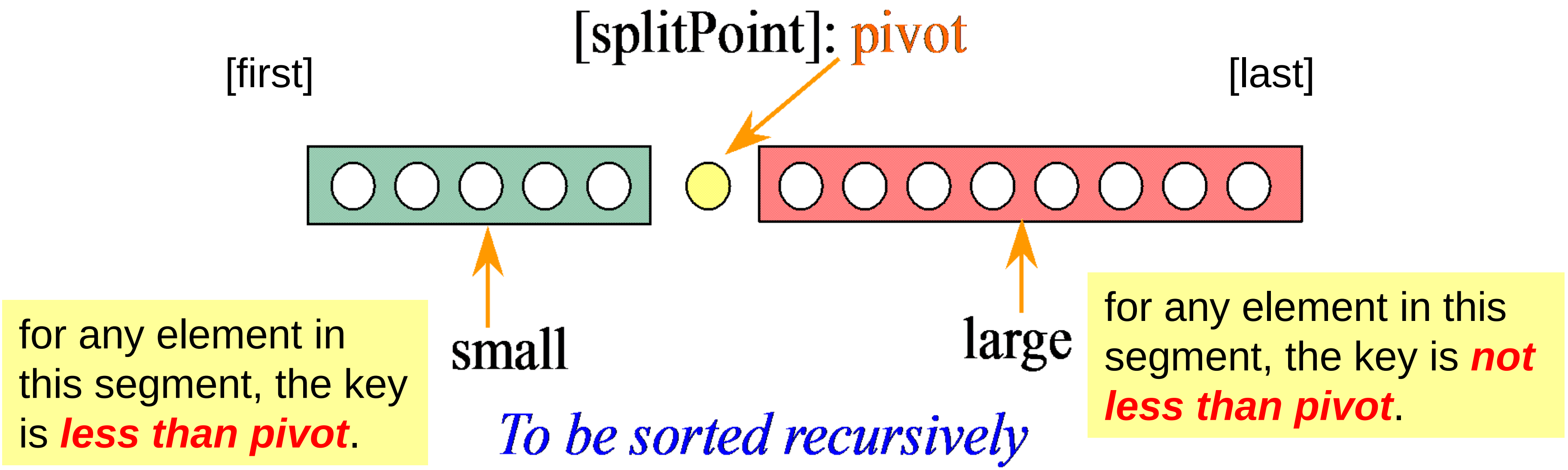
计算机问题求解 – 论题2-10
- 排序与选择

2016年4月21日

问题1:

你能否利用下图解释快速排序的基本思想?



```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

To sort an entire array A, the initial call is QUICKSORT(A, 1, $A.length$)

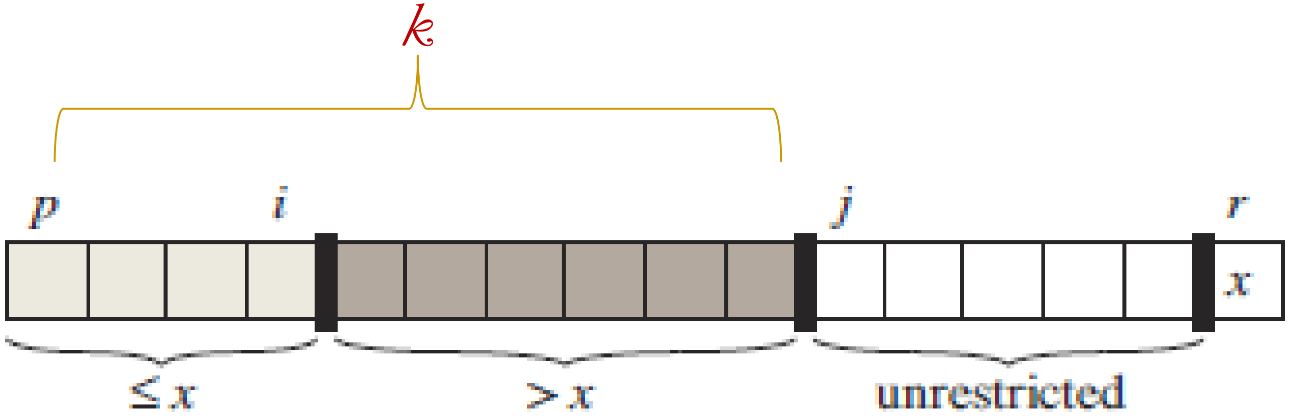
问题2:

都是用divide-and conquer策略,这与Mergesort有什么不同?

关键是Partition

Partition过程的核心是一个循环，执行r-p次。

在第k+1次循环开始前，格局如下：



问题3:

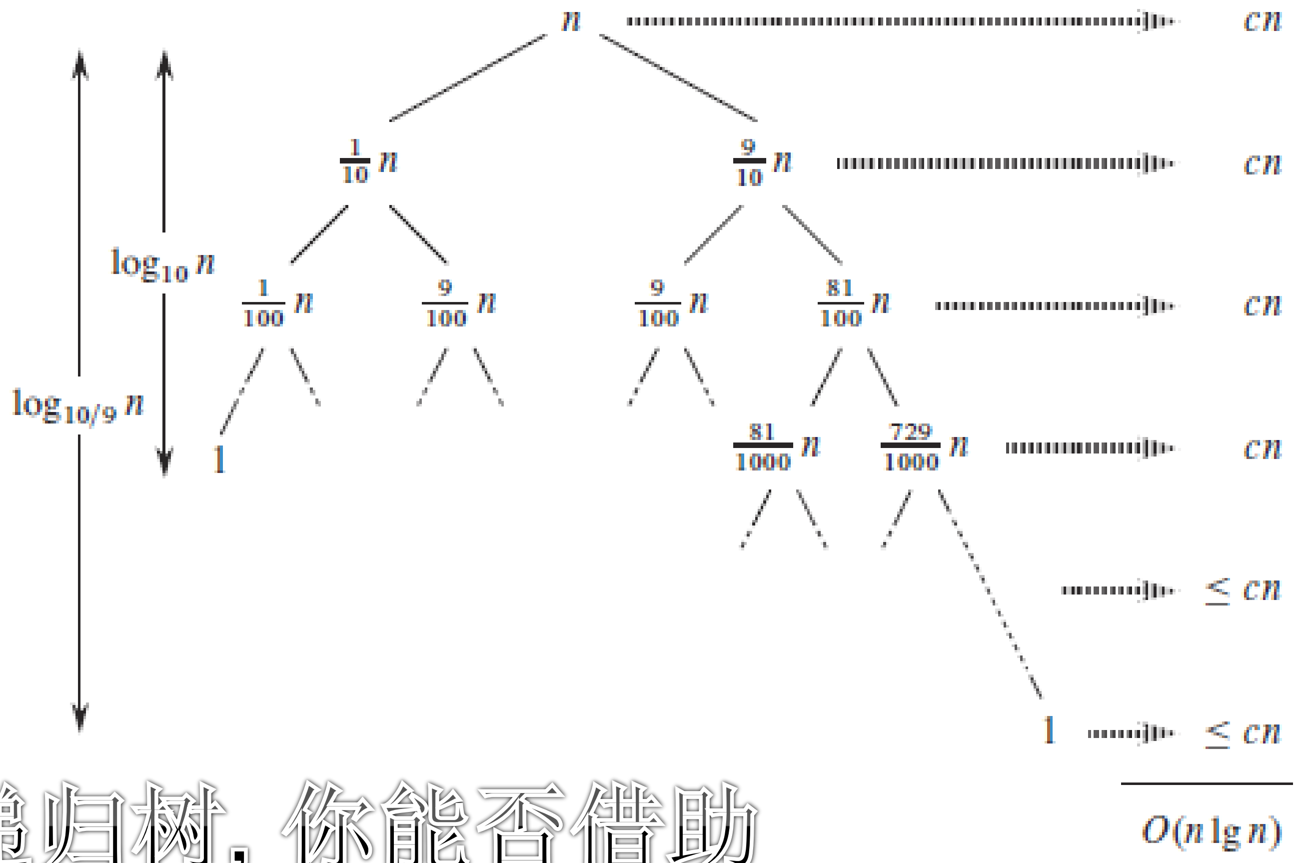
你能否借助此图，解释利用循环不变式如何证明这个过程的正确性？

```
QUICKSORT(A, p, r)
if p < r
    q = PARTITION(A, p, r)
    QUICKSORT(A, p, q − 1)
    QUICKSORT(A, q + 1, r)
```

To sort an entire array A, the initial call is QUICKSORT(A, 1, $A.length$)

问题4:

这个递归算法的时间性能递归式是什么？



问题5:

什么是递归树，你能否借助这个图解释影响Quicksort效率的因素？

In fact, any split of *constant proportionality* yields a recursion tree of depth $\Theta(\lg n)$, where the cost at each level is $O(n)$. The running time is therefore $O(n \lg n)$ whenever the split has constant proportionality.

问题6：

你看了这句话有什么想法？

问题7：

为什么直观上也会觉得快速排序的平均效率更接近最好情况，而不是最坏情况？

直觉对于探索很重要。

“大胆假设，小心求证！”

Worst-Case Performance of Quicksort

$$T(n) = \max_{0 \le q \le n-1} (T(q) + T(n-q-1)) + \Theta(n)$$

q是Partition的返回值。

问题8：

你能说说解这个递归的策略吗？

Guess and verifying

随机快速排序的期望代价

- 快速排序的主要代价其实就是Partition的代价。
- Partition的代价主要是循环中的比较操作。

Lemma 7.1

Let X be the number of comparisons performed in line 4 of PARTITION over the entire execution of QUICKSORT on an n-element array. Then the running time of QUICKSORT is $O(n + X)$.

Proof By the discussion above, the algorithm makes at most n calls to PARTITION, each of which does a constant amount of work and then executes the **for** loop some number of times. Each iteration of the **for** loop executes line 4. ■

随机快速排序的期望时间：

- 定义随机变量X：
  - 任意给定一个随机待排序列σ，X(σ)表示对σ进行排序时进行的比较次数
- 关于QS中的“比较”的两个事实：
  - 任何两个元素之间最多比较1次。
    - 比较只发生在pivot和其它元素之间
  - 如果将A中元素按照大小重命名为$z_1,z_2,\cdots,z_n$，定义z_{ij}为$\{z_i,z_{i+1},\cdots,z_j\}$，则$z_i$和$z_j$进行比较 iff z_i或者z_j在z_{ij}中首先被选为pivot

问题9：为什么？

如何计算X的期望？

Our analysis uses indicator random variables (see Section 5.2). We define

$X_{ij} = \mathrm{I}\{z_i \text{ is compared to } z_j\}$,

$$\mathrm{E}[X] = \mathrm{E}\left[\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right]$$

X 的期望值

$$
\begin{aligned}
\mathrm{E}[X] &= \mathrm{E}\left[\sum_{i=1}^{n-1}\sum_{j=i+1}^{n} X_{ij}\right] = \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \mathrm{E}[X_{ij}] \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \underline{\Pr\{z_i \text{ is compared to } z_j\}} \\
&= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-i+1}
\end{aligned}
$$

$$
\boxed{
\begin{aligned}
&= \Pr\{z_i \text{ or } z_j \text{ is first pivot chosen from } Z_{ij}\} \\
&= \Pr\{z_i \text{ is first pivot chosen from } Z_{ij}\} \\
&\quad + \Pr\{z_j \text{ is first pivot chosen from } Z_{ij}\} \\
&= \frac{1}{j-i+1} + \frac{1}{j-i+1} \\
&= \frac{2}{j-i+1}.
\end{aligned}
}
$$

$$
\begin{aligned}
\mathrm{E}[X] &= \sum_{i=1}^{n-1}\sum_{j=i+1}^{n} \frac{2}{j-i+1} \\
&= \sum_{i=1}^{n-1}\sum_{k=1}^{n-i} \frac{2}{k+1} \\
&< \sum_{i=1}^{n-1}\sum_{k=1}^{n} \frac{2}{k} \\
&= \sum_{i=1}^{n-1} O(\lg n) \\
&= O(n \lg n).
\end{aligned}
$$

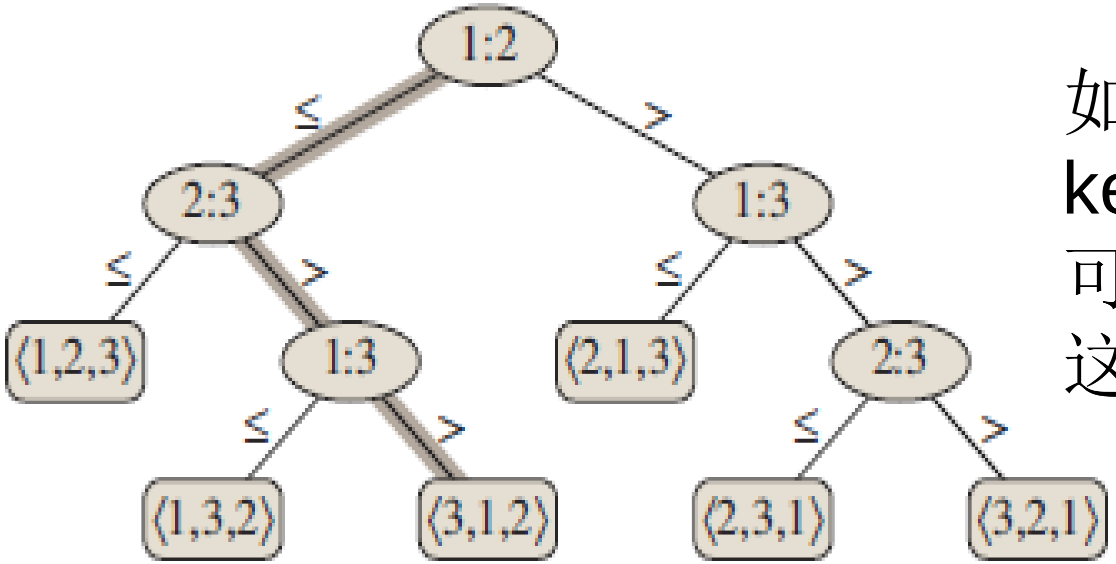
令$k=j-i$

$O(n\log n)$； 还能改进吗？

Sorting is a central problem in many areas of computing so it is no surprise to see an approach to solving the problem as one of the top 10. Joseph JaJa describes Quicksort as one of the best practical sorting algorithm for general inputs. In addition, its complexity analysis and its structure have been a rich source of inspiration for developing general algorithm techniques for various applications.

2000年, IEEE Computing in Science and Engineering 评选20世纪对科学与工程发展影响最大的10个算法, 快速排序位列其中。

Decision-tree Model



如果允许的操作只是比较key的值，任一排序过程可以用左边的树表示。这里，n=3。

$$\lg(n!) = n\lg n$$

问题10：

为什么至少是6个叶子？

问题11：

问题的lower bound和算法的performance有什么不同，有什么关系？

线性的排序算法

- 上面所讨论的lower bound是针对基于key比较的排序算法的。
- 充分利用附加的条件，可得到更高效的算法：
  - counting sort;
  - radix sort;
  - bucket sort;

问题12：
利用了什么条件，为什么能提高效率？

Counting sort

```
COUNTING-SORT(A, B, k)
let C[0..k] be a new array
for i = 0 to k
    C[i] = 0
for j = 1 to A.length
    C[A[j]] = C[A[j]] + 1
// C[i] now contains the number of elements equal to i.
for i = 1 to k
    C[i] = C[i] + C[i - 1]
// C[i] now contains the number of elements less than or equal to i.
for j = A.length downto 1
    B[C[A[j]]] = A[j]
    C[A[j]] = C[A[j]] - 1
```

	1	2	3	4	5	6	7	8
A	2	5	3	0	2	3	0	3

	0	1	2	3	4	5
C	2	0	2	3	0	1

(a)

	0	1	2	3	4	5
C	2	2	4	7	7	8

(b)

Radix sort:

```
RADIX-SORT(A, d)
  for i = 1 to d
      use a stable sort to sort array A on digit i
```

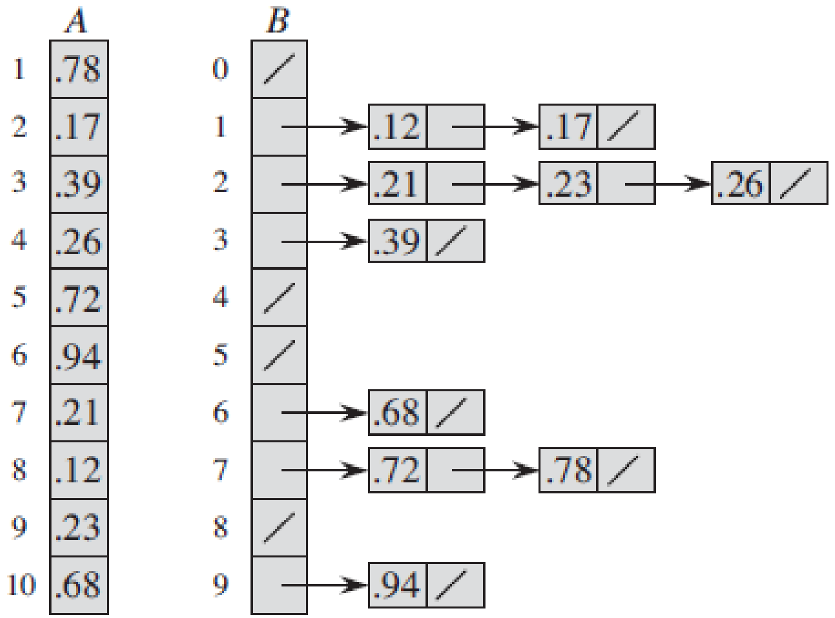
假设我们没有用稳定算法进行digit i的排序，你能举一个例子说明排序失败吗？

329	720	720	329
457	355	329	355
657	436	436	436
839	457	839	457
436	657	355	657
720	329	457	720
355	839	657	839

Bucket sort

BUCKET-SORT(A)

let $B[0\,..\,n-1]$ be a new array
$n = A.length$
for $i = 0$ **to** $n - 1$
    make $B[i]$ an empty list
for $i = 1$ **to** n
    insert $A[i]$ into list $B[\lfloor nA[i] \rfloor]$
for $i = 0$ **to** $n - 1$
    sort list $B[i]$ with insertion sort
concatenate the lists $B[0], B[1], \ldots, B[n-1]$ together in order



问题13:

什么是“stable sort”，在算法设计中有什么意义？

问题求解与信息收集

- 问题：任给一个长度为*n*的序列，其元素属于一个全序集，找出其中的最小/最大元素。
  - 需要执行*n*-1次比较操作
- 但是：如果问题是要既找出最小元素，也找出最大元素，并不需要执行2*n*-2次操作。

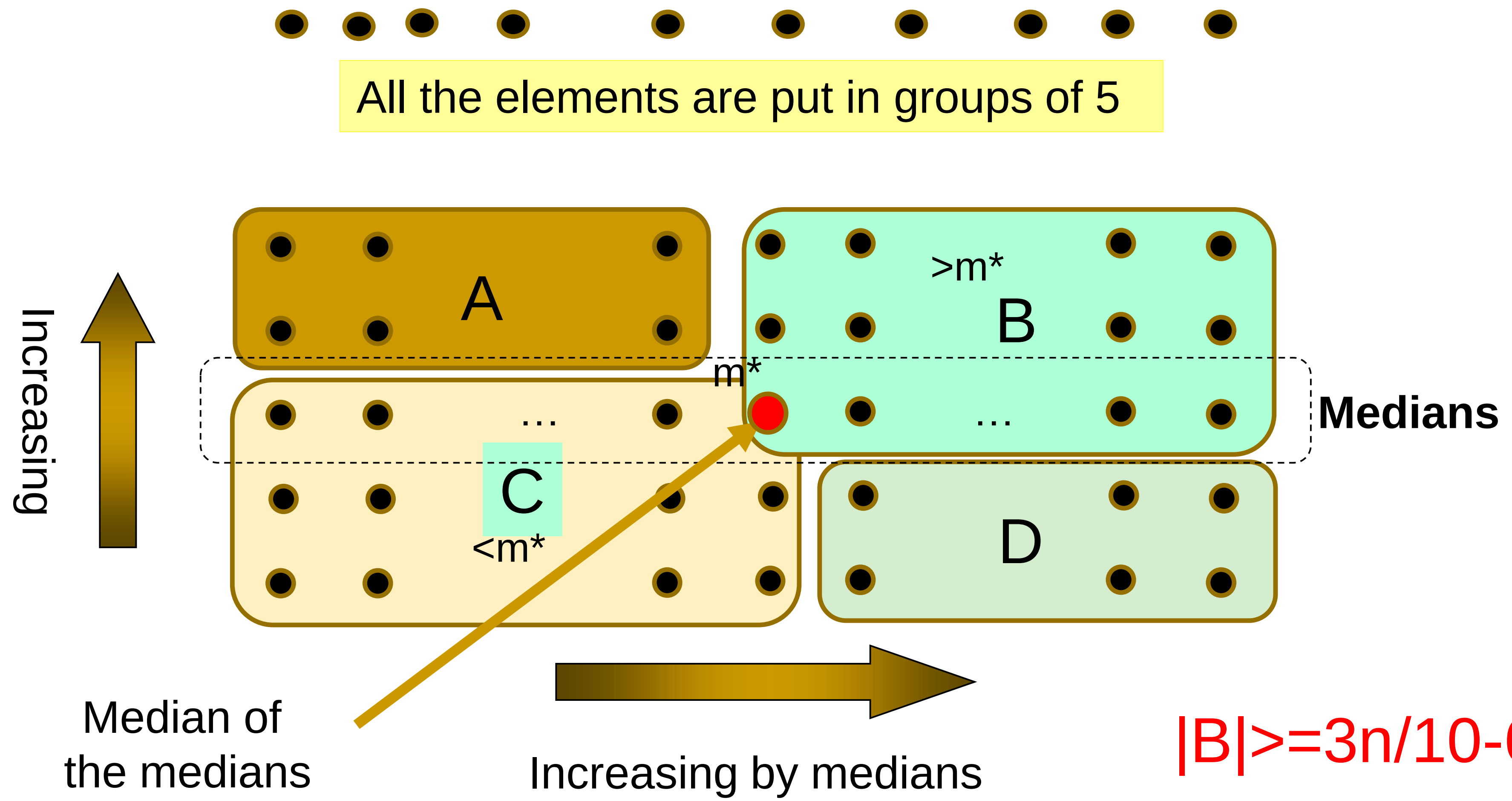
问题14：为什么？

Selection Problem

```
RANDOMIZED-SELECT(A, p, r, i)
if p == r
    return A[p]
q = RANDOMIZED-PARTITION(A, p, r)
k = q − p + 1
if i == k          // the pivot value is the answer
    return A[q]
elseif i < k
    return RANDOMIZED-SELECT(A, p, q − 1, i)
else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

这个算法很象Quicksort，但只需要对一个子问题递归。
采用特殊的方法选择pivot，可以得到保证一定均衡的确定算法！得到最坏情况下的线性算法

Partition Improved: the Strategy



|B|>=3n/10-6

Construct the Partition and recursive call

- 寻找m^*, 5个元素小组的中位数集合的中位数
  - 5个元素小组的中位数寻找，需要6次比较.
- 采用m^*为pivot，对A数组进行划分
  - 比较A和D区中所有元素和m^*
  - 令 $S_1=C\cup\{x|x\in A\cup D \text{ and } x<m^*\}$
  - 令 $S_2=B\cup\{x|x\in A\cup D \text{ and } x>m^*\}$
- 如果i=|S1|+1,返回A[|S1|+1]
- 如果 i<=|S1|, 在 S1递归寻找i顺序数; 否则在S2中递归寻找i-|S1|-1顺序数

Counting the Number of Comparisons

$$T(n) \le 6 \times {}^{n}\!/_{5} + T({}^{n}\!/_{5}) + n + T({}^{7n}\!/_{10} + 6)$$

The extreme case: all the elements in A∪D in one subset.

Finding the median in every group of 5

Finding the median of the medians

Comparing all the elements in A∪D with *m**

可使用代入法求解上述递归式：T(n)=O(n)

我们还需要分析这个算法的平均执行时间吗？

问题15:
比较6次找出5个数中的中位数,比你想像的要难,你试试?

Open topics:

- 请严格证明快速排序算法的正确性
  - 部分正确性+完全正确性
- 证明count sort是稳定的
- 证明：randomized-select算法的期望运行时间是Θ(n)的

课外作业

- TC 第7章：
  - Ex.7.1-2; 7.2-4; 7.3-2; 7.4-2;
  - Prob.7.4; 7.5
- TC 第8章：
  - Ex.8.1-3; 8.1-4; 8.2-4; 8.3-4; 8.4-2;
  - Prob.8.2
- TC 第9章：
  - Ex.9.1-1; 9.3-5; 9.3-7