排序

快速排序：比较次数

14-A5

时间究竟是什么
假使人家不问我，我
像很明了
假使要我解释起来，我就茫无头绪

庞拉尔·特尔内拉在这场造梦运动中出力最多，她成功地
将纸牌算命从推演未来应用到追溯过往

邓俊辉
deng@tsinghua.edu.cn

递推分析（1/2）

❖ 记期望的比较次数为 $T(n)$： $T(1)=0,\ T(2)=1,\ \ldots$

❖ 可以证明：$T(n)=\mathcal{O}(n\log n)$ ...

$$T(n) = (n-1) + \frac{1}{n} \cdot \sum_{k=0}^{n-1} [T(k) + T(n-k-1)] = (n-1) + \frac{2}{n} \cdot \sum_{k=0}^{n-1} T(k)$$

$$n \cdot T(n) = n \cdot (n-1) + 2 \times \sum_{k=0}^{n-1} T(k)$$

$$(n-1) \cdot T(n-1) = (n-1) \cdot (n-2) + 2 \times \sum_{k=0}^{n-2} T(k)$$

递推分析（2/2）

$$n \cdot T(n) \ - \ (n-1) \cdot T(n-1) \ = \ 2 \cdot (n-1) \ + \ 2 \times T(n-1)$$

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

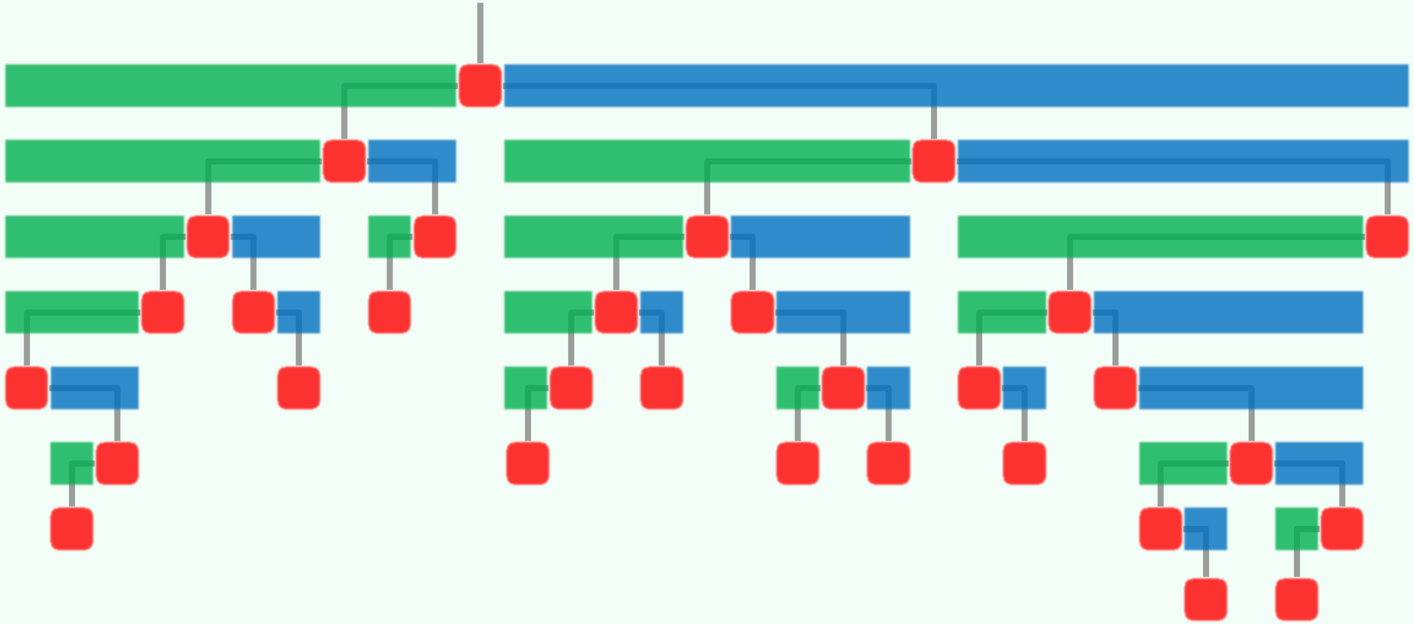
$$\frac{T(n)}{n+1} - \frac{T(n-1)}{n} = \frac{4}{n+1} - \frac{2}{n}$$

$$\frac{T(n)}{n+1} = \frac{T(n)}{n+1} - \frac{T(1)}{2} = 4 \cdot \sum_{k=2}^{n} \frac{1}{k+1} - 2 \cdot \sum_{k=2}^{n} \frac{1}{k} = 2 \cdot \sum_{k=1}^{n+1} \frac{1}{k} + \frac{2}{n+1} - 4 \approx 2 \cdot \ln n$$

$$T(n) \ \approx \ 2 \cdot n \cdot \ln n \ = \ (2 \cdot \ln 2) \cdot n \log n \ \approx \ 1.386 \cdot n \log n$$

后向分析（1/2）

❖ 设经排序后得到的**输出**序列为： $\{ a_0,\ a_1,\ a_2,\ \ldots\ldots\ldots,\ a_i,\ \ldots\ldots\ldots,\ a_j,\ \ldots\ldots\ldots,\ a_{n-1} \}$



❖ 这一输出与具体使用何种算法**无关**

故可使用**Backward Analysis**

❖ 比较操作的**期望**次数应为

$$T(n) = \sum_{i=0}^{n-2} \sum_{j=i+1}^{n-1} Pr(i,j) = \sum_{j=1}^{n-1} \sum_{i=0}^{j-1} Pr(i,j)$$

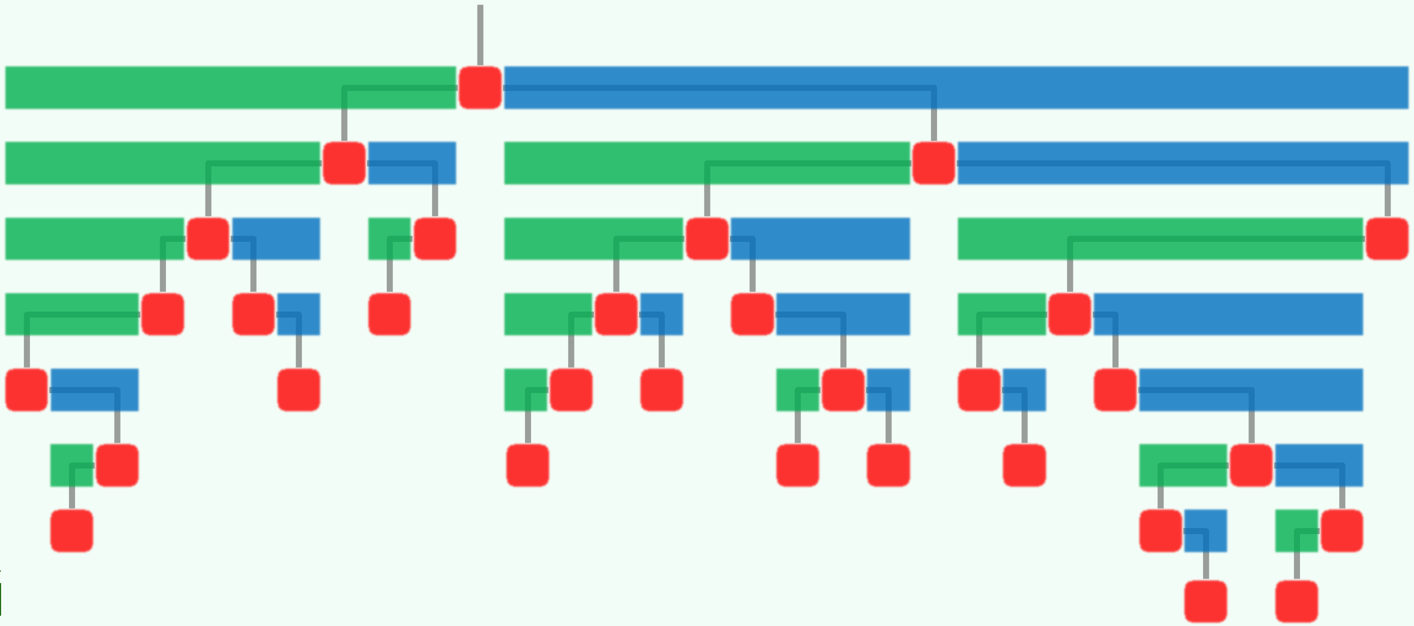
亦即，每一对 $\langle a_i, a_j \rangle$ 在排序过程中会做比较之**概率**的总和

后向分析（2/2）

❖ quickSort的过程及结果可理解为：

将所有元素逐个地**转化**为pivot

$\{\ a_0,\ a_1,\ a_2,\ \ldots\ldots\ldots,\ a_i,\ \ldots\ldots\ldots,\ a_j,\ \ldots\ldots\ldots,\ a_{n-1}\ \}$



❖ 若 $k \notin [i, j]$，则

a_k **早于或晚于** a_i 和 a_j 被转化，

均与 $Pr(i, j)$ **无关**

❖ 进一步地，$\langle a_i, a_j \rangle$ 会做比较，当且仅当

在 $\{\ a_i,\ a_{i+1},\ a_{i+2},\ \ldots,\ a_{j-2},\ a_{j-1},\ a_j\ \}$ 中，a_i 或 a_j **率先**被转化

$$T(n) = \sum_{j=1}^{n-1} \sum_{i=0}^{j-1} Pr(i, j) = \sum_{j=1}^{n-1} \sum_{d=1}^{j} \frac{2}{d+1} \approx \sum_{j=1}^{n-1} 2 \cdot (\ln j - 1) \le 2 \cdot n \cdot \ln n$$

对比

Vector sorter	#compare	#move	cache-friendly
Insertionsort	$\mathcal{O}(n) \sim \mathcal{O}(n^2)$ expected-$\mathcal{O}(n^2)$	$\mathcal{O}(n) \sim \mathcal{O}(n^2)$ expected-$\mathcal{O}(n^2)$	
Selectionsort	$\Theta(n^2)$	$\mathcal{O}(n)$	
Heapsort	2.0*nlogn	1.0*nlogn	percolateDown(): ✗[i] ✗[j]
Mergesort	1.0*nlogn	1.5*nlogn	merge(): ✓[i] ✓[j] ✓[k]
Quicksort	expected-1.386*nlogn w.h.p.	expected-(1.386/2 = 0.694)*nlogn	✓✓pivot ✓[lo] ✓[hi]