FSE598 前沿计算技术

模块 3　算法设计与分析
单元 2　排序算法
第 1 讲　快速排序

本课程的部分内容是基于 Thomas H.Cormen、Charles E.Leiserson 等人的“算法简介”教材

课程概要

学习内容

- 采用分治技术的快速排序
- 快速排序的关键思路
- 快速排序的最佳、最坏和平均情况的复杂度
- 复杂度证明
- 随机化快速排序

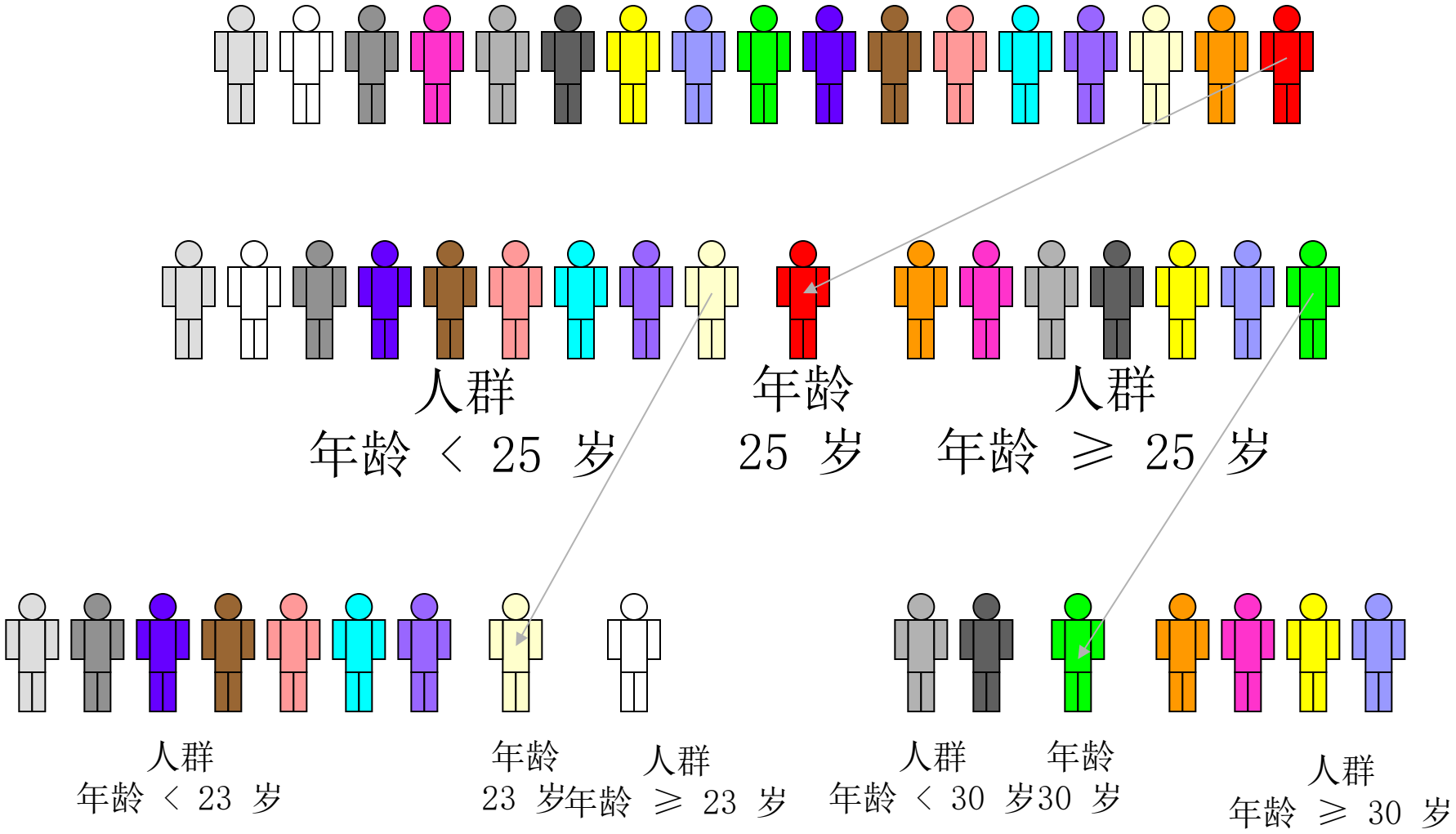
分治编程技术

- 分治：
  - 一种递归编程技术
  - “分”：将一个问题分解为几个较小的子问题（几个 size-m 问题）
  - “治”：用递归方式求解每个子问题
  - 组合：把这些解进行组合
- 排序算法
  - 快速排序
  - 堆排序
  - 二叉搜索树及排序
  - 归并排序
  - 选择排序

快速排序

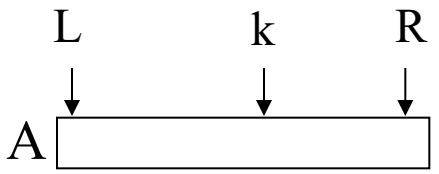
- 分：将数字序列划分为 2 个子序列，使得第一个子序列中的每个数字都小于或等于第二个子序列中的各个元素；
- 治：用递归方法对每个子序列进行排序
- 组合：这一部分不需要任何操作，所有元素已经分类到位。

示例：把人群按年龄分类


人群
年龄 < 25 岁
年龄
25 岁
人群
年龄 ≥ 25 岁
人群
年龄 < 23 岁
年龄
23 岁
人群
年龄 ≥ 23 岁
人群
年龄 < 30 岁
年龄
30 岁
人群
年龄 ≥ 30 岁

快速排序算法

```
Quicksort (A, L,R)
    if R > L then
        k := Partition(A, L, R);
        Quicksort (A, L, k-1);
        Quicksort (A, k+1, R);
```



该算法的关键思路是分区（“分”）模块，它会重新排列列表，使得：

- 元素 $A[k]$（$pivot$）位于列表中的最终位置
- $A[L]..A[k-1]$ 中的所有元素*都要小于* $A[k]$。
- $A[k+1]..A[R]$ *中的所有元素*都要等于或大于 $A[k]$。

分区模块是如何实现的？

理想情况下，我们可以重新排列列表，使一半元素位于 k 的左侧，一半元素位于 k 的右侧。在实践中：

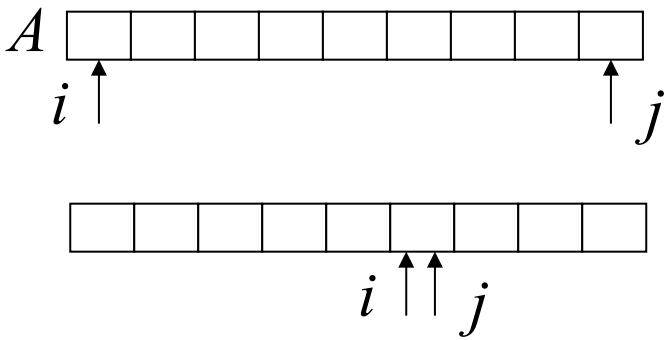
1. 选择 $A[R]$ 作为 $pivot$ 元素，目的是使其移动到列表中的正确位置。
2. 从左侧开始扫描，直至找到一个 $\geqslant A[R]$ 的元素。
3. 从右侧开始扫描，直至找到一个 $< A[R]$ 的元素。
4. 交换这两个元素。
5. 继续这种方式，直至 L 和 R 扫描指针相遇。
6. 将 $A[R]$ 与右侧子列表中最靠左的元素交换（即指针指向的元素）。

划分/分区算法

```
Partition(A, L, R)
    v := A[R];
    i := L;
    j := R;
    while i < n do
        while A[i] < v do
            i:=i+1;
        while j > i and A[j] ≥ v do
            j := j – 1;
        if  i < j then
            swap(A[i], A[j]);
    swap(A[i], A[R]);
    return I;
```

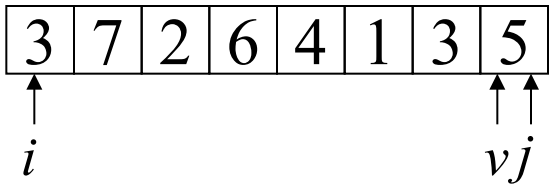
运行时间是多少？

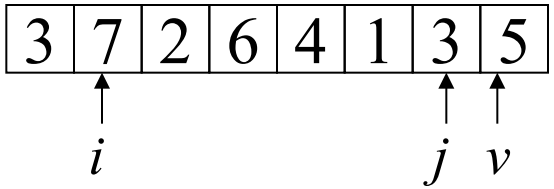
$= n^2$ ？

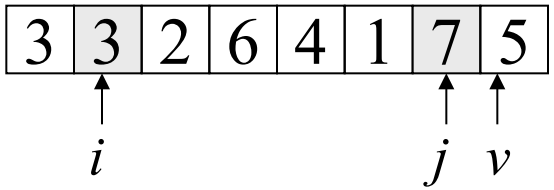


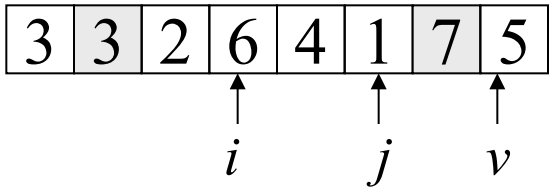
$\Theta(n)$

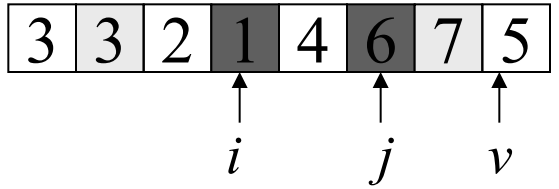
示例

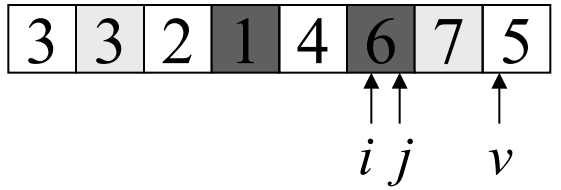

3 7 2 6 4 1 3 5
i
v j

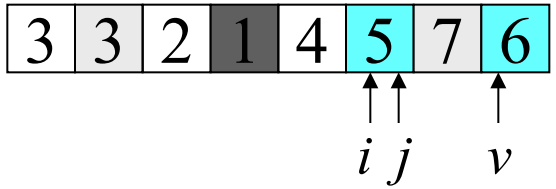

3 7 2 6 4 1 3 5
i
j v

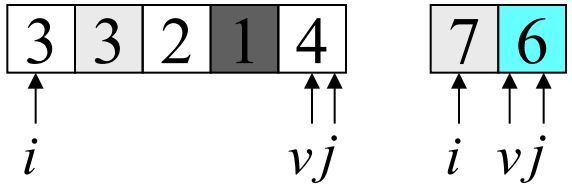

3 3 2 6 4 1 7 5
i
j v


3 3 2 6 4 1 7 5
i
j
v


3 3 2 1 4 6 7 5
i
j
v


3 3 2 1 4 6 7 5
i j
v


3 3 2 1 4 5 7 6
i j
v


3 3 2 1 4
i
v j
7 6
i
v j

快速排序的复杂度

复杂度取决于分区操作。

最佳情况的分区：分成两个相等的子列表

$$T(n)=\begin{cases}\Theta(1) & if\ n=1\\ 2T(n/2)+\Theta(n) & otherwise\end{cases} \Rightarrow \quad T(n) = \Theta(n\lg n)$$

最坏情况的分区：列表已经排序的情况：

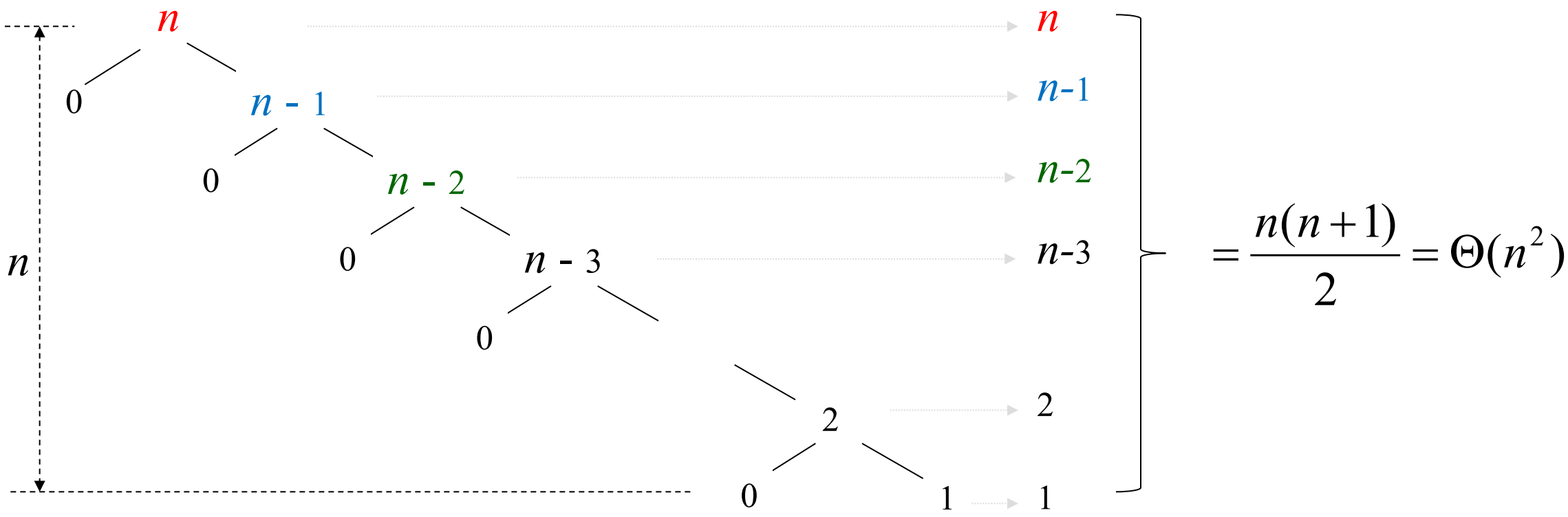
右侧的子列表：0 个元素；左侧的子列表：$n-1$ 个元素

$$T(n)=T(n-1)+\Theta(n)=T(n-2)+\Theta(n-1)+\Theta(n)$$

$$=\cdots=\sum_{k=1}^{n}\Theta(k)=\Theta\sum_{k=1}^{n}(k)=\Theta\left(\frac{n(n+1)}{2}\right)=\Theta(n^2)$$

最坏情况的递归树

$$T(n) = n + T(n\text{-}1) \quad = n + (n\text{-}1) + T(n\text{-}2) = n + (n\text{-}1) + (n\text{-}2) + \ldots$$



$$= \frac{n(n+1)}{2} = \Theta(n^2)$$

快速排序的复杂度（接上页）

很慢？

不平衡情况的分区：按 9：1（90%：10%）的比例划分为 2 个子列表。

定理：按 9:1 比例分区的快速排序的时间复杂度 $\Theta(n\lg n)$

与最佳情况的复杂度相同

我们也可以证明：

- 对于 99：1 的分区：$\Theta(n\lg n)$
- 对于 n：1 或 n：100 的分区：$\Theta(n^2)$

证明（通过迭代方法）

(1) $T(n)=T(\frac{9}{10}n)+T(\frac{1}{10}n)+n$

(2) $=\left(T(\frac{9}{10}\frac{9}{10}n)+T(\frac{1}{10}\frac{9}{10}n)+\frac{9}{10}n\right)+\left(T(\frac{9}{10}\frac{1}{10}n)+T(\frac{1}{10}\frac{1}{10}n)+\frac{1}{10}n\right)+n$

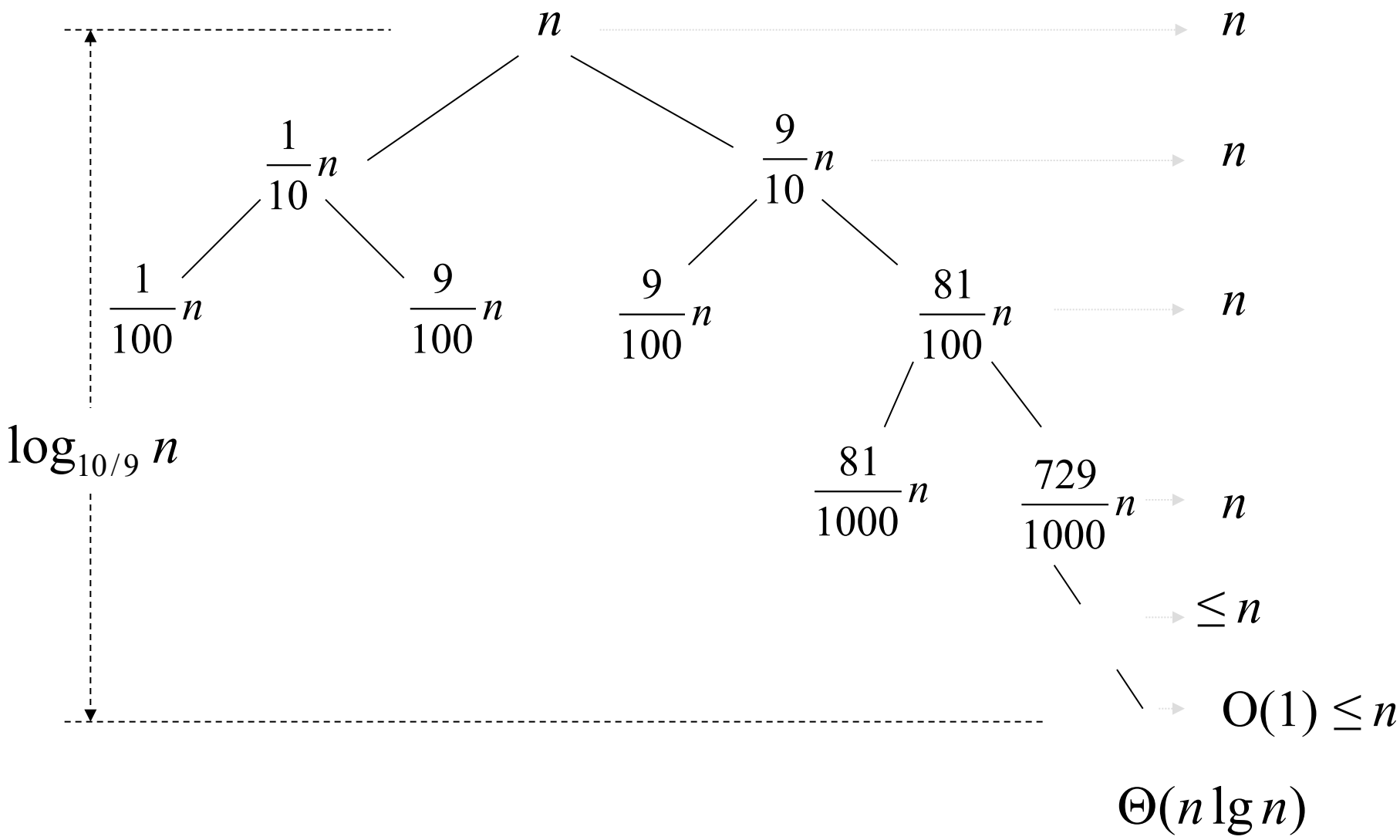
$=\left(T(\frac{81}{100}n)+T(\frac{9}{100}n)\right)+\left(T(\frac{9}{100}n)+T(\frac{1}{100}n)\right)+n+n \quad = 2^{2}*T(\leq 1)+2^{1}*n$

(k) $=2^{k}T(\leq 1)+kn=2^{k}+kn$ k 由最长路径决定

当 $k=\log_{10/9}n$ ⇒ T（）中的所有变量值均 ≤ 1

$$T(n)=2^{k}+kn=2^{\log_{10/9}n}+n\log_{10/9}n=O(n\lg n)$$

证明（通过递归树）


n
n
$\frac{1}{10}n$
$\frac{9}{10}n$
n
$\frac{1}{100}n$
$\frac{9}{100}n$
$\frac{9}{100}n$
$\frac{81}{100}n$
n
$\log_{10/9} n$
$\frac{81}{1000}n$
$\frac{729}{1000}n$
n
$\leq n$
$O(1) \leq n$
$\Theta(n \lg n)$

快速排序的平均复杂度

- 你不需要完全理解此证明中的所有具体内容，但要掌握如何评估平均运行时间的思路。
- 按 q-to-(n-q) 的比例分区各种排列方式的概率相同，其中 $q = 1, 2, \ldots, n\text{-}1$

$$T(n) = \frac{1}{n}\sum_{q=1}^{n-1}\big(T(q)+T(n-q)\big)+\Theta(n)$$

$$= \frac{2}{n}\sum_{k=1}^{n-1}\big(T(k)\big)+\Theta(n)$$

$n = 5$

$\boldsymbol{T(1)} + T(4)$

$T(2) + T(3)$

$T(3) + T(2)$

$T(4) + \boldsymbol{T(1)}$

- 我们将用替代方法证明，递归式的解（下一页）为：

$$T(n) = \Theta(n\lg n)$$

快速排序平均复杂度的证明

归纳假设：$T(k) \le ak\lg k + b \quad for \quad k < n$

$$T(n) = \frac{2}{n}\sum_{k=1}^{n-1}\left(T(k)\right) + \Theta(n)$$

$$\le \frac{2}{n}\sum_{k=1}^{n-1}\left(ak\lg k + b\right) + \Theta(n) \quad = \frac{2}{n}a\sum_{k=1}^{n-1}\left(k\lg k\right) + \frac{2}{n}b(n-1) + \Theta(n)$$

我们可以证明：$\sum_{k=1}^{n-1}\left(k\lg k\right) \le \frac{1}{2}n^2\lg n - \frac{1}{8}n^2$

$$T(n) \le \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \frac{2b}{n}(n-1) + \Theta(n)$$

$$\le an\lg n + b + (\Theta(n) + b - an/4) \quad \le an\lg n + b \Rightarrow T(n) = \Theta(n\lg n)$$

选择一个足够大的 n

快速排序的随机版本

- 快速排序的性能取决于输入排列；
- 最坏情况复杂度 $O(n^2)$ 和平均情况复杂度 $O(n\lg n)$
- 如果输入排列的概率不相等，我们可以利用随机数生成器，使其具有相同的概率：$\text{Random}(l, r) \rightarrow i$，$i$ 介于 l 和 r 之间

```
Randomized-partition (A, L, R)
    i := Random(L, R)
    swap(A[R], A[i])
    return Partition(A, L, R)
```

```
Randomized-quicksort (A, L, R)
    if R > L then
        i := Randomized-partition(A, L,R)
        Randomized-quicksort (A, L, i-1)
        Randomized-quicksort (A, i+1, R)
```