计算机问题求解－论题2-1

－ 算法问题与解题的算法

2017年02月25日

Part I
Algorithmic Problem

问题1:

什么是a well-specified computational problem?

具体地说，就“解一元二次方程”而言，我们这里的“求解”与你中学学的“求解”有什么不同吗？

排序问题

Input: A sequence of n numbers $\langle a_1, a_2, \ldots, a_n \rangle$.

Output: A permutation (reordering) $\langle a'_1, a'_2, \ldots, a'_n \rangle$ of the input sequence such that $a'_1 \leq a'_2 \leq \cdots \leq a'_n$.

问题2:

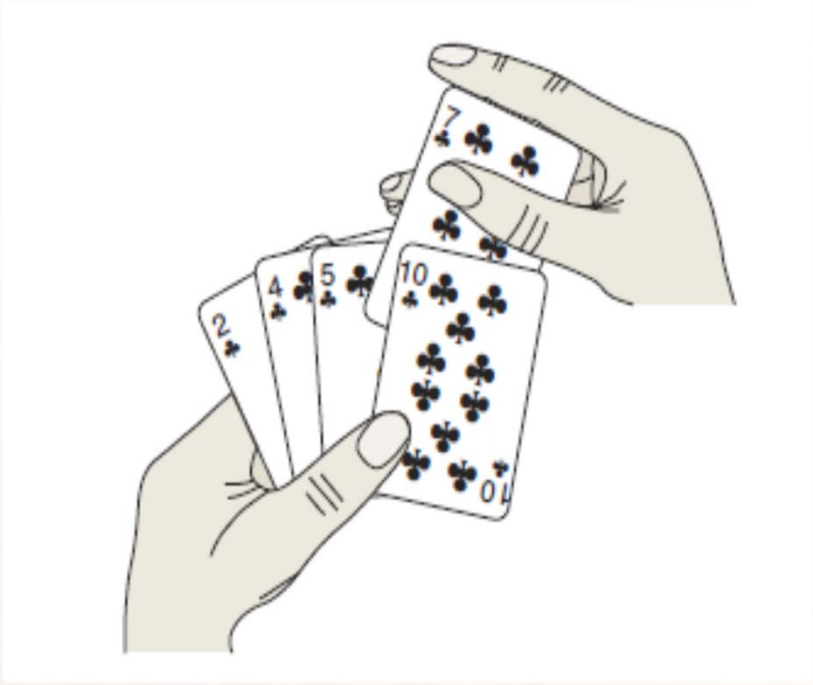
有多少个不同的instances,
它们有好坏之分吗?

算法“实现”输入/输出之间的关系

- 给定一个问题，我们如何讨论一个算法:
  - 基本思路
  - 过程描述
  - 证明其正确
  - 讨论其效率

This is the framework used throughout the courses to think about the design and analysis of algorithms.

基本思路－有时非常简单！



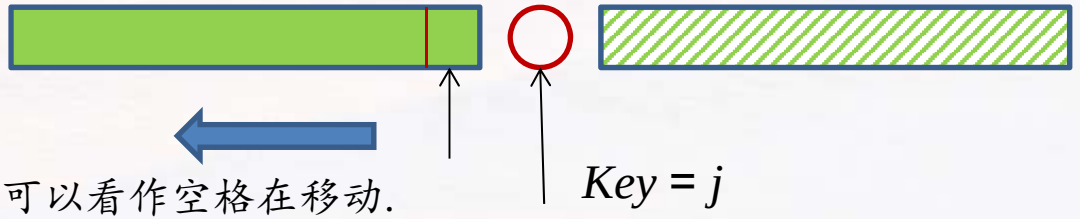
问题3：
你能否描述一下“插入排序”的基本思路与很多人玩牌时的习惯做法之间的关联？

Procedure in Pseudocode

```
INSERTION-SORT(A)
1  for j = 2 to A.length
2      key = A[j]
3      // Insert A[j] into the sorted sequence A[1 .. j − 1].
4      i = j − 1
5      while i > 0 and A[i] > key
6          A[i + 1] = A[i]
7          i = i − 1
8      A[i + 1] = key
```

问题4:

你能解释一下6，8两行的A[*i*+1]吗？



问题5:

什么是loop invariant,它有什么作用?

插入排序过程中的循环不变量

At the start of each iteration of the **for** loop of lines 1–8, the subarray $A[1..j-1]$ consists of the elements originally in $A[1..j-1]$, but in sorted order.

不变是相对于“变”而言的，变的是什么呢？

为什么插入排序算法是正确的?

可以借助于循环不变量来证明:

Initialization: It is true prior to the first iteration of the loop.

Maintenance: If it is true before an iteration of the loop, it remains true before the next iteration.

Termination: When the loop terminates, the invariant gives us a useful property that helps show that the algorithm is correct.

问题6:

你能否针对插入排序给出具体的描述?

问题7：

利用loop invariant证明算法正确与用数学归纳法证明数学命题正确有什么异同？

Part I
算法的时间代价分析

用于算法分析的算法实现模型

Before we can analyze an algorithm, we must have a model of the implementation technology that we will use, including a model for the resources of that technology and their costs. For most of this book, we shall assume a generic one-processor, ***random-access machine (RAM)*** model of computation as our implementation technology and understand that our algorithms will be implemented as computer programs. In the RAM model, instructions are executed one after another, with no concurrent operations.

问题8：为什么“必须”？

The RAM model contains instructions commonly found in real computers: arithmetic (such as add, subtract, multiply, divide, remainder, floor, ceiling), data movement (load, store, copy), and control (conditional and unconditional branch, subroutine call and return). Each such instruction takes a constant amount of time.

问题9：

关于上述模型中的操作和数据，有什么限制吗？限制的原则是什么？

操作必须是”常量时间“内完成, 什么意思?

INSERTION-SORT(A)	*cost*	*times*
1 **for** $j = 2$ **to** $A.length$	c_1	n
2 $key = A[j]$	c_2	$n-1$
3 // Insert $A[j]$ into the sorted sequence $A[1..j-1]$.	0	$n-1$
4 $i = j-1$	c_4	$n-1$
5 **while** $i > 0$ and $A[i] > key$	c_5	$\sum_{j=2}^{n} t_j$
6 $A[i+1] = A[i]$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
7 $i = i-1$	c_7	$\sum_{j=2}^{n} (t_j - 1)$
8 $A[i+1] = key$	c_8	$n-1$

顺便问一句：
为什么是n，
而不是n-1？

$$T(n) = c_1 n + c_2(n-1) + c_4(n-1) + c_5 \sum_{j=2}^{n} t_j + c_6 \sum_{j=2}^{n} (t_j - 1)$$
$$+ c_7 \sum_{j=2}^{n} (t_j - 1) + c_8(n-1) .$$

可是 t_j 是多少？

问题10：

你能解释worst-case和average-case吗？

它们是问题固有的吗？

没有必要算这么复杂

$$
\begin{aligned}
T(n) &= c_1 n + c_2(n-1) + c_4(n-1) + c_5\left(\frac{n(n+1)}{2} - 1\right) \\
&\quad + c_6\left(\frac{n(n-1)}{2}\right) + c_7\left(\frac{n(n-1)}{2}\right) + c_8(n-1) \\
&= \left(\frac{c_5}{2} + \frac{c_6}{2} + \frac{c_7}{2}\right)n^2 + \left(c_1 + c_2 + c_4 + \frac{c_5}{2} - \frac{c_6}{2} - \frac{c_7}{2} + c_8\right)n \\
&\quad - (c_2 + c_4 + c_5 + c_8) .
\end{aligned}
$$

We can express this worst-case running time as $an^2 + bn + c$ for constants a, b, and c that again depend on the statement costs c_i; it is thus a ***quadratic function*** of n.

更进一步：$O(n^2)$

问题11：

你能理解为什么插入排序算法的复杂度必然是平方量级的吗？

“逆序”

- 如果输入序列没有重复元素，不妨假设输入就是 $\{1,2,\ldots,n\}$ 的某种排列;
- 所谓“逆序”是指在输入序列中存在一对元素（不一定相邻）： $<x_i, x_j>$ 满足 $x_i>x_j$, but $i<j$;
- 显然，排序的任务就是消除所有的“逆序”。

- 两个相关的问题：
  - 输入中最多可能有多少个“逆序”？
  - 算法中关键运算（比如：比较运算）次数与逆序消除的个数是什么关系？

同样的问题,不同的方法

```
MERGE-SORT(A, p, r)
1  if p < r
2      q = ⌊(p + r)/2⌋
3      MERGE-SORT(A, p, q)
4      MERGE-SORT(A, q + 1, r)
5      MERGE(A, p, q, r)
```

Divide-and-Conquer
Recursion

To sort the entire sequence $A = \langle A[1], A[2], \ldots, A[n] \rangle$, we make the initial call MERGE-SORT$(A, 1, A.length)$, where once again $A.length = n$.

问题12：

为什么这个算法是正确的？

```
MERGE(A, p, q, r)
1  n1 = q − p + 1
2  n2 = r − q
3  let L[1..n1 + 1] and R[1..n2 + 1] be new arrays
4  for i = 1 to n1
5      L[i] = A[p + i − 1]
6  for j = 1 to n2
7      R[j] = A[q + j]
8  L[n1 + 1] = ∞
9  R[n2 + 1] = ∞
10 i = 1
11 j = 1
12 for k = p to r
13     if L[i] ≤ R[j]
14         A[k] = L[i]
15         i = i + 1
16     else A[k] = R[j]
17         j = j + 1
```

将递归得到的两部分放入两个新数组。

将两个新数组的内容按照大小放入原数组。

问题13：

你能否先解释p,q,r和循环变量k?

循环部分的循环不变量是什么？

At the start of each iteration of the **for** loop of lines 12–17, the subarray $A[p..k-1]$ contains the $k-p$ smallest elements of $L[1..n_1+1]$ and $R[1..n_2+1]$, in sorted order. Moreover, $L[i]$ and $R[j]$ are the smallest elements of their arrays that have not been copied back into A.

Mergesort的代价：递归式

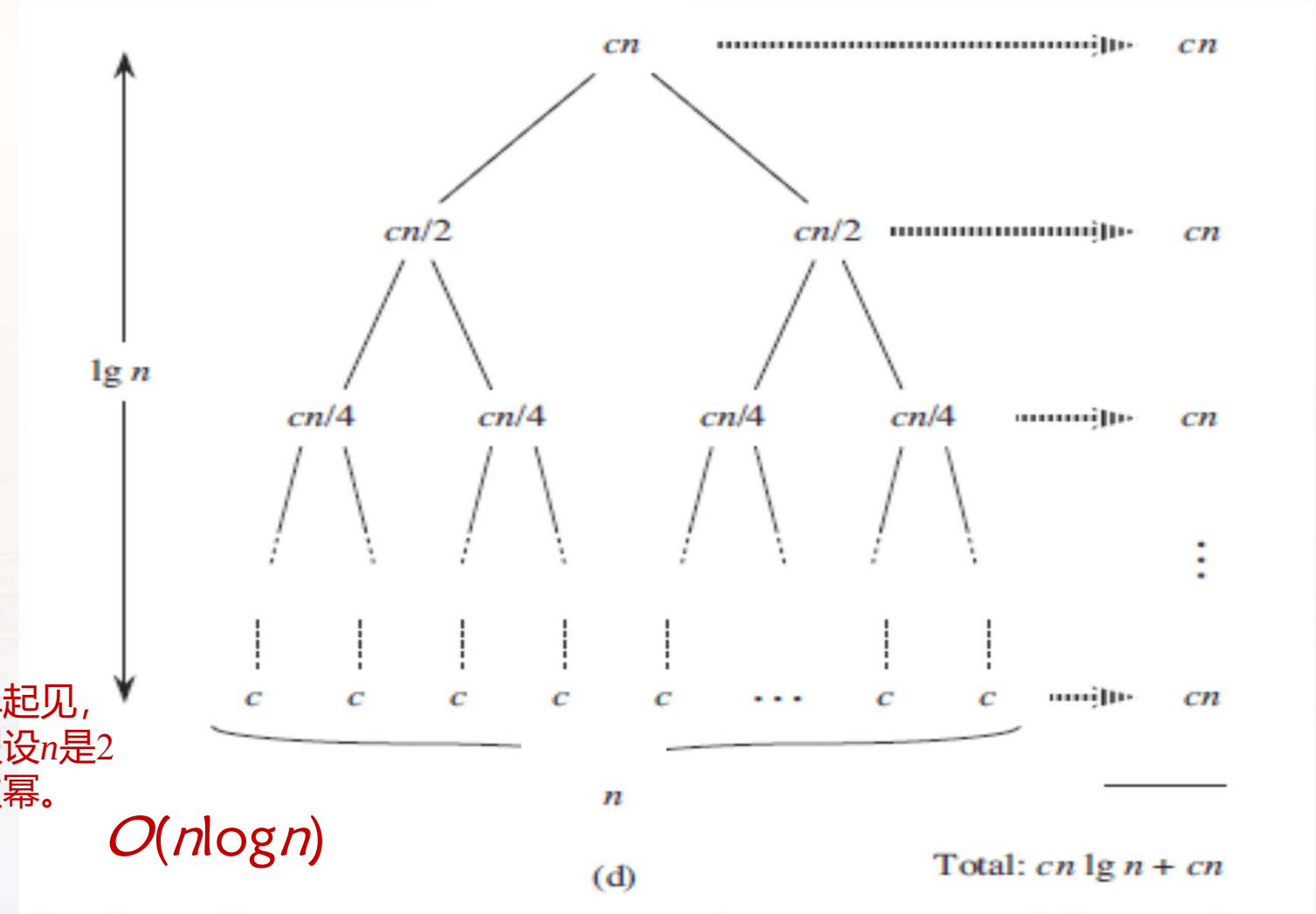
Divide: The divide step just computes the middle of the subarray, which takes constant time. Thus, $D(n) = \Theta(1)$.

Conquer: We recursively solve two subproblems, each of size $n/2$, which contributes $2T(n/2)$ to the running time.

Combine: We have already noted that the MERGE procedure on an n-element subarray takes time $\Theta(n)$, and so $C(n) = \Theta(n)$.

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1\,, \\ 2T(n/2) + \Theta(n) & \text{if } n > 1\,. \end{cases}$$

合并排序效率比插入排序高



为简单起见，
不妨假设n是2
的整次幂。

$O(n\log n)$

课外作业

- TC prob.2-1 – 2-4;
- TC prob.3-2 – 3-4

你应该理解下堂课我们
为什么该讨论计数了？