计算机问题求解 – 论题2–7

- 排序与选择

课程研讨

- TC第7、8、9章

问题1：Quicksort

- 你能简述Quicksort的执行过程吗？
- PARTITION中的loop invariant是什么？
- 你能证明PARTITION是totally correct吗？
- 你能证明QUICKSORT是totally correct吗？

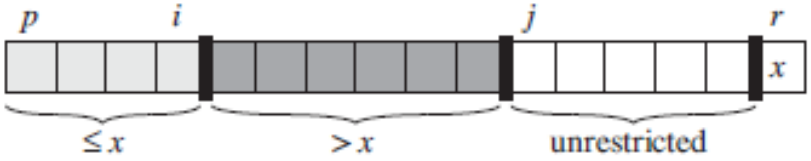
```
QUICKSORT(A, p, r)
1  if p < r
2      q = PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)

PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

问题1：Quicksort

- 你能简述Quicksort的执行过程吗？
- PARTITION中的loop invariant是什么？
- 你能证明PARTITION是totally correct吗？
- 你能证明QUICKSORT是totally correct吗？

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

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```



1. If $p \le k \le i$, then $A[k] \le x$.
2. If $i + 1 \le k \le j - 1$, then $A[k] > x$.
3. If $k = r$, then $A[k] = x$.

问题1： Quicksort (续)

- What is the running time of Quicksort when array A contains distinct elements and is sorted in decreasing order?
- What is the running time of Quicksort when all elements of array A have the same value?
- What is the best case?
  What is the worst case?

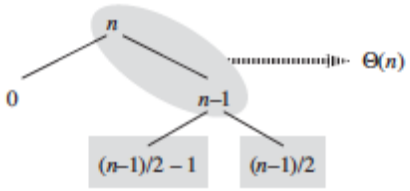
```
QUICKSORT(A, p, r)
1  if p < r
2      q = PARTITION(A, p, r)
3      QUICKSORT(A, p, q − 1)
4      QUICKSORT(A, q + 1, r)

PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

问题1： Quicksort (续)

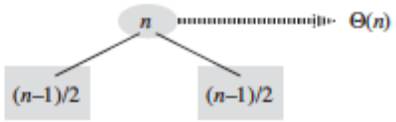
- What is the running time of Quicksort when array A contains distinct elements and is sorted in decreasing order?
- What is the running time of Quicksort when all elements of array A have the same value?

- What is the best case?
  What is the worst case?

- How about the average case?



bad



good

average?

Ordering the Elements with 3 Colors

- Suppose an array *A* consists of *n* element, each of which is red, white or blue. Design a linear algorithm to sort the array so that all the reds come before the whites, and which come before all the blues. The only operations permitted are:
  - Examine(*A*, *i*) – report the color of the *i*th element, and
  - Swap(*A*, *i*, *j*) – swap the *i*th element of *A* with the *j*th element.

Matching Bolts and Nuts

- You are given a collection of *n* bolts of different widths and *n* corresponding nuts. You can determine whether the nut is
  - larger than the bolt
  - smaller than the bolt
  - matches the bolt exactly
- However, there is **no way** to compare two nuts together or two bolts together.
- The problem is to match each bolt to its nut.

问题1： Quicksort (续)

- RANDOMIZED-QUICKSORT与QUICKSORT有什么不同？
- 这种改变有什么意义？
  -

- RANDOMIZED-QUICKSORT的运行时间主要耗费在什么操作？

```
RANDOMIZED-QUICKSORT(A, p, r)
1  if p < r
2      q = RANDOMIZED-PARTITION(A, p, r)
3      RANDOMIZED-QUICKSORT(A, p, q − 1)
4      RANDOMIZED-QUICKSORT(A, q + 1, r)
```

```
RANDOMIZED-PARTITION(A, p, r)
1  i = RANDOM(p, r)
2  exchange A[r] with A[i]
3  return PARTITION(A, p, r)
```

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

问题1：Quicksort (续)

- RANDOMIZED-QUICKSORT与QUICKSORT有什么不同？
- 这种改变有什么意义？
  - In exploring the average-case behavior of quicksort, we have made an assumption that all permutations of the input numbers are equally likely. In an engineering situation, however, we cannot always expect this assumption to hold.
- RANDOMIZED-QUICKSORT的运行时间主要耗费在什么操作？

```
RANDOMIZED-QUICKSORT(A, p, r)
1  if p < r
2      q = RANDOMIZED-PARTITION(A, p, r)
3      RANDOMIZED-QUICKSORT(A, p, q − 1)
4      RANDOMIZED-QUICKSORT(A, q + 1, r)
```

```
RANDOMIZED-PARTITION(A, p, r)
1  i = RANDOM(p, r)
2  exchange A[r] with A[i]
3  return PARTITION(A, p, r)
```

```
PARTITION(A, p, r)
1  x = A[r]
2  i = p − 1
3  for j = p to r − 1
4      if A[j] ≤ x
5          i = i + 1
6          exchange A[i] with A[j]
7  exchange A[i + 1] with A[r]
8  return i + 1
```

问题1：Quicksort (续)

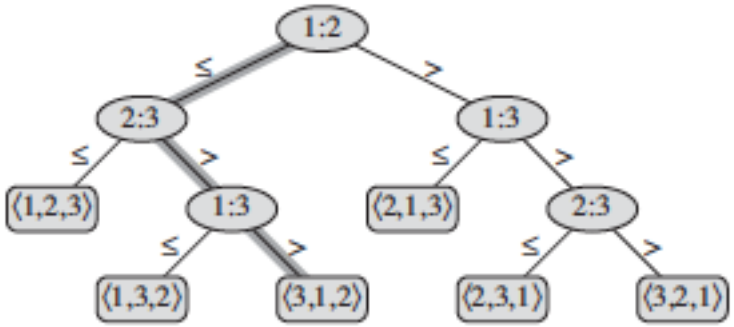
- 比较次数的计算
  - 为什么每对元素最多比较1次？
  - 你能解释以下计算过程吗？

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

  - 然后如何计算expected running time？

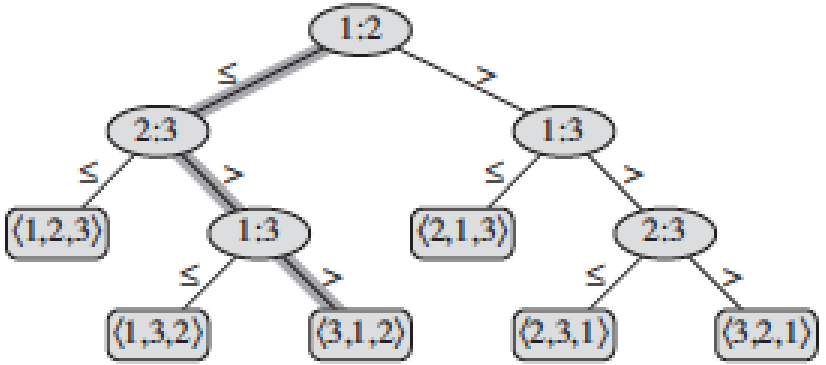
问题2：Sorting in linear time

- 什么叫做comparison sorts？
  -
- 你是怎么理解decision tree的？
  它与comparison sorts的运行时间有什么关系？
  - 它有多少个叶子？
  - 它有多少层？



问题2：Sorting in linear time

- 什么叫做comparison sorts？
  - The sorted order they determine is based only on comparisons between the input elements
- 你是怎么理解decision tree的？
  它与comparison sorts的运行时间有什么关系？
  - 它有多少个叶子？
  - 它有多少层？

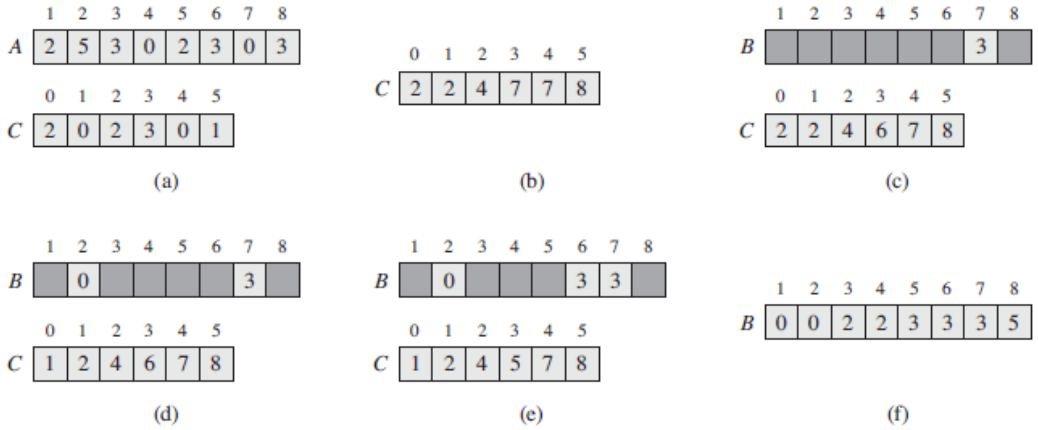


问题2：Sorting in linear time

- 你能简述counting sort的执行过程吗？
- 为什么它是stable的？（什么叫stable？QUICKSORT是吗？）

  你能不能将最后一步改为从左往右扫描，并仍保证stable？
- 它的使用有哪些局限性（缺点）？

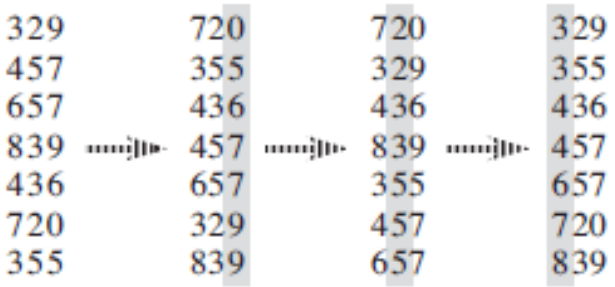
```
COUNTING-SORT(A, B, k)
 1  let C[0..k] be a new array
 2  for i = 0 to k
 3      C[i] = 0
 4  for j = 1 to A.length
 5      C[A[j]] = C[A[j]] + 1
 6  // C[i] now contains the number of elements equal to i.
 7  for i = 1 to k
 8      C[i] = C[i] + C[i - 1]
 9  // C[i] now contains the number of elements less than or equal to i.
10  for j = A.length downto 1
11      B[C[A[j]]] = A[j]
12      C[A[j]] = C[A[j]] - 1
```



问题2：Sorting in linear time (续)

- 你能简述radix sort的执行过程吗？
- 为什么它需要调用一个stable sort？
  能不能改为从高位开始排序？
- 你如何理解
  We have some flexibility in how to break each key into digits.
- 它的使用有哪些局限性（缺点）？

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



问题3：selection problem

- 什么是选择问题？

- 找到最大元或最小元，需要比较多少次？为什么？
- 找到最大元和最小元，需要比较多少次？为什么？
- 找到第二大元，需要比较多少次？为什么？

问题3：selection problem

- 什么是选择问题？

Input: A set A of n (distinct) numbers and an integer i, with $1 \leq i \leq n$.

Output: The element $x \in A$ that is larger than exactly $i - 1$ other elements of A.

- 找到最大元或最小元，需要比较多少次？为什么？
- 找到最大元和最小元，需要比较多少次？为什么？
- 找到第二大元，需要比较多少次？为什么？

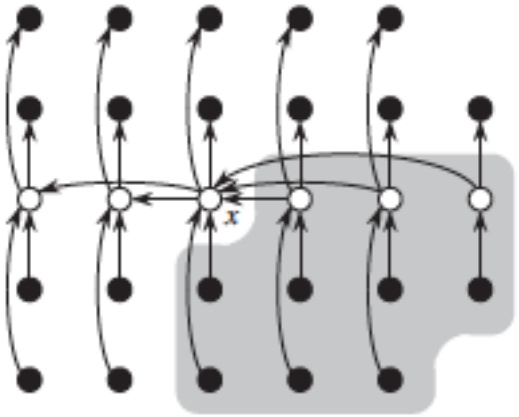
问题3：selection problem (续)

- 你能简述RANDOMIZED-SELECT的执行过程吗？
- 它的best case和worst case分别是什么？
- 你会把递归改为迭代吗？

```
RANDOMIZED-SELECT(A, p, r, i)
1  if p == r
2       return A[p]
3  q = RANDOMIZED-PARTITION(A, p, r)
4  k = q − p + 1
5  if i == k            // the pivot value is the answer
6       return A[q]
7  elseif i < k
8       return RANDOMIZED-SELECT(A, p, q − 1, i)
9  else return RANDOMIZED-SELECT(A, q + 1, r, i − k)
```

问题3：selection problem (续)

- 你能简述SELECT的执行过程吗？
- 如果每组7个元素行不行？3个行不行？



$$3\left(\left\lceil\frac{1}{2}\left\lceil\frac{n}{5}\right\rceil\right\rceil-2\right)\geq\frac{3n}{10}-6\,.$$

$$T(n)\leq\begin{cases}O(1) & \text{if } n<140\,,\\ T(\lceil n/5\rceil)+T(7n/10+6)+O(n) & \text{if } n\geq 140\,.\end{cases}$$

Weighted Median

- For n distinct elements $x_1, x_2, \ldots, x_n$
  - Positive weights $w_1, w_2, \ldots, w_n$
  - $\sum_{i=1}^{n} w_i = 1$
- Weighted (lower) median: x_k satisfying
  - $\sum_{x_i < x_k} w_i < \frac{1}{2}$ and
  - $\sum_{x_i > x_k} w_i \leq \frac{1}{2}$

赛马问题

- 共有25匹马，每次可选5匹马进行比赛，并得到次序（无法计时）。问至少要比赛多少次才能确定跑得最快、次快和第三快的三匹马，并证明其最优性。

- 提示
  - 类比于寻找最大值、最小值、第二大值
  - 信息单元

*k*th Largest Element in Two Arrays

- Given two sorted arrays with n and m elements respectively, design an algorithm to find the kth largest element in the totally $(m+n)$ elements in $O(\log m + \log n)$ and explain the time complexity.

最小未出现自然数

- n个大小各不相同的自然数，找出不在这个自然数序列中出现的最小自然数。
  - “1、2、4、5”中最小未出现是3。

- 分别就下面两种情况设计算法
  - 若n个元素是已排序的
  - 若n个元素是未排序的

Finding the “Heavy” Element

- Find the element *i* with *freq(i)* > *n*/2 in an array of *n* elements. Here, *freq(i)* is defined as the number of occurrence of *i* in the array.