_2, \ldots, a_n$，从多到少排成一列，$a_1' \geq a_2' \geq \ldots \geq a_n'$。

一个简单的方法是：首先通过两两比较大小，在 n 个数中确定 a_1'；然后，再通过两两比较大小在剩下的 n 个数中确定 a_2'；重复操作，直到所有数都依大小排好序。

很显然此方法总共需要 $n(n-1)/2$ 次两两比较。能不能用更少的两两比较次数，就可以把 n 个数依大小排好序呢？

北京奥运会奖牌榜

截至8月20日

					总计
1	中国	45	14	20	79
2	美国	26	27	28	81
3	英国	16	10	11	37
4	俄罗斯	13	14	18	45
5	澳大利亚	11	12	13	36
6	德国	11	8	9	28
7	韩国	8	10	6	24
8	日本	8	6	9	23
9	意大利	6	7	7	20
10	乌克兰	5	5	8	18
11	法国	4	12	14	30
12	荷兰	4	5	4	13
13	牙买加	4	3		7
14	罗马尼亚	4	1	3	8
15	西班牙	3	5	2	10
16	波兰	3	4	1	8
17	新西兰	3	1	5	9
18	斯洛伐克	3	1		4
19	加拿大	2	6	5	13
20	肯尼亚	2	4	2	8

排名前20名

林汉志 编制 新华社发

5.1 分而治之 - 依大小排序（续一）

分而治之方法如下：首先将个数平均分成两部分；然后，分别把每一部分依大小排好顺序；最后，把这两个序(列)合并成一个依大小排好的序(列)。

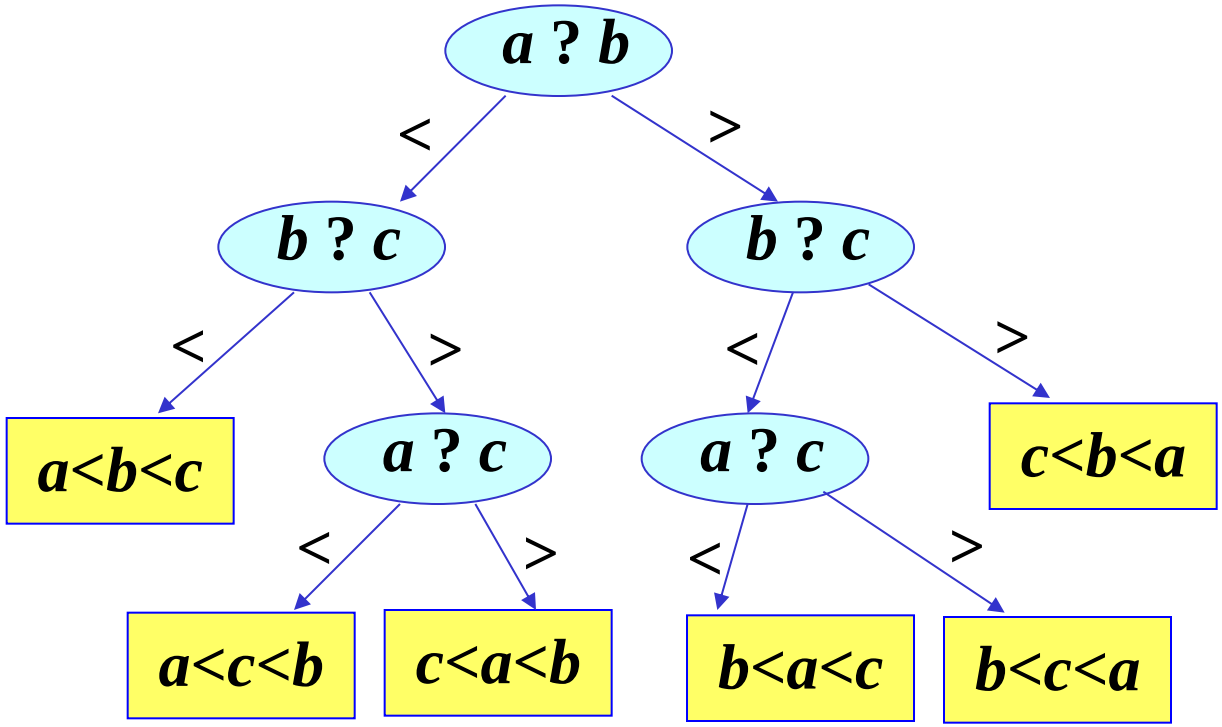
那么，这个方法的关键是，如何用尽量少的比较次数将两个已经依大小排好顺的序(列) S_1 和 S_2 合并成一个序(列)。

我们可以这样做：分别取 S_1 和 S_2中最大的，两两比较，大的那个数就是a_1'；把它从 S_1 或者 S_2中取出，再对剩下的两个序(列)进行同样的操作，即可确定a_2'；重复此过程即可把这两个序(列)合并成一个依大小排好的序(列)。

练习题：证明上述方法需要$O(n\log n)$ 次比较。

5.1 分而治之 - 依大小排序（续二）

我们下面说明上述算法是最优的，亦即没有算法可以用更少次数的比较将任意个数依大到小排序。为此我们引进一个**判定树**，用它可以刻画任意一个排序算法。考虑一个简单的例子：将三个不同的数***a, b,*** 和 ***c*** 依大到小排序。



5.1 分而治之 - 依大小排序（续三）

定理 3 将任意n 个数依大到小排序，每个算法都至少需要 $cn \log n$ 次比较，这里 $c > 0$，且 n 足够大。

证明. 考虑任意一个排序算法和它的一个二叉树表示。显然，树上的每一个叶子节点，都对应了n 个数的一个可能排序结果，而且，一共有 $n!$ 个叶子节点。另外，注意到，从树的根子节点到叶子节点的路长就等于确定相应排序结果所需要的比较次数。因为 $n > 1$，所以有

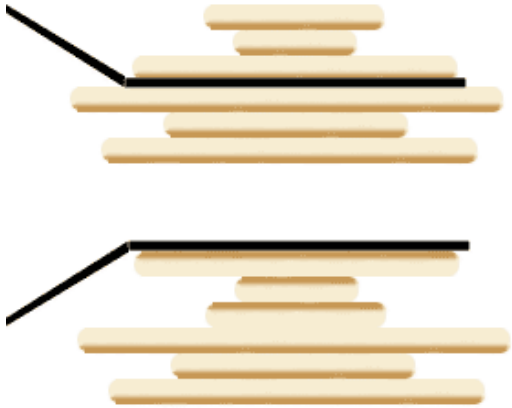
$$n! \geq n(n-1)\,(n-2)\ldots(\lceil n/2 \rceil) \geq (n/2)^{n/2}\,,$$

由此可得，当时$n \geq 4$ 时，有

$$\log n! \geq (n/2)\log(n/2) \geq (n/4)\log n\,。$$

5.1分而治之 - 煎饼依大小排序

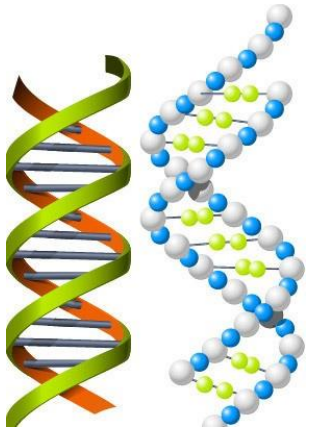
美国几何学家**Jacob E. Goodman**曾提出一个有趣的问题：仅用锅铲将 ***n*** 个煎饼按照大小依次排序，最少需要多少次翻转呢？易知，将最大的煎饼放在最下面最多需要**2**次，所以最多需要 **2*n*-3**次。



5.1 分而治之 - 煎饼依大小排序（续一）

1975年在哈佛念书的**比尔·盖茨**和他的导师发表了一篇文章：Gates, W. and Papadimitriou, C. “Bounds for Sorting by Prefix Reversal.” Discrete Mathematics. 27, 47~57, 1979。给出了一种方法，仅需要 **5*n*/3**次。你能比盖茨做得更好吗？

目前，人们所知道的最小次数介于 **15*n*/14**和**18*n*/11**。另外，人们发现该问题与**DNA**序列变换有关。



5.1 确定第 k 小的元素

现在我们考虑另外一个著名的搜索问题：在含有 n 个元素的集合 S 中找第 k 小的元素。一个简单的方法就是，将这个元素从小到大排一个顺序 (例如可以采用**混并-搜索算法**)，然后确定出排第 k 个位置的元素。根据**定理 3** 可知，这个搜索方法需要 $O(n\log n)$ 次比较运算。

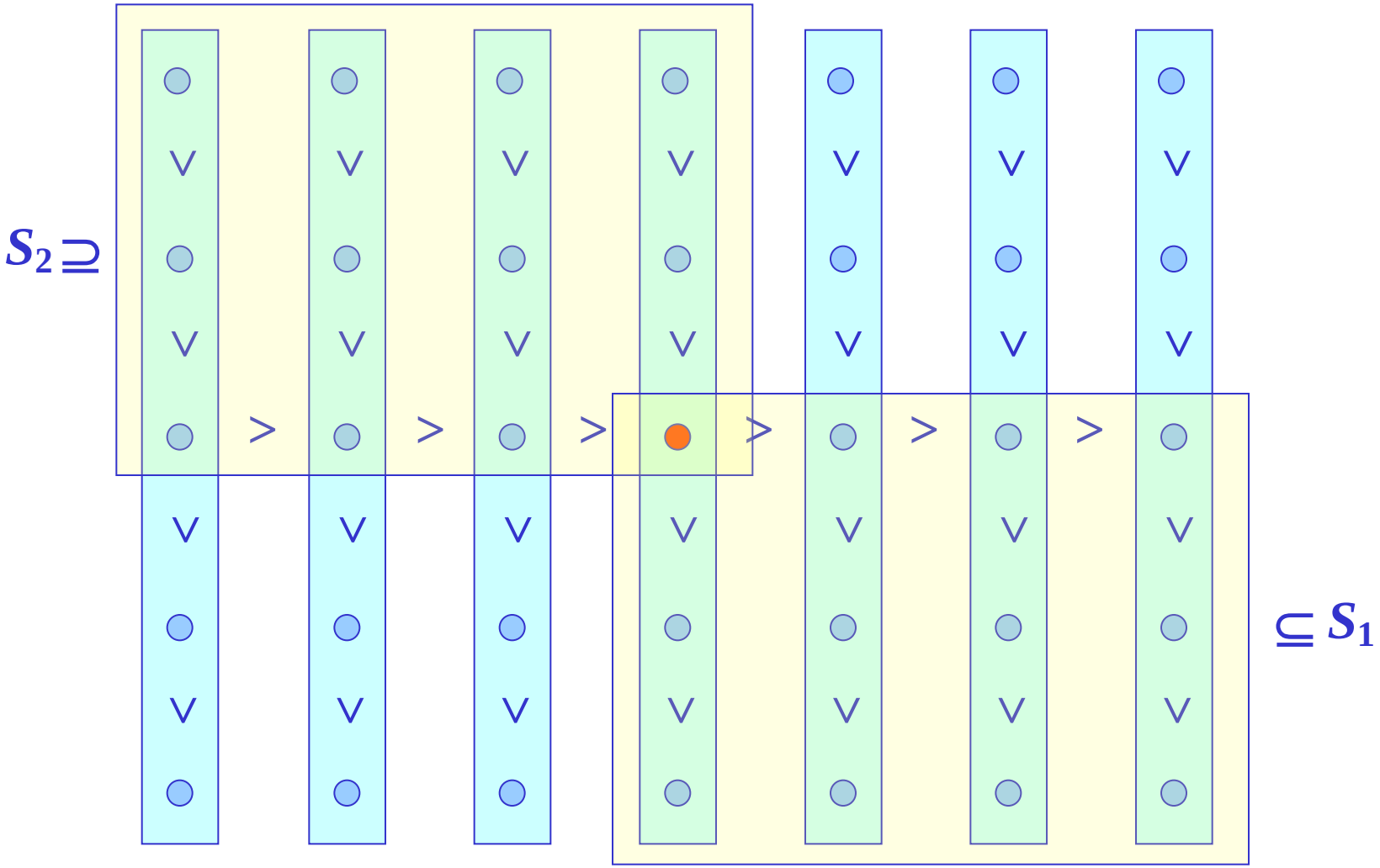
下面我们说明如何采用更加精细的分而治之策略，通过 $O(n)$ 次比较运算确定第 k 小的元素。我们的基本想法是，首先将给定的集合划分为三个子集合 S_1, S_2, S_3 使得，S_1 为所有小于 m 的元素集合，S_2 为所有等于 m 的元素集合，而S_3 所有大于 m 的元素的集合。然后，计算出集合 S_1 和 S_2 所含有的元素的个数，即可确定第k 小的元素究竟是在集合 S_1中，还是在 S_2，或者 S_3 中。通过这个方法，我们就可以将原问题转化为一个小的子问题了。

5.1 确定第 k 小的元素（续一）

为了设计这样的一个线性时间算法，我们必须找到一个适当的集合划分方法，使得集合 S_1 和 S_3 每一个所含有的元素个数不超过 S 所含有元素的个数的一个常数倍。

算法的关键在于如何选取划分数 m。我们将集合 S 划分为若干个小的集合，每一个集合含有5 个元素（最多有一个含有不到 5 个元素）。将每一个小集合依照大小排一个顺序，并选出中间的一个数。所有小集合的中间数构成一个集合 M，它仅含有 $n/5$ 个元素。我们可以求出 M 的中间数。而所用的时间是求 n 个元素的中间数所用时间的1/5。

5.1 确定第 k 小的元素（续二）



5.1 确定第 k 小的元素（续三）

```
procedure Select(k, S)
if |S| < 50 then
   sort S;
   return the k-th smallest element in S.
else
     divide S into ⌊|S|/5⌋ 5-element sequences;
     sort each 5-element sequences.
     M ← the sequence of medians of 5-element sequences.
     m ← Select(⌈|M|/2⌉, M).
     S1 := {s ∈ S | s < m};
     S2 := {s ∈ S | s = m};
     S3 := {s ∈ S | s > m};
     if |S1| ≥ k then
          return Select(k, S1);
     else if |S1| + |S2| ≥ k then
               return m.
               else return Select(k − |S1| − |S2|, S3).
```

5.1 确定第 k 小的元素（续四）

定理 5 算法 Select 可以在$O(n)$ 时间内确定个 n 元素中第 k 小的元素。

证明. 注意，集合 M 最多含有 $n/5$ 个元素，因而最多调用 **Select**($\lceil |M|/2 \rceil$, M) 算法 $T(n/5)$ 次。

此外， S 中至少有$1/4$ 个元素小于或者等于 m，且至少有 $1/4$ 个元素大于或者等于 m。下面我们证明集合 S_1 和 S_3 每一个最多含有 $3n/4$ 个元素。因此，最多递归调用 **Select**(k, S_1) 算法或者 **Select**($k-|S_1|-|S_2|$， $|S_3|$) 算法 $T(3n/4)$ 次。同时注意到，其他运算最多需要 $O(n)$ 时间。因而，对于任意 $n > 5$ 有

$$T(n) \le T(n/5) + T(3n/4) + cn,$$

由此可得, $T(n) \le 20cn$。

5.1 确定第 k 小的元素（续五）

应用判定树模型，我们同样可以证明 Select 是最好的的算法：其时间复杂度与最优算法的复杂度仅差一个常数倍。 考虑确定集合 S 中的第 k 小的元素的任意一个算法，设其判定树为 T 。注意， T 中的每一条路 P 都定义了集合 S 上的这样一个关系 R_P：$a_i R_P a_j$ 如果两个不同元素 a_i 和 a_j 在路 P 上的某一个节点处进行了比较，且结果为 $a_i < a_j$。 此外，设 R_p^*为 R_p 的传递闭包使得，aR_P^*b 当且仅当存在$e_1 R_P e_2, e_2 R_P e_3, \ldots, e_{m-1} R_p e_m$，其中 $a = e_1$ 和 $b = e_m$。直观上可以看出，若 $a_i R_P^* a_j$，则与路 P 相关联的一系列比较可以确定 $a_i < a_j$。

引理 2 若由路 P 可确定元素a_m 是S 集合中第 k 小的元素，则对每一个 $i \neq m$，$1 \le i \le n$，或者有 $a_i R_P^* a_m$ 或者有 $a_m R_P^* a_i$。

5.1 确定第 k 小的元素（续六）

证明. 采用反证法。假设，存在一个元素 a_u 它与 a_m 不存在关系 R_P^*。我们要证明，在集合 S 中将 a_u 放在 a_m 的前面或者a_m 的后面，可以导致与假设相矛盾的结论：路 P 可以确定 a_m 是集合S 中第 k 小的元素。

令 $S_1=\{a_j \mid a_j\, R_P^*\, a_u\}$， $S_2=\{a_j \mid a_u R_P^*\, a_j\}$，另令$S_3$ 为 S 中其他元素的集合。根据假设，a_u 和 a_m 都属于集合 S_3 。

如果 a_j 属于集合 S_1 (或者 S_2) 且 $a_l R_P^* a_j$ (或者 $a_j\, R_P^* a_l$)，那么由传递性， a_l 也属于S_1 (或者 S_2)。这样我们就可以构造一个线性序关系 R，它与 R_P^* 是兼容的使得，属于S_1 的所有元素都在 S_3 中的元素的前面，而后者都在 S_2 中的元素前面。

5.1 确定第 k 小的元素（续七）

证明. (续) 根据假设，元素 a_u 与集合S_3 中的任何元素都不满足关系 R_P^*。下面我们考虑两种情形。

情形 1. $a_u R a_m$。此时，我们可以找到一个新的线性序关系R’，它与 R 几乎是一样的，除了 a_u 被恰排在 a_m 后面。R’ 与 R_P 也是兼容的。对 R 和 R^*中的任意一个，我们都可以为 a’ 找到不同的整数使得，它们分别或者满足 R 或者满足 R’。但是，元素 a_m 不可能在两个序关系中都是第 k 小的元素，这是因为 a_m 在R’中的位置要比 R 在中的位置靠后一位。由此可知，如果 S 中的某个元素与 a_m 不存在关系 R_P^*，那么 T 就不可能正确地确定集合 S 中的第 k 小的元素。

情形 2. $a_m R a_u$。此时可以用同样的论证对称地分析。

5.1 确定第 k 小的元素（续八）

定理 6 任意一个可确定 n 个元素中第 k 小的元素的算法都至少需要进行 $n-1$ 次比较运算。

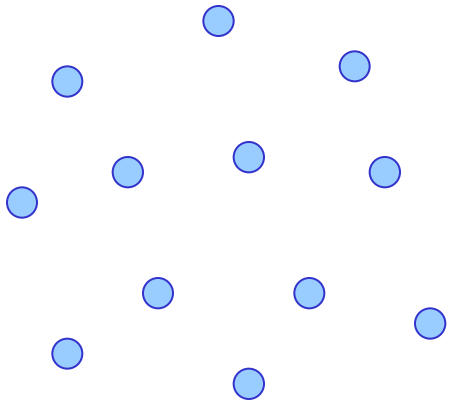
证明. 考虑可确定集合 S 中 第 k 小的元素的任一算法，设 T 为其判定树，P 为 T 上的一条路， 而 a_m 是 S 中第 k 小的元素。对于每一个 a_i，$i \neq m$， 定义 a_i 的**关键比较**为在 P 上与 a_i 有关的第一个满足以下某个条件的比较：

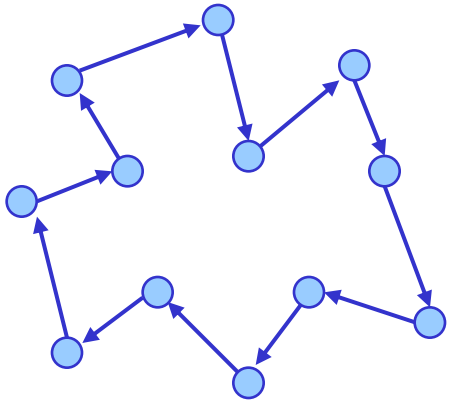
1. a_i 与 a_m 比较，
2. a_i 与 a_j 比较， $a_i R_P a_j$ 且 $a_j R_P^* a_m$，
3. a_i 与a_j 比较， $a_j R_P a_i$, 且 $a_m R_P^* a_j$，

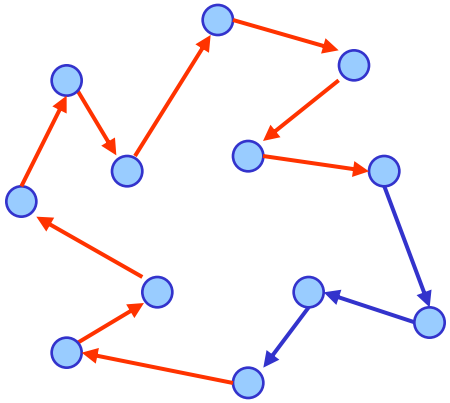
显然，除了元素a_m 以外，每一个 a_i 都有一个关键比较。而且，没有一个比较是两个相互比较的元素的关键比较。因此，路 P 的长度至少是 $n-1$ 因为 $n-1$ 个元素有关键比较。

5.1 旅行商问题

我们下面考虑欧氏平面上的**旅行商问题** (简记：**E-TSP**)。一个旅行商需要在 ***n*** 个城市推销其公司的产品。他从公司所在城市出发，然后访问每一个城市一次，最后回来。他面临的问题是，如何依次访问这些城市，使得旅行的距离最短。这里我们不妨假设，每一个城市可以视为平面上的一个点，两座城市之间的距离等于相应点的欧氏距离。





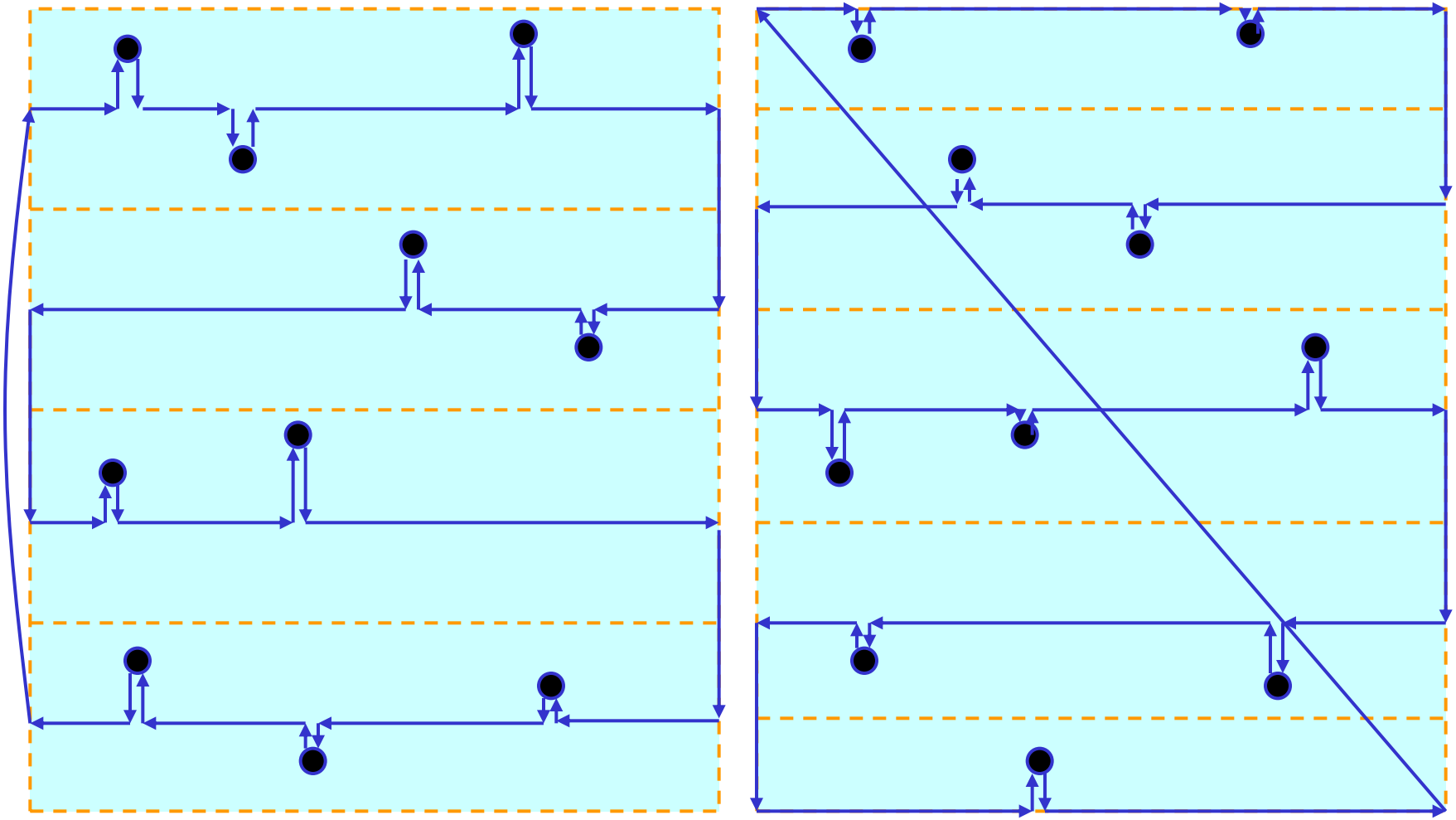


5.1 旅行商问题（续一）

条化方法就是将一个正方形区域划分成若个几何上全等的条块（长方形），然后在每一个条块内构造一条路。具体步骤如下：

- 将正方形划分成高度为 $\sqrt{n}/\delta$ 的水平条块。
- 从最上面的条块的左侧开始，在其中间水平线上向右旅行。若走到了某一个城市点的正下方或者正上方，则直接访问该点，然后回到其下方或者上方的中间线位置。用同样的方式，访问所有的城市点和条块。
- 最后从最下面的条块的最左侧或者右侧回到起点，即最上面的条块的左侧。

5.1 旅行商问题（续二）



(a)　　(b)

5.1 旅行商问题（续三）

定理 7 求解单位正方形上旅行商问题的条化方法可以找到一条回路其长度不超过最短回路的 $2\sqrt{n} + \sqrt{2} + 1$ 倍。

证明. 用 L_{SM} 记条化方法求得的回路的长度，并用 α_h (α_v) 记在每一个条块内沿水平（或者垂直）方向走过的距离。令用 β 表示从右下角回到左上角走过的距离，用 γ 表示从中间线走到城市点并返回的距离。易知，我们有

$$\begin{aligned} L_{SM} &\le \alpha_h \sqrt{n}/\delta \quad + \alpha_v + \gamma n + \beta \\ &\le \sqrt{n}/\beta \ +1 + \ n2(\delta/2\sqrt{n}) + \sqrt{2} \\ &= \sqrt{n}(1/\delta + \delta) + 1 + \sqrt{2} \end{aligned}$$

取 $\delta = 1$，即得 L_{SM} 的一个上界。

5.1 旅行商问题（续四）

我们可以将**条化方法**做如下的改进：在将单位正方形划分完以后，将条块的中间线视为新的边界线，然后以类似地方式，沿着条块的边界线（而不是中间线）行走。图(b) 给出了这种不同的行走方式。我们可以从两种不同方式得到的两条回路中，选取一条短的作为条化方法的解。

定理 8 求单位正方形上旅行商问题的改进条化方法可以给出一条回路，其长度不超过最短回路的长度的 $\sqrt{(2n)} + \sqrt{2}+1$倍。

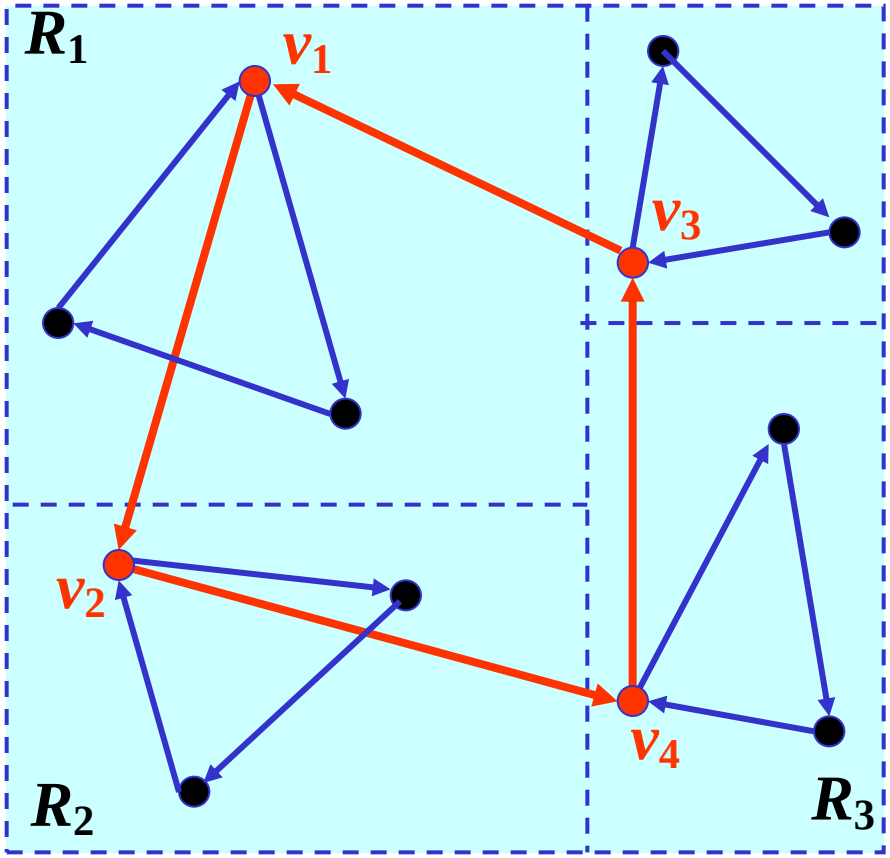
证明. 我们可以采用与**定理 7** 的证明几乎一样的论证和分析。惟一不同之处是，走到每一个城市点并返回的距离之和仍然是 $\delta/(2\sqrt{n})$。

5.1 旅行商问题（续五）

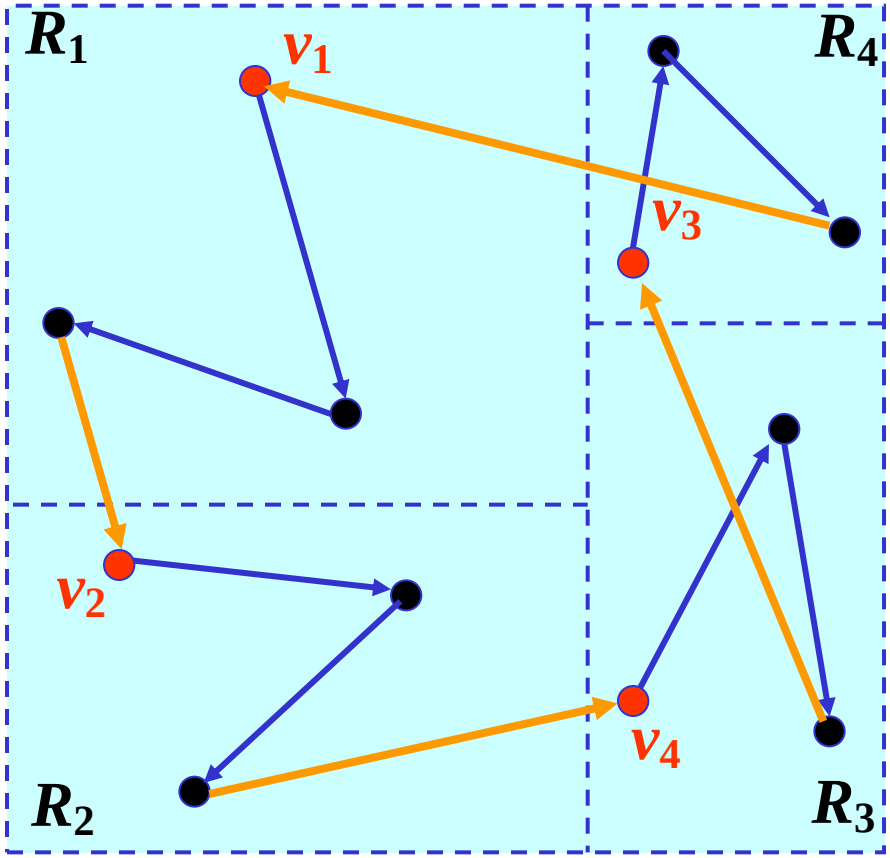
下面我们介绍卡普提出的一个**划分算法**。与前面介绍的**条化方法**类似，它也是采用等划分策略，只是它首先将一个单位正方形 划分为 $\sqrt{(n/k)}$ 个垂直条块使得每一个条块含有 $\sqrt{(nk)}$ 个给定的点，然后再将每一个条块划分为$\sqrt{(n/k)}$ 个长方形使得每一个长方形含有 k 个给定的点。经过这样的划分，单位正方形就被划分为许多个条块（和长方形），每一个含有相同数目的给定点，但是它们的面积不一定相等。

将划分完以后，对每一个R_i，我们求出 R_i 中个 k 点的最优回路，然后，选取一个点 $v_i \in R_i$，并用**条化方法**求出经过 n/k 个选出的点的一条回路；最后，通过选取捷径，得到一条回路T_{PA}。

5.1 旅行商问题（续六）



R_i 中的**最优回路**
经过所有 v_i 的**一条回路**



走捷径得到**一条回路**

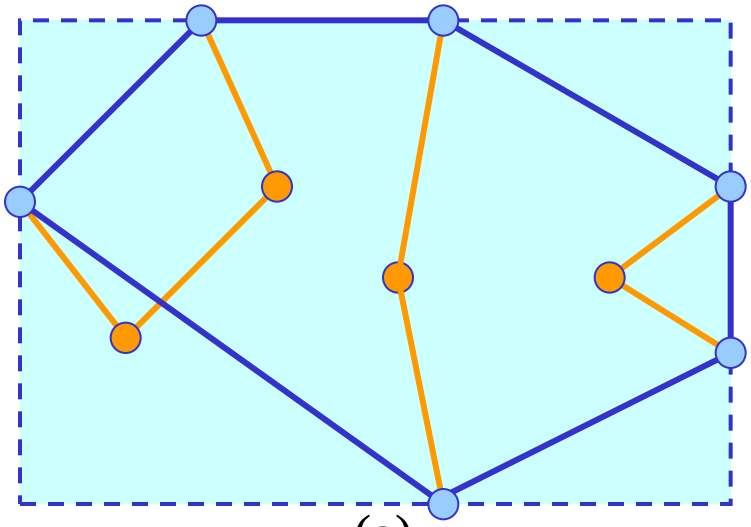
5.1 旅行商问题（续七）

引理 3 设T_{opt} 为平面上给定 n 个点的最优回路，L_i 为T_{opt} 在 R_i 中的边长之和。另设 L_i^* 为 R_i 中的最优回路的长度。则有

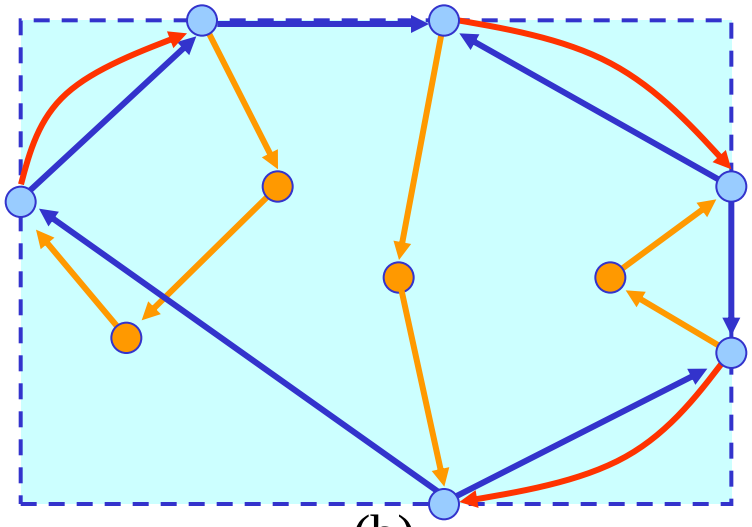
$$L_i \le L_i^* + 3P(R_i)/2.$$

证明. 对每一条跨过 R_i 边界的边，我们定义一个新的边界点。注意，总共有偶数多个边界点。我们将所有的边界点连接起来，形成一个圈 (图(a))，其长度不会超过 $P(R_i)$。 然后，我们添加一些边 (图(b)) 构成一个匹配使得其所含的边长不超过 $P(R_i)$ 的一半。这样就构造了一个欧拉图 (图(b))。最后， 再选取一系列捷径，我们就可以构造一条包含内部点和边界点的回路 (图(c))。我们还可以将这条捷径进一步缩短，使其仅包含内部点 (图(d))。

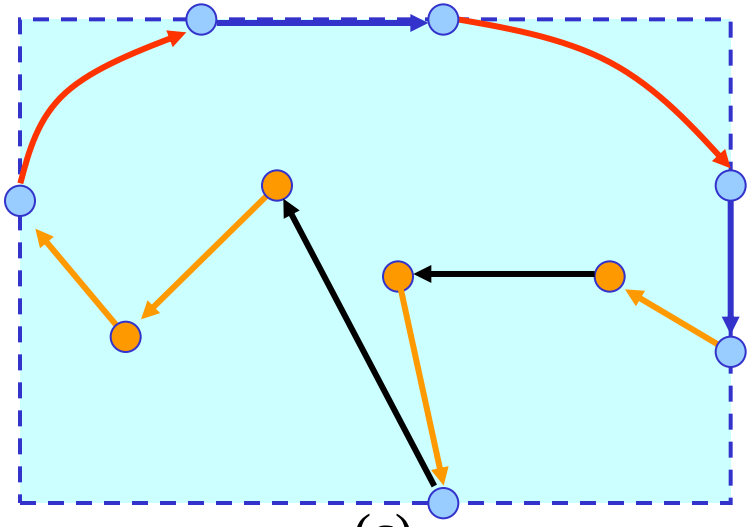
5.1 旅行商问题（续八）



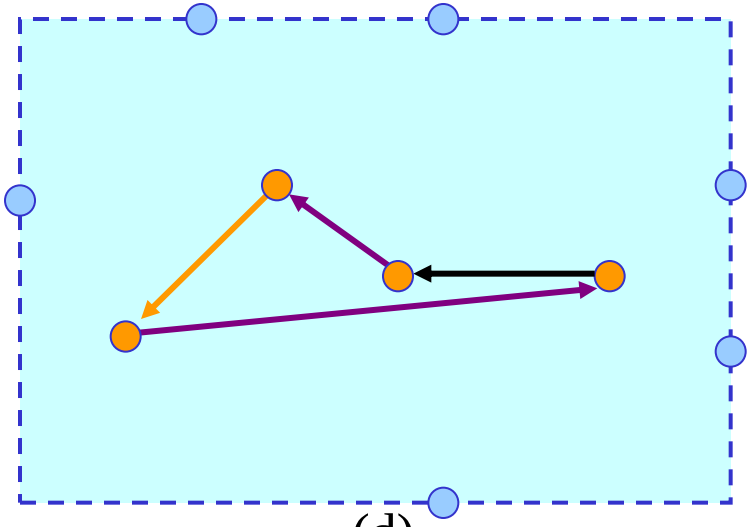
(a)



(b)



(c)



(d)

5.1 旅行商问题（续九）

定理 9 设 L_{opt} 为平面上给定 n 个点的最优回路的长度，而 L_{PA} 为划分算法所生成的回路的长度。则 $L_{PA} \le L_{opt} + O(\sqrt{n/k})$。

证明. 依据**定理 8** 和**引理 2**，可得

$$
\begin{aligned}
L_{PA} &\le \sum_{i=1}^{n/k} L_i^* + L_{SM} \\
&\le \sum_{i=1}^{n/k} \left(L_i + \frac{3}{2}P(R_i)\right) + (2n/k)^{1/2} + O(1) \\
&= L_{opt} + \frac{3}{2}\left(2(n/k)^{1/2} + 2(n/k)^{1/2}\right) + (2n/k)^{1/2} + O(1) \\
&= L_{opt} + O\left((n/k)^{1/2}\right)
\end{aligned}
$$

复习题

请在1-60中间想好一个自然数，比如你的年龄，然后告诉我，它在下面六个表中的哪几个表中出现了，我可以猜出它是多少。请问我是怎么做到的呢？

1	3	5	7	9	11
13	15	17	19	21	23
25	27	29	31	33	35
37	39	41	43	45	47
49	51	53	55	57	59

2	3	6	7	10	11
14	15	18	19	22	23
26	27	30	31	34	35
38	39	42	43	46	47
50	51	54	55	58	59

4	5	6	7	12	13
14	15	20	21	22	23
28	29	30	31	36	37
38	39	44	45	46	47
52	53	54	55	60	13

8	9	10	11	12	13
14	15	24	25	26	27
28	29	30	31	40	41
42	43	44	45	46	47
56	57	58	59	60	13

16	17	18	19	20	21
22	23	24	25	26	27
28	29	30	31	48	49
50	51	52	53	54	55
56	57	58	59	60	31

32	33	34	35	36	37
38	39	40	41	42	43
44	45	46	47	48	49
50	51	52	53	54	55
56	57	58	59	60	46