算法概论

习题解答

薛健

工程科学学院

Last Modified: 2020.1.4

UCAS

主要内容

主要内容

1 习题解答

- 数组循环移位
- 递归程序设计
- 多米诺骨牌问题的分治算法
- 元素平均移动次数
- 5 元素中位数和排序
- 多米诺骨牌问题的动态规划算法
- 邮局位置问题
- 凝聚图和转置图

主要内容

1 习题解答

- 数组循环移位
- 递归程序设计
- 多米诺骨牌问题的分治算法
- 元素平均移动次数
- 5 元素中位数和排序
- 多米诺骨牌问题的动态规划算法
- 邮局位置问题
- 凝聚图和转置图

问题描述及简单算法

Exercise (1)

已知一个长度为 n 的数组和一个正整数 k，并且最多只能使用一个用于交换的附加空间单元，试设计算法得到原数组循环右移 k 次的结果并分析算法的时间复杂度。

问题描述及简单算法

Exercise (1)

已知一个长度为 n 的数组和一个正整数 k，并且最多只能使用一个用于交换的附加空间单元，试设计算法得到原数组循环右移 k 次的结果并分析算法的时间复杂度。

- 一般算法设计：

Algorithm `ShiftRight`($A[], n, k$)

```
k ← k mod n;
for i ← 1 to k do
    temp ← A[n − 1];
    for j ← n − 1 to 1 do
        A[j] ← A[j − 1];
    end
    A[0] ← temp;
end
```

问题描述及简单算法

Exercise (1)

已知一个长度为 n 的数组和一个正整数 k，并且最多只能使用一个用于交换的附加空间单元，试设计算法得到原数组循环右移 k 次的结果并分析算法的时间复杂度。

- 一般算法设计：

Algorithm ShiftRight($A[], n, k$)

```
k ← k mod n;
for i ← 1 to k do
    temp ← A[n − 1];
    for j ← n − 1 to 1 do
        A[j] ← A[j − 1];
    end
    A[0] ← temp;
end
```

- 当 $k \bmod n > n/2$ 时可以转换为循环左移 $n - k \bmod n$ 次
- 时间复杂度 $W(n) \in \Theta(kn) \Rightarrow \Theta(n^2)$

线性复杂度算法

1. 直接把元素放到右移 k 以后的位置，这样每个元素只需要移动一次，可保证时间复杂度为 $\Theta(n)$

线性复杂度算法

1. 直接把元素放到右移 k 以后的位置，这样每个元素只需要移动一次，可保证时间复杂度为 $\Theta(n)$

```
Algorithm ShiftRightFast(A[], n, k)
if n = 1 then return;
d ← GCD(n, k);
m ← n div d;
for i ← 0 to d − 1 do
    temp ← A[(i + (m − 1) ∗ k) mod n];
    for j ← m − 1 down to 1 do
        curr ← (i + j ∗ k) mod n;
        prev ← (i + (j − 1) ∗ k) mod n;
        A[curr] ← A[prev];
    end
    A[i mod n] ← temp;
end
```

线性复杂度算法 (CONT.)

2. 将原数组分割为两部分 AB，其中 B 的长度为 $k \bmod n$，那么最终我们需要的结果就是 BA，要得到这一结果，可用下面的思路：先将 A 逆序，再将 B 逆序，再将整个数组逆序，即可得到 BA
    - 复杂度：$W(n) = A(n) = 2n$
    - C++ STL 中 `std::rotate` 所使用的算法

线性复杂度算法 (CONT.)

2. 将原数组分割为两部分 AB，其中 B 的长度为 $k \bmod n$，那么最终我们需要的结果就是 BA，要得到这一结果，可用下面的思路：先将 A 逆序，再将 B 逆序，再将整个数组逆序，即可得到 BA
   - 复杂度：$W(n) = A(n) = 2n$
   - C++ STL 中 `std::rotate` 所使用的算法

3. 当 $k \bmod n < n/2$ 时，可将原数组分割为 $A_l A_r B$，其中 A_l 的长度与 B 的长度相同均为 $k \bmod n$，则需要的最终结果是 $BA_l A_r$，该结果可通过两步得到，先将 A_l 与 B 交换，再将 A_l 与 A_r 交换位置（注意这时二者长度不同）；若 $k \bmod n \geq n/2$，则可将原问题转化为循环左移 $n - k \bmod n$ 次，思路一样。
   - 关键是长度不同的两个数组如何交换位置
   - 复杂度可以达到 $\Theta(n)$
   - 分治法？

主要内容

1 习题解答

- 数组循环移位
- 递归程序设计
- 多米诺骨牌问题的分治算法
- 元素平均移动次数
- 5 元素中位数和排序
- 多米诺骨牌问题的动态规划算法
- 邮局位置问题
- 凝聚图和转置图

Exercise (2)

定义文件 xx.tar.gz 的产生方式如下:

- 以 xx 为文件名的文件通过 tar 和 gzip 打包压缩产生，该文件中以字符串的方式记录了一个非负整数;
- 或者以 xx 为名的目录通过 tar 和 gzip 打包压缩产生，该目录中包含若干 xx.tar.gz。

其中，$x \in [0, 9]$。现给定一个根据上述定义生成的文件 00.tar.gz (该文件从课程网站下载)，请确定其中包含的以 xx 为文件名的文件个数以及这些文件中所记录的非负整数之和。

答案：文件数：2177；非负整数之和：10854822。

主要内容

问题描述

Exercise (3)

现有 n 块"多米诺骨牌" $s_1, s_2, \cdots, s_n$ 水平放成一排，每块骨牌 s_i 包含左右两个部分，每个部分赋予一个非负整数值，如下图所示为包含 6 块骨牌的序列。骨牌可做 180 度旋转，使得原来在左边的值变到右边，而原来在右边的值移到左边，假设不论 s_i 如何旋转，$L[i]$ 总是存储 s_i 左边的值，$R[i]$ 总是存储 s_i 右边的值，$W[i]$ 用于存储 s_i 的状态：当 $L[i] \leq R[i]$ 时记为 0，否则记为 1，试采用分治法设计算法求 $\sum_{i=1}^{n-1} R[i] \cdot L[i+1]$ 的最大值，以及当取得最大值时每个骨牌的状态。下面是 $n = 6$ 时的一个例子。

s_1	s_2	s_3	s_4	s_5	s_6
5 | 8	4 | 2	9 | 6	7 | 7	3 | 9	11 | 10

问题描述

Exercise (3)

现有 n 块“多米诺骨牌” $s_1, s_2, \cdots, s_n$ 水平放成一排，每块骨牌 s_i 包含左右两个部分，每个部分赋予一个非负整数值，如下图所示为包含 6 块骨牌的序列。骨牌可做 180 度旋转，使得原来在左边的值变到右边，而原来在右边的值移到左边，假设不论 s_i 如何旋转，$L[i]$ 总是存储 s_i 左边的值，$R[i]$ 总是存储 s_i 右边的值，$W[i]$ 用于存储 s_i 的状态：当 $L[i] \le R[i]$ 时记为 0，否则记为 1，试采用分治法设计算法求 $\sum_{i=1}^{n-1} R[i] \cdot L[i+1]$ 的最大值，以及当取得最大值时每个骨牌的状态。下面是 $n = 6$ 时的一个例子。

s_1	s_2	s_3	s_4	s_5	s_6
5 | 8	4 | 2	9 | 6	7 | 7	3 | 9	11 | 10

解题思路：

5 | 8	4 | 2	9 | 6	7 | 7	3 | 9	11 | 10

问题描述

Exercise (3)

现有 n 块“多米诺骨牌”$s_1, s_2, \cdots, s_n$ 水平放成一排，每块骨牌 s_i 包含左右两个部分，每个部分赋予一个非负整数值，如下图所示为包含 6 块骨牌的序列。骨牌可做 180 度旋转，使得原来在左边的值变到右边，而原来在右边的值移到左边，假设不论 s_i 如何旋转，$L[i]$ 总是存储 s_i 左边的值，$R[i]$ 总是存储 s_i 右边的值，$W[i]$ 用于存储 s_i 的状态：当 $L[i] \le R[i]$ 时记为 0，否则记为 1，试采用分治法设计算法求 $\sum_{i=1}^{n-1} R[i] \cdot L[i+1]$ 的最大值，以及当取得最大值时每个骨牌的状态。下面是 $n = 6$ 时的一个例子。

$$\underset{s_1}{\boxed{5 \mid 8}} \quad \underset{s_2}{\boxed{4 \mid 2}} \quad \underset{s_3}{\boxed{9 \mid 6}} \quad \underset{s_4}{\boxed{7 \mid 7}} \quad \underset{s_5}{\boxed{3 \mid 9}} \quad \underset{s_6}{\boxed{11 \mid 10}}$$

解题思路：

$$\overbrace{\boxed{0} \quad \boxed{5 \mid 8} \quad \boxed{4 \mid 2} \quad \boxed{9}}^{S_1} \Big| \overbrace{\boxed{6} \quad \boxed{7 \mid 7} \quad \boxed{3 \mid 9} \quad \boxed{11 \mid 10} \quad \boxed{0}}^{S_2}$$

算法描述

Algorithm StoneLargest$(L[], R[], W[], f, l)$

```
if f < l then
    mid ← (f + l)/2;
    L1 ← StoneLargest(L, R, W, f, mid − 1);
    R1 ← StoneLargest(L, R, W, mid + 1, l);
    (Lt, Rt, Wt) ← (L[f..l], R[f..l], W[f..l]);
    Swap(L[mid], R[mid]); W[mid] ← −W[mid];
    L2 ← StoneLargest(L, R, W, f, mid − 1);
    R2 ← StoneLargest(L, R, W, mid + 1, l);
    if L1 + R1 < L2 + R2 then m ← L2 + R2 ;
    else (L[f..l], R[f..l], W[f..l]) ← (Lt, Rt, Wt); m ← L1 + R1 ;
else if f == l then
    m ← m1 ← R[f − 1] ∗ L[f] + R[f] ∗ L[f + 1];
    m2 ← R[f − 1] ∗ R[f] + L[f] ∗ L[f + 1];
    if m1 < m2 then
        Swap(L[f], R[f]); W[f] ← −W[f]; m ← m2;
    end
else m ← R[f − 1] ∗ L[f] ;
return m;
```

算法描述

Algorithm StoneLargest($L[], R[], W[], f, l$)

if $f < l$ **then**
    $mid \leftarrow (f+l)/2$;
    $L1 \leftarrow$ StoneLargest($L, R, W, f, mid-1$);
    $R1 \leftarrow$ StoneLargest($L, R, W, mid+1, l$);
    $(Lt, Rt, Wt) \leftarrow (L[f..l], R[f..l], W[f..l])$;
    Swap($L[mid], R[mid]$); $W[mid] \leftarrow -W[mid]$;
    $L2 \leftarrow$ StoneLargest($L, R, W, f, mid-1$);
    $R2 \leftarrow$ StoneLargest($L, R, W, mid+1, l$);
    if $L1 + R1 < L2 + R2$ **then** $m \leftarrow L2 + R2$;
    else $(L[f..l], R[f..l], W[f..l]) \leftarrow (Lt, Rt, Wt)$; $m \leftarrow L1 + R1$;
else if $f == l$ **then**
    $m \leftarrow m1 \leftarrow R[f-1] * L[f] + R[f] * L[f+1]$;
    $m2 \leftarrow R[f-1] * R[f] + L[f] * L[f+1]$;
    if $m1 < m2$ **then**
        Swap($L[f], R[f]$); $W[f] \leftarrow -W[f]$; $m \leftarrow m2$;
    end
else $m \leftarrow R[f-1] * L[f]$;
return m;

$$\boldsymbol{W(n) = 4W(n/2) + \Theta(n) \Rightarrow W(n) \in \Theta(n^2)}$$

主要内容

1 习题解答

- 数组循环移位
- 递归程序设计
- 多米诺骨牌问题的分治算法
- 元素平均移动次数
- 5 元素中位数和排序
- 多米诺骨牌问题的动态规划算法
- 邮局位置问题
- 凝聚图和转置图

问题及快速排序情况

Exercise (4)

试分析比较快速排序和归并排序在平均情况下元素移动次数。

- 快速排序
  - 每次比较，元素需要移动的可能性是多少？(是 1/2 吗?)
  - 元素移动发生在 Partition 阶段，而每次调用 Partition 时，子序列中元素两两之间都未经过比较，因此可以认为元素排列是随机的，从而 $pivot$ 元素为第 k 小元素的可能性为 $1/n$，且 Partition 结束后 $pivot$ 元素处于第 k 个位置
  - 每次 ele 与 $pivot$ 比较后需要移动当且仅当 $ele < pivot$ 且位于序列位置的 $(k, n]$ 区间内或者 $ele \geq pivot$ 且位于序列位置的 $[1, k)$ 区间内
  - 当序列元素随机均匀分布时这两种情况的概率分别为 $\dfrac{n-k}{n-1}$ 和 $\dfrac{k-1}{n-1}$

快速排序情况 (CONT.) 和归并排序情况

- **快速排序** (续)
  - 因此每次 Partition 的平均移动次数可计算如下：

$$M_{\text{ave}}(n) = \frac{1}{n}\sum_{k=1}^{n}\left(\frac{n-k}{n-1}\cdot(k-1)+\frac{k-1}{n-1}\cdot(n-k)\right)$$
$$= \frac{2}{n}\sum_{k=1}^{n}\frac{(n-k)(k-1)}{n-1} = \frac{n-2}{3} \approx \frac{n-1}{3}$$

  - 总的平均移动次数约为比较次数的 $1/3$，即大约 $0.462n\lg n$！
- **归并排序**
  - 使用最原始的归并程序，元素移动次数与子序列的元素数目相当
  - 总的移动次数约为 $n\lg n$，超过快速排序的两倍多！

补充讨论

- 归并程序的平均比较次数：

补充讨论

- 归并程序的平均比较次数：
  - m 个元素和 n 个元素归并可能产生的排列数：$C_{m+n}^{m} = C_{m+n}^{n}$

补充讨论

- 归并程序的平均比较次数：
  - m 个元素和 n 个元素归并可能产生的排列数：$C_{m+n}^{m} = C_{m+n}^{n}$
  - 比较次数：$C = m + n - S$，S 为剩余直接拷贝的元素数
  - 则 $S \geq s$ 的概率：

$$q_s = \begin{cases} \dfrac{C_{m+n-s}^{m} + C_{m+n-s}^{n}}{C_{m+n}^{m}}, & 1 \leq s \leq m+n; \\ 0, & s > m+n. \end{cases}$$

补充讨论

- 归并程序的平均比较次数：
  - m 个元素和 n 个元素归并可能产生的排列数：$C_{m+n}^{m} = C_{m+n}^{n}$
  - 比较次数：$C = m + n - S$，S 为剩余直接拷贝的元素数
  - 则 $S \geq s$ 的概率：

$$q_s = \begin{cases} \dfrac{C_{m+n-s}^{m} + C_{m+n-s}^{n}}{C_{m+n}^{m}}, & 1 \leq s \leq m+n; \\ 0, & s > m+n. \end{cases}$$

  - 则 S 的均值为：$\mu_{mn} = q_1 + q_2 + \cdots = \dfrac{m}{n+1} + \dfrac{n}{m+1}$

补充讨论

- 归并程序的平均比较次数：
  - m 个元素和 n 个元素归并可能产生的排列数：$C_{m+n}^{m} = C_{m+n}^{n}$
  - 比较次数：$C = m + n - S$，S 为剩余直接拷贝的元素数
  - 则 $S \geq s$ 的概率：

$$q_s = \begin{cases} \dfrac{C_{m+n-s}^{m} + C_{m+n-s}^{n}}{C_{m+n}^{m}}, & 1 \leq s \leq m+n; \\ 0, & s > m+n. \end{cases}$$

  - 则 S 的均值为：$\mu_{mn} = q_1 + q_2 + \cdots = \dfrac{m}{n+1} + \dfrac{n}{m+1}$
  - 平均比较次数 $C_{\mathsf{ave}} = m + n - \mu_{mn}$

补充讨论

- 归并程序的平均比较次数：
  - m 个元素和 n 个元素归并可能产生的排列数：$C_{m+n}^{m} = C_{m+n}^{n}$
  - 比较次数：$C = m + n - S$，S 为剩余直接拷贝的元素数
  - 则 $S \geq s$ 的概率：

$$q_s = \begin{cases} \dfrac{C_{m+n-s}^{m} + C_{m+n-s}^{n}}{C_{m+n}^{m}}, & 1 \leq s \leq m+n; \\ 0, & s > m+n. \end{cases}$$

  - 则 S 的均值为：$\mu_{mn} = q_1 + q_2 + \cdots = \dfrac{m}{n+1} + \dfrac{n}{m+1}$
  - 平均比较次数 $C_{\text{ave}} = m + n - \mu_{mn}$
  - 更深入的分析请参考：
    Donald E. Knuth. The Art of Computer Programming (Volume 3: Section 5.2.4)

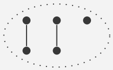
主要内容

1 习题解答

- 数组循环移位
- 递归程序设计
- 多米诺骨牌问题的分治算法
- 元素平均移动次数
- 5 元素中位数和排序
- 多米诺骨牌问题的动态规划算法
- 邮局位置问题
- 凝聚图和转置图

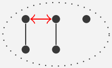
习题 5(A)：最多用 6 次比较找 5 个元素的中位数

第 3 次比较



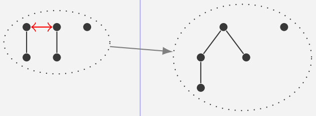
习题 5(A)：最多用 6 次比较找 5 个元素的中位数

第 3 次比较



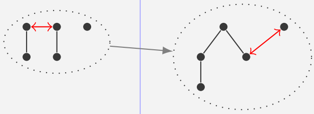
习题 5(A)：最多用 6 次比较找 5 个元素的中位数

第 3 次比较　　第 4 次比较

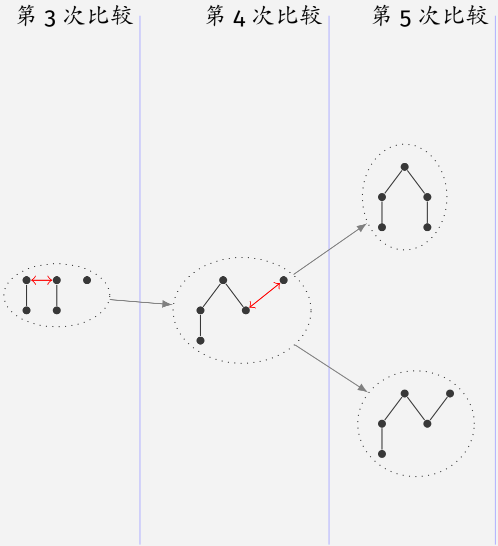


习题 5(A)：最多用 6 次比较找 5 个元素的中位数

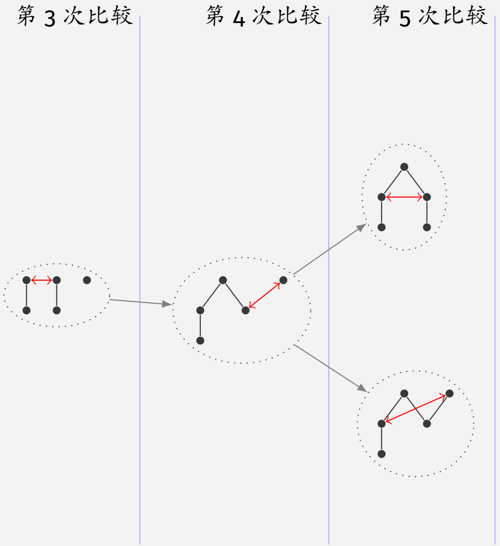
第 3 次比较 第 4 次比较



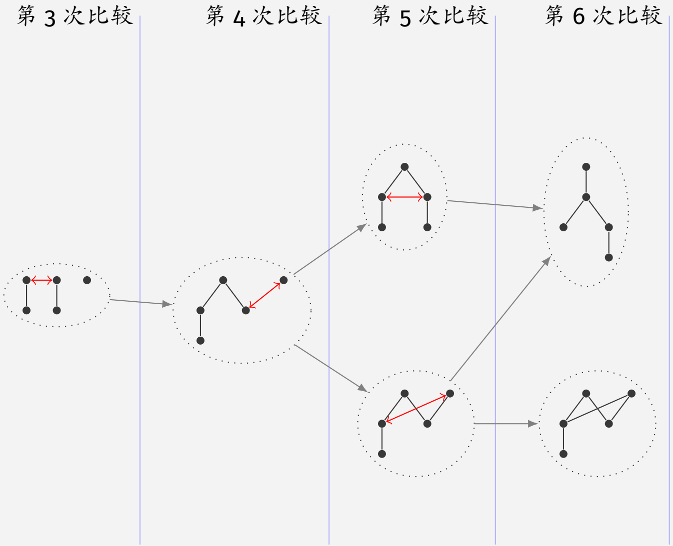
习题 5(A)：最多用 6 次比较找 5 个元素的中位数



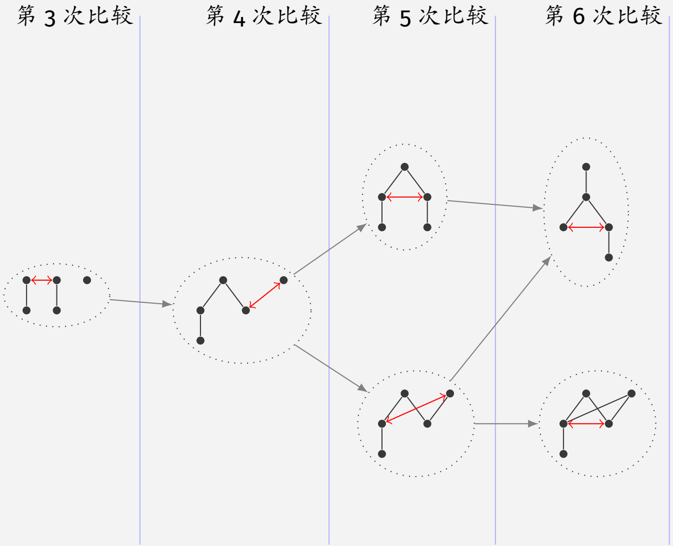
习题 5(A)：最多用 6 次比较找 5 个元素的中位数



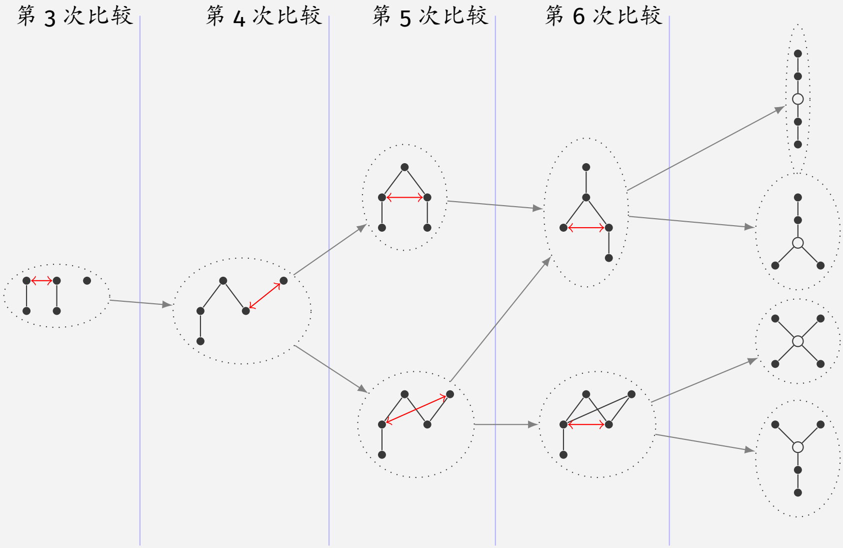
习题 5(A)：最多用 6 次比较找 5 个元素的中位数



习题 5(A)：最多用 6 次比较找 5 个元素的中位数

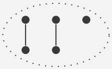


习题 5(A)：最多用 6 次比较找 5 个元素的中位数



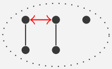
习题 5(B)：最多用 7 次比较对 5 个元素排序

第 3 次比较



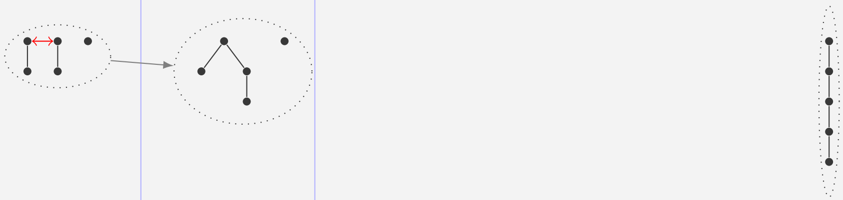
习题 5(B)：最多用 7 次比较对 5 个元素排序

第 3 次比较



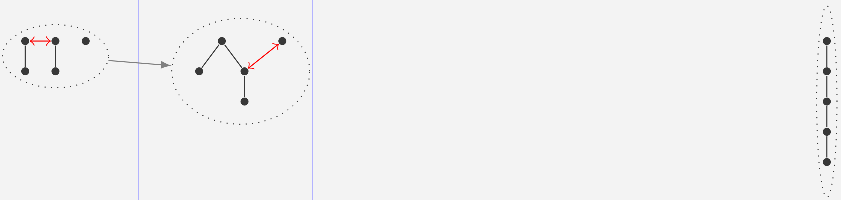
习题 5(B)：最多用 7 次比较对 5 个元素排序

第 3 次比较　第 4 次比较

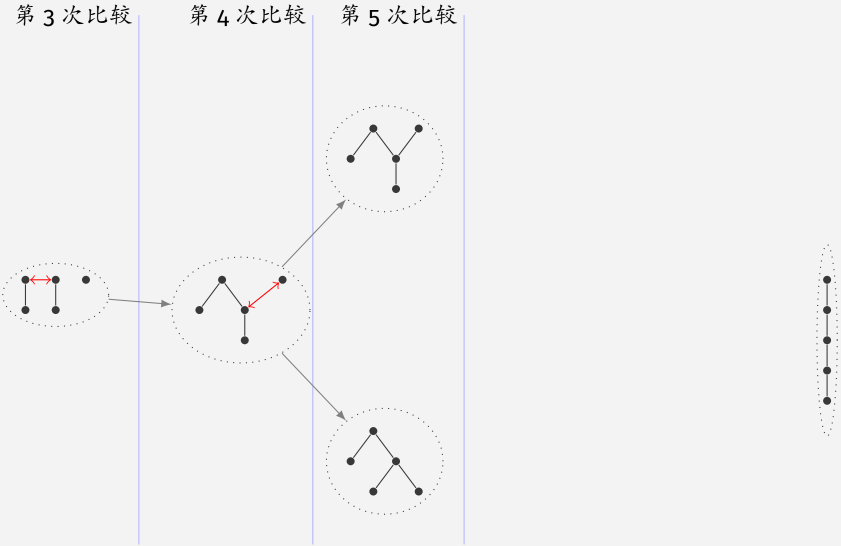


习题 5(B)：最多用 7 次比较对 5 个元素排序

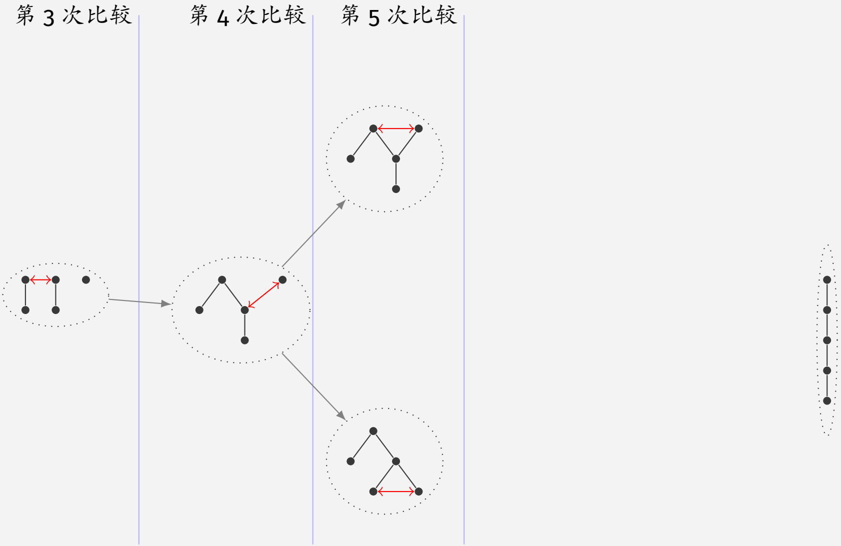
第 3 次比较 第 4 次比较



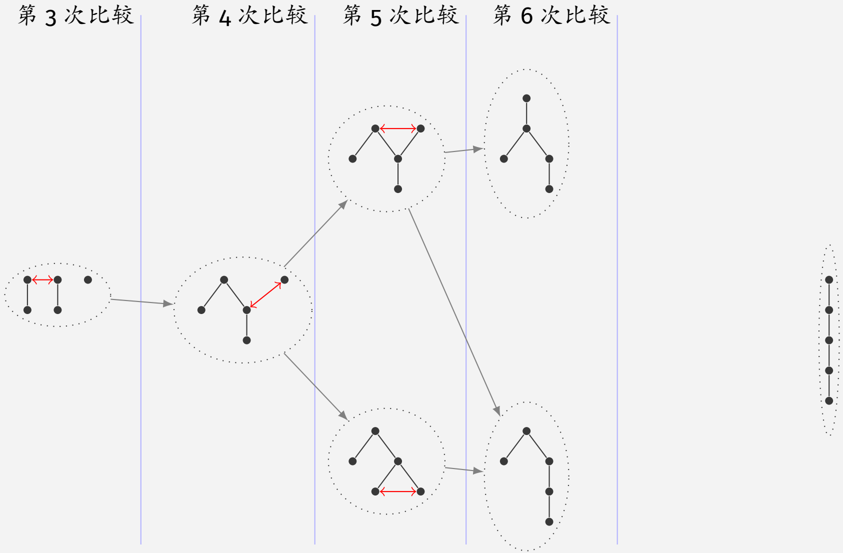
习题 5(b)：最多用 7 次比较对 5 个元素排序



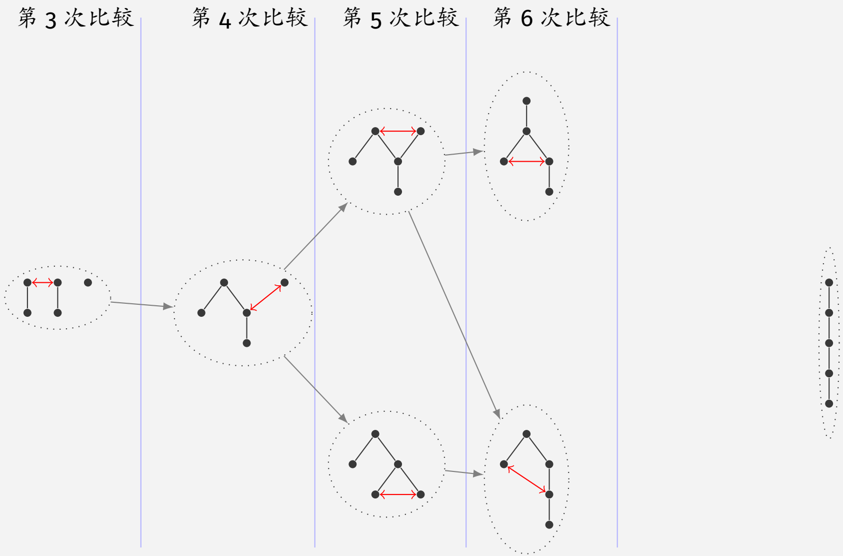
习题 5(B)：最多用 7 次比较对 5 个元素排序



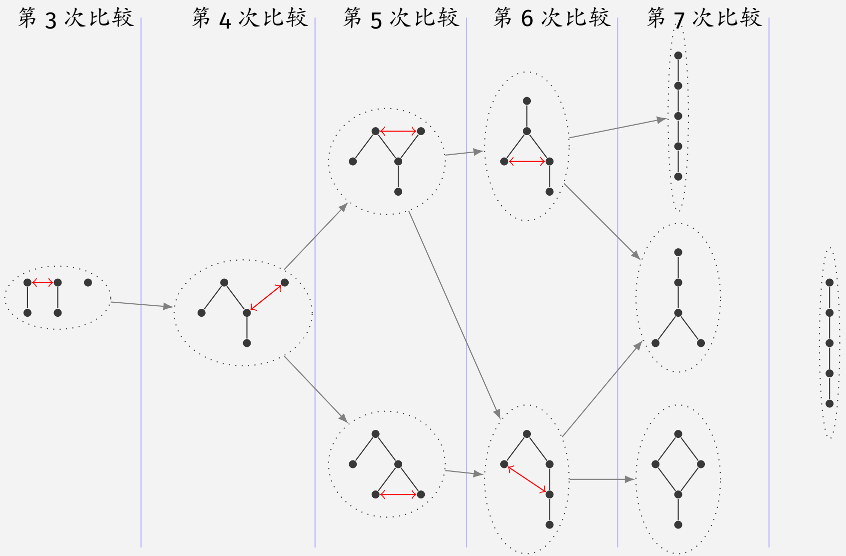
习题 5(b)：最多用 7 次比较对 5 个元素排序



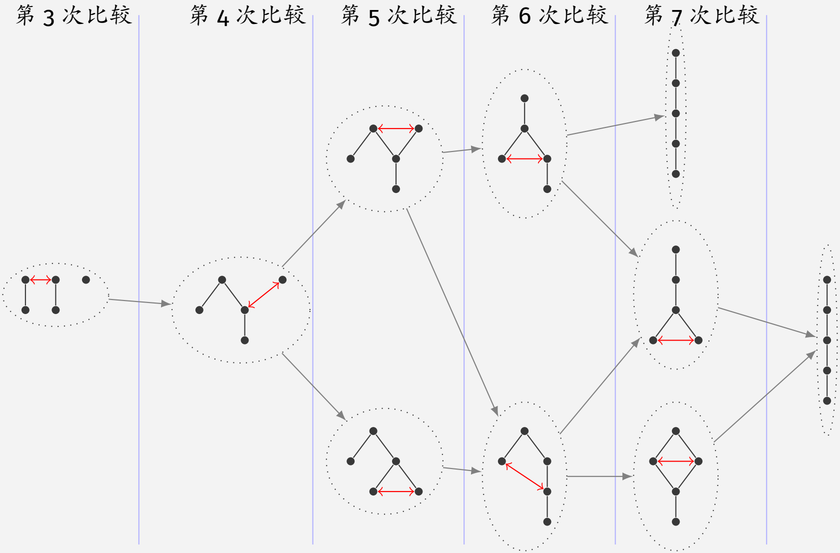
习题 5(b)：最多用 7 次比较对 5 个元素排序



习题 5(B)：最多用 7 次比较对 5 个元素排序



习题 5(B)：最多用 7 次比较对 5 个元素排序



主要内容

问题描述

Exercise (6)

现有 n 块“多米诺骨牌” $s_1, s_2, \cdots, s_n$ 水平放成一排，每块骨牌 s_i 包含左右两个部分，每个部分赋予一个非负整数值，如下图所示为包含 6 块骨牌的序列。骨牌可做 180 度旋转，使得原来在左边的值变到右边，而原来在右边的值移到左边，假设不论 s_i 如何旋转，$L[i]$ 总是存储 s_i 左边的值，$R[i]$ 总是存储 s_i 右边的值，$W[i]$ 用于存储 s_i 的状态：当 $L[i] \le R[i]$ 时记为 0，否则记为 1，试设计时间复杂度为 $O(n)$ 的动态规划算法求 $\sum_{i=1}^{n-1} R[i] \cdot L[i+1]$ 的最大值，以及当取得最大值时每个骨牌的状态。下面是 $n=6$ 时的一个例子。

s_1	s_2	s_3	s_4	s_5	s_6
5 | 8	4 | 2	9 | 6	7 | 7	3 | 9	11 | 10

动态规划算法设计

- 问题分割：
  - 子问题 (s,i)：当第 i 块骨牌状态为 s 时求 $\max(\sum_{k=1}^{i-1} R[k]L[k+1])$
  - 用二维数组 M 记录子问题结果，显然 M 的大小为 $2\times n$
  - 原问题求解目标：$\max(M[0,n],M[1,n])$
- 是否具有最优子结构性质：
  - 设 $(L[1..n],R[1..n])$ 为某一个最优解状态，若第 n 块骨牌状态确定，则需在 $L[n]$ 确定的情况下求 $(\sum_{k=1}^{n-2} R'[k]L'[k+1])+R'[n-1]L[n]$ 的最大值，可以肯定 $(L'[1..n-1],R'[1..n-1])=(L[1..n-1],R[1..n-1])$，否则用 $(L'[1..n-1],R'[1..n-1])$ 代替 $(L[1..n-1],R[1..n-1])$ 将得到一个更优的解，与假设矛盾
- 建立子问题求解的递推方程：
  - 设第 i 块骨牌 s_i 上的两个数为 a_i,b_i，且 $a_i\le b_i$，则：

$$\begin{aligned} M[0,1]&=M[1,1]=0\\ M[0,i+1]&=\max\{(M[0,i]+b_i\cdot a_{i+1}),(M[1,i]+a_i\cdot a_{i+1})\}\\ M[1,i+1]&=\max\{(M[0,i]+b_i\cdot b_{i+1}),(M[1,i]+a_i\cdot b_{i+1})\}\end{aligned}$$

算法描述

Algorithm `StoneLargest`$(L[], R[], W[])$

$M[0,1] \leftarrow M[1,1] \leftarrow 0;$
for $i \leftarrow 1$ **to** $n-1$ **do**
    $(a_i, b_i) \leftarrow (W[i]=0) \mathbin{?} (L[i], R[i]) \;:\; (R[i], L[i]);$
    $(a_{i+1}, b_{i+1}) \leftarrow (W[i+1]=0) \mathbin{?} (L[i+1], R[i+1]) \;:\; (R[i+1], L[i+1]);$
    if $(M[0,i] + b_i \cdot a_{i+1}) > (M[1,i] + a_i \cdot a_{i+1})$ **then**
        $M[0,i+1] \leftarrow M[0,i] + b_i \cdot a_{i+1};\ P[0,i+1] \leftarrow 0;$
    else $M[0,i+1] \leftarrow M[1,i] + a_i \cdot a_{i+1};\ P[0,i+1] \leftarrow 1\,;$
    if $(M[0,i] + b_i \cdot b_{i+1}) > (M[1,i] + a_i \cdot b_{i+1})$ **then**
        $M[1,i+1] \leftarrow M[0,i] + b_i \cdot b_{i+1};\ P[1,i+1] \leftarrow 0;$
    else $M[1,i+1] \leftarrow M[1,i] + a_i \cdot b_{i+1};\ P[1,i+1] \leftarrow 1\,;$
end
if $M[0,n] > M[1,n]$ **then** $M \leftarrow M[0,n];\ W[n] \leftarrow 0\,;$
else $M \leftarrow M[1,n];\ W[n] \leftarrow 1\,;$
for $i \leftarrow n$ **down to** 2 **do**
    $W[i-1] \leftarrow W[i] = 0 \mathbin{?} P[0,i] \;:\; P[1,i];$
end
return $(W[], M);$

主要内容

1 习题解答

- 数组循环移位
- 递归程序设计
- 多米诺骨牌问题的分治算法
- 元素平均移动次数
- 5 元素中位数和排序
- 多米诺骨牌问题的动态规划算法
- 邮局位置问题
- 凝聚图和转置图

Exercise (7)

在一条街上有 n 所房子，$H[i]$ $(1 \le i \le n)$ 是第 i 所房子离街道起点处的距离（以米为单位），假定 $H[1] < H[2] < \cdots < H[n]$。目前该街道上还没有一所邮局，现计划新建若干所邮局，使得每所房子到最近的邮局距离在 100 米以内。试设计一个时间复杂度为 $O(n)$ 的算法，计算出新建邮局的位置，即每所新建邮局离街道起点处的距离 $P[j]$ $(1 \le j \le m)$，同时确保新建邮局个数 m 最小。

- 贪心方法思路：
  - 首先确定第一所邮局位置应为 $P[1] = H[1] + 100$；
  - 其后从 $H[2]$ 开始检查每个 $H[i]$，若 $H[i] > P[m] + 100$，则 m 增 1，$P[m] = H[i] + 100$，否则继续检查下一所房子位置 $H[i+1]$；
  - 最后，如果 $P[m] > H[n]$，则令 $P[m] = H[n]$

算法描述

Algorithm `PostOffice`($H[], n$)

$P[1] \leftarrow H[1] + 100$;
$m \leftarrow 1$;
for $i \leftarrow 2$ **to** n **do**
    if $H[i] > P[m] + 100$ **then**
        $m \leftarrow m + 1$;
        $P[m] \leftarrow H[i] + 100$;
    end
end
if $P[m] > H[n]$ **then** $P[m] \leftarrow H[n]$;
return (P, m);

算法的正确性

反证法.

- 假设存在某种更优的邮局位置设置 $P'[1..k]$，使得所需新建的邮局数目更少为 $k < m$
- 根据 $P[i]$ 的位置选择有 $P'[i] \leq P[i]$，否则：若有 $P'[a-1] \leq P[a-1]$ 而 $P'[a] > P[a]$，设 $P[a] = H[b] + 100$，则 $P'[a] - H[b] > 100$ 而 $P'[a-1] - H[b] \leq P[a-1] - H[b] < -100$，即房子 $H[b]$ 未被 P' 覆盖；
- 因此 $P'[k] \leq P[k]$，且由于 $m > k$，必存在 p 使 $H[p] > P[k] + 100$，所以 $P'[k] < H[p] - 100$，即存在房子 $H[p]$ 未被 P' 覆盖，与 P' 是原问题的解矛盾。

$\square$

主要内容

1 习题解答

- 数组循环移位
- 递归程序设计
- 多米诺骨牌问题的分治算法
- 元素平均移动次数
- 5 元素中位数和排序
- 多米诺骨牌问题的动态规划算法
- 邮局位置问题
- 凝聚图和转置图

问题描述及解题思路

Exercise (8)

给定有向图 G:

(1) 证明图 G 的凝聚图 $G\downarrow$ 是有向无环图。

(2) 若图 G 以邻接表的形式存储，试写出一个算法求图 G 的转置图 G^T。

- 证明凝聚图是有向无环图，用反证法，利用凝聚图中假设存在的环路构造原图 G 中的环路
- 根据转置图的定义，只需将原图 G 的所有边反转方向即可得到图 G 的转置图 G^T，即遍历原图邻接表中的每条边，反转方向后加入到新图的邻接表中

(1) 证明凝聚图是有向无环图

Proof.

用反证法，假设 $s_0, s_1, \cdots, s_k$ $(s_k = s_0, k \geq 1)$ 是图 G 的凝聚图 $G\downarrow$ 中的一个环路，令 S_i 是图 G 中对应于 $G\downarrow$ 中顶点 s_i 的强连通分量 $(0 \leq i \leq k,\ S_k = S_0)$，对于每一个 i $(0 \leq i \leq k-1)$，$G\downarrow$ 中存在一条边 $s_i s_{i+1}$，意味着 G 中存在一条边 $v_i w_i$，其中 $v_i \in S_i$ 且 $w_i \in S_{i+1}$，而又由于每个 S_i 的强连通性，在图 G 中必存在一条从 w_i 到 v_{i+1} 的路径，记为 $w_i \to v_{i+1}$，由此可以在图 G 中构造一条环路如下：$v_0, w_0 \to v_1, w_1 \to \cdots \to v_{k-1}, w_{k-1} \to v_k (= v_0)$，从而这条环路上的所有顶点将属于同一个强连通分量，这与假设相矛盾，因此 $G\downarrow$ 中不存在环路，也就是说图 G 的凝聚图 $G\downarrow$ 是有向无环图。 □

(2) 求转置图算法描述

```
Algorithm Transpose(G)
将图 G 的顶点加入 G^T;
foreach v ∈ G^T do trAdjVertices[v] ← ∅;
foreach v ∈ G do
    remAdj ← adjVertices[v];
    while remAdj ≠ null do
        w ← First(remAdj);
        trAdjVertices[w] ← Cons(v, trAdjVertices[w]);
        remAdj ← Rest(remAdj);
    end
end
return G^T;
```

其中，First、Rest 和 Cons 均为链表基本操作，分别是：取链表第一个元素、取除第一个元素外的剩余链表、将一个元素加到链表头构成一个新的链表。