分治法

Jun Wu

wujun@yzu.edu.cn

March 14, 2018

分治法设计算法的步骤：

- Divide 将问题分成多个子问题；
- Conquer 递归地解决每个子问题；
- Combine将子问题的解合并成原问题的解；

分治算法的分析方法：

- 根据Divide过程划分的子问题数目及子问题的规模，列出相应的递归方程；
- 求解递归方程，得到算法的复杂度。

Problem: (折半查找)

Instance: 有序数组和查找目标;
Output: 目标是否在数组中。

1. **Divide:** Check middle element.
2. **Conquer:** Recursively search 1 subarray.
3. **Combine:** Trivial.

3 5 7 8 9 12 15

(a)

3 5 7 8 9 12 15

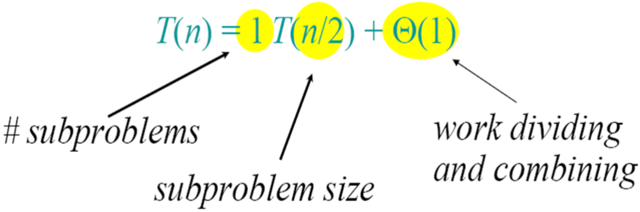
(b)

3 5 7 8 9 12 15

(c)

图: 折半查找

折半查找复杂度分析



- Master定理
- 由Master定理的Case2，知$T(n) = \Theta(\log n)$。

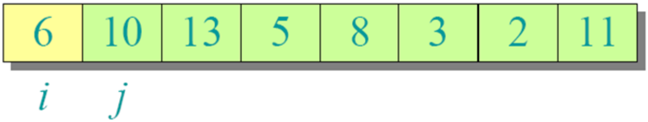
基本思路

Quick Sort:

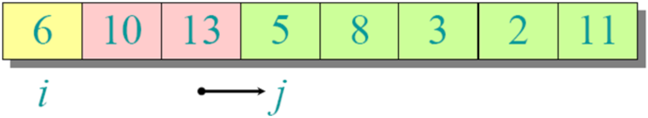
1. **Divide:** 按照某个“pivot”将数组划分成两部分；

2. **Conquer:** Recursively sort 2 subarrays.
3. **Combine:** Trivial.

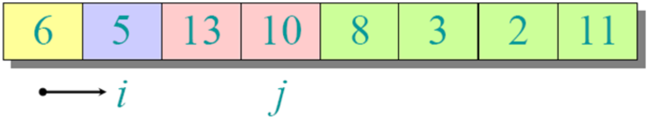
Partitioning subroutine



Partitioning subroutine

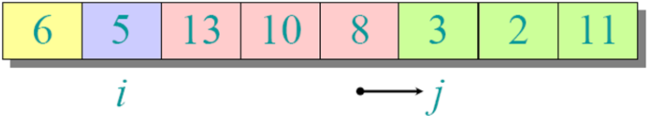


Partitioning subroutine

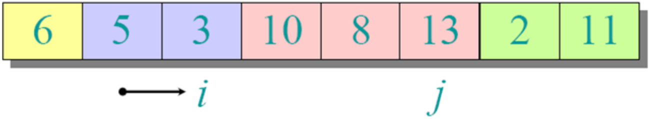


Partitioning subroutine

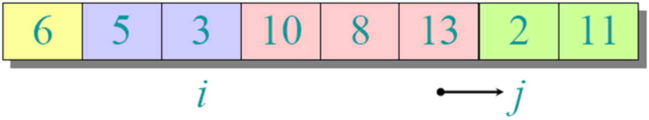
Partitioning subroutine



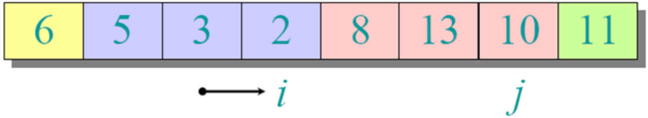
Partitioning subroutine



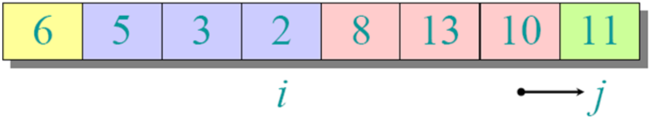
Partitioning subroutine



Partitioning subroutine

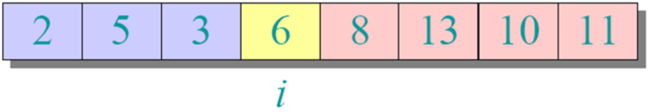


Partitioning subroutine




6
5
3
2
8
13
10
11
i
j

Partitioning subroutine



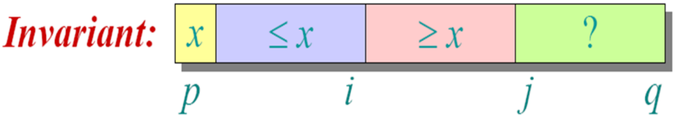
Partition subroutine

PARTITION(A, p, r)

```
x ← A[p]
i ← p
for j ← p + 1 to r do
    if A[j] ≤ x then
        i ← i + 1
        EXCHANGE A[i] ↔ A[j]
    end if
end for
EXCHANGE A[i] ↔ A[p]
RETURN i
```

The analysis of Partition subroutine

- 正确性：



- 复杂度：

Running time $= O(n)$ for n elements.

快速排序

QUICK-SORT(A, p, r)

```
if p < r then
    q ← PARTITION(A, p, r)
    QUICK-SORT(A, p, q)
    QUICK-SORT(A, q + 1, r)
end if
```

- Initial call: QUICK-SORT$(A, 1, length(A))$.
- 复杂度分析？

最坏情况

- 输入是已经排好序的：升序（或降序）
- Pivot总是最小（或最大）
- Pivot两边总是，一边是0个，一边是n-1个

$$
\begin{aligned}
T(n) &= T(0) + T(n-1) + \Theta(n) \\
&= \Theta(1) + T(n-1) + \Theta(n) \\
&= T(n-1) + \Theta(n) \\
&= \Theta(n^2)
\end{aligned}
$$

最好情况

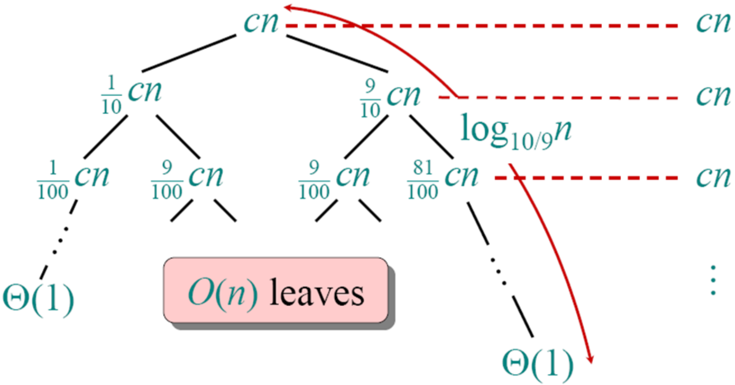
- 每次总是在Pivot两边各分得一半

$$\begin{aligned} T(n) &= 2T(\frac{n}{2}) + \Theta(n) \\ &= \Theta(n \log n) \end{aligned}$$

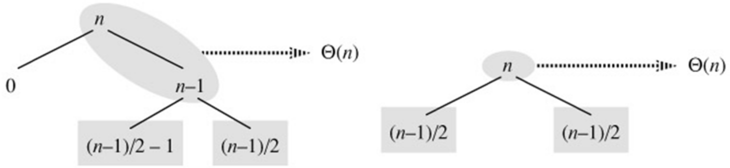
- 较好的情况: 每次总是在Pivot两边分得：$1:9$

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

较好的情况



$$cn\log_{10} n \leq T(n) \leq cn\log_{\frac{10}{9}} n$$



$$\begin{cases} L(n) = 2U(\frac{n}{2}) + \Theta(n) \\ U(n) = L(n-1) + \Theta(n) \end{cases}$$

$$\Downarrow$$

$$\begin{aligned} L(n) &= 2L(\frac{n}{2} - 1) + \Theta(n/2) + \Theta(n) \\ &= \Theta(n \log n) \end{aligned}$$

- How can we make sure we are usually lucky?

Randomized quicksort

- IDEA: 随机选取Pivot
- 运行时间与输入独立
- 没有特殊的某个输入能导致最坏情况
- 无须对输入的分布情况作出假设

RANDOMIZED-PARTITION(A, p, r)

1: $i \leftarrow$ RANDOM(p, r)

2: EXCHANGE $A[p] \leftrightarrow A[i]$

3: **RETURN** PARTITION(A, p, r)

$$T(n) = \begin{cases} T(0) + T(n-1) + \Theta(n), & \text{if } 0 : n-1 \text{ split} \\ T(1) + T(n-2) + \Theta(n), & \text{if } 1 : n-2 \text{ split} \\ \vdots & \\ T(n-1) + T(0) + \Theta(n), & \text{if } n-1 : 0 \text{ split} \end{cases}$$

- 引入特征随机变量X_k:$\mathbb{E}[X_k] = \Pr[X_k = 1] = 1/n$.

$$T(n) = \sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \Theta(n)).$$

Expectation complexity of randomized quicksort

$$\mathbb{E}[T(n)] = \mathbb{E}\left[\sum_{k=0}^{n-1} X_k(T(k) + T(n-k-1) + \Theta(n))\right].$$

- 根据期望的线性特性，

$$\mathbb{E}[T(n)] = \sum_{k=0}^{n-1} \mathbb{E}[X_k(T(k) + T(n-k-1) + \Theta(n))].$$

- 根据Pivot选取的独立性

$$\begin{aligned}
\mathbb{E}[T(n)] &= \sum_{k=0}^{n-1}(\mathbb{E}[X_k]\,\mathbb{E}[(T(k) + T(n-k-1) + \Theta(n))]) \\
&= \frac{1}{n}\sum_{k=0}^{n-1}(\mathbb{E}[T(k) + \mathbb{E}[T(n-k-1)) + \Theta(n) \\
&= \frac{2}{n}\sum_{k=0}^{n-1}\mathbb{E}[T(k)] + \Theta(n)
\end{aligned}$$

Expectation complexity of randomized quicksort

Lemma 1

$$\sum_{k=2}^{n-1} k \log k \le \frac{1}{2}n^2 \log n - \frac{1}{8}n^2.$$

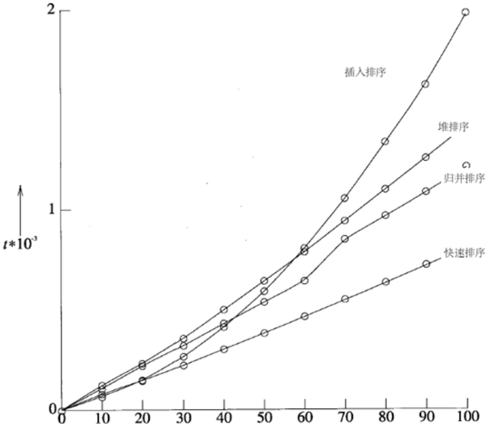
- 猜测$\mathbb{E}[T(n)] = O(n \log n)$，即$\mathbb{E}[T(n)] \le cn \log n$.
- 替换：

$$\begin{aligned}
\mathbb{E}[T(n)] &= \frac{2}{n}\sum_{k=0}^{n-1} \mathbb{E}[T(k)] + \Theta(n) \\
&= \frac{2}{n}\sum_{k=2}^{n-1} \mathbb{E}[T(k)] + \Theta(n) \\
&\le \frac{2c}{n}\left(\frac{1}{2}n^2 \log n - \frac{1}{8}n^2\right) + \Theta(n) \\
&= cn \log n - \left(\frac{cn}{4} - \Theta(n)\right) \\
&\le cn \log n
\end{aligned}$$

排序算法比较

方法	最坏复杂性	平均复杂性
冒泡排序	n^2	n^2
插入排序	n^2	n^2
选择排序	n^2	n^2
堆排序	$n\log n$	$n\log n$
归并排序	$n\log n$	$n\log n$
快速排序	n^2	$n\log n$

排序算法比较



- 是否存在更好的算法？

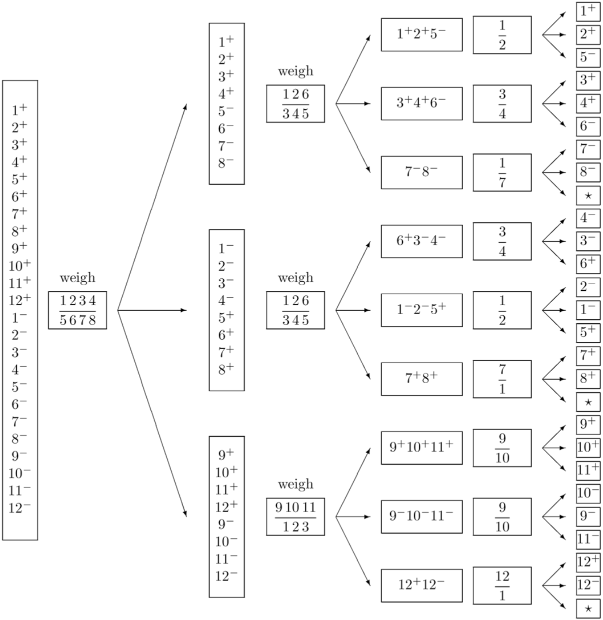
键值比较排序的下界

- n个输入（假设互不相同），输入可能性有n种
- 比较一次最多将状态空间一分为二
- 因此下界为$\Omega(n\log n)$
- 分治法设计出有效算法的关键在于划分是否均衡

Problem: (称重问题)

*12*个同样的球，已知其中只有一个球的重量与其它的不同，称*3*次找出这个球？

称重问题的决策树



weigh
1 2 3 4 / 5 6 7 8
weigh
1 2 6 / 3 4 5
weigh
1 2 6 / 3 4 5
weigh
9 10 11 / 1 2 3
1+2+5−
3+4+6−
7−8−
6+3−4−
1−2−5+
7+8+
9+10+11+
9−10−11−
12+12−

问题定义

Problem: (线性时间选择)

实例： 长度为n的正整数序列A，和整数$k, 1 \leq k \leq n$。

问题： 输出A中第k小的元素。

- 直接的算法？
- 复杂度？
- 更好的算法？回顾Partition过程：

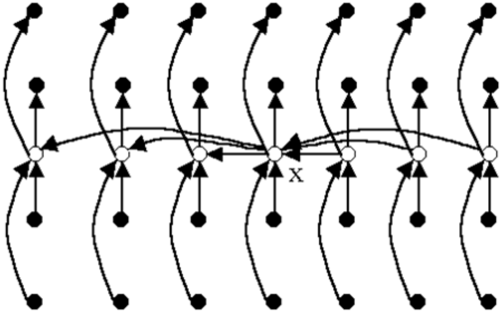
Randomized-Selection

RANDOMIZED-SELECT(A, p, r, k)

```
if p = r then
    RETURN A[p]
end if
q ← RANDOMIZED-PARTITION(A, p, r)
i ← q − p + 1
if i = k then
    RETURN A[i]
end if
if k < i then
    RETURN RANDOMIZED-SELECT(A, p, q − 1, k)
else
    RETURN RANDOMIZED-SELECT(A, q + 1, r, k − i)
end if
```

Deterministic selection

- Randomized selecting 给了我们一个提示：
- 若能找到一个Pivot将数组较均匀的划分开来，就能实现线性时间选择。



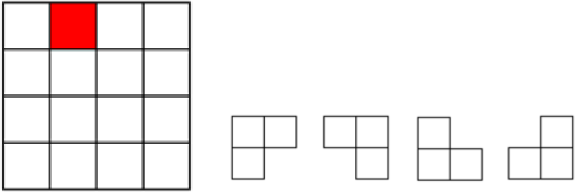
- 将A划分成$\lceil \frac{n}{5} \rceil$组，每组5个元素；
- 将每组排好序：$O(n)$;
- 将每组的第3小元素取出构成数组B;
- 找到B的中位数x: $T(\lceil \frac{n}{5} \rceil)$;
- 以x为Pivot划分A，得到的小数组最多有$\frac{7n}{10}+6$个元素。

$$T(n) = \begin{cases} \Theta(1), & n \leq 140 \\ T(\lceil \frac{n}{5} \rceil) + T(\frac{7n}{10}+6) + \Theta(n), & n > 140 \end{cases}$$

棋盘覆盖问题

Problem:

实例： 给定$2^k \times 2^k$棋盘，如图*a*所示。
问题： 有*4*种类型的骨牌，如图*b*所示。问是否存在所给棋盘的一个完美覆盖？



(a) 棋盘　　(b) 骨牌

图: 棋盘覆盖问题

完美覆盖

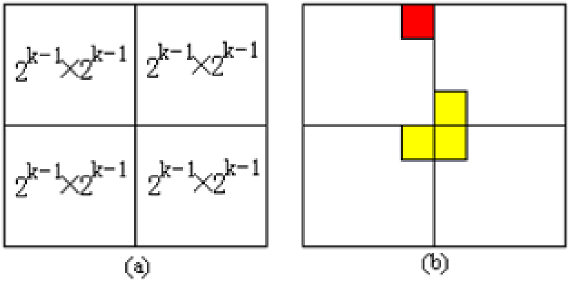
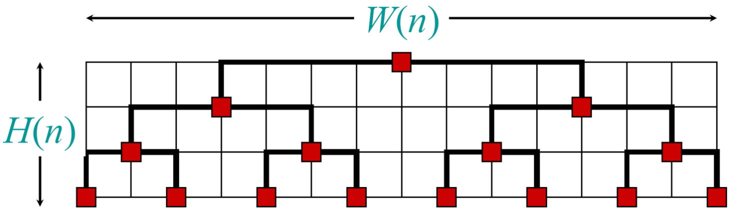


图: 棋盘覆盖的分治算法

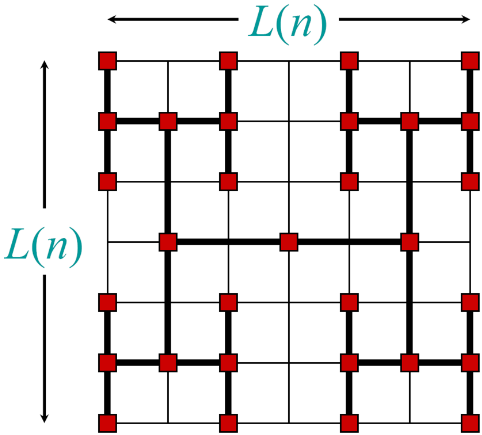
- 复杂度分析:

$$T(k) = \begin{cases} \Theta(1), & k = 1 \\ 4T(k-1) + \Theta(1), & k > 1 \end{cases}$$

- 将n个叶子的二叉树嵌入到网格上
- 有两种布局方式:正常的树形和H-树
- 哪种布局占用面积小?



VLSI布局



大整数乘法

- 问题的规模是整数的长度
- 我们熟悉的算法复杂度$O(n^2)$
- 分治算法：
  1. Divide：将整数X, Y从中间划分开
  $$X = a2^{n/2} + b,\ \ Y = c2^{n/2} + d$$
  2. Conquer：递归调用计算$ac, bd, (a-b)(c-d)$
  3. Combine:
  $$\begin{aligned} X \cdot Y &= (a2^{n/2} + b)(c2^{n/2} + d) \\ &= ac2^n + (ac + bd - (a-b)(c-d))2^{n/2} + bd \end{aligned}$$
- 复杂性分析：
$$T(n) = 3T(n/2) + \Theta(n) = n^{\log 3} = n^{1.59}$$

- 我们熟悉的矩阵乘算法$O(n^3)$
- 分治法：

Divide：

$$\left[\begin{array}{c|c} r & s \\ \hline t & u \end{array}\right] = \left[\begin{array}{c|c} a & b \\ \hline c & d \end{array}\right] \cdot \left[\begin{array}{c|c} e & f \\ \hline g & h \end{array}\right]$$

$$C \quad = \quad A \quad \cdot \quad B$$

$$r = ae + bg$$

$$s = af + bh$$

$$t = ce + dg$$

$$u = cf + dh$$

- 如下定义14个$\frac{n}{2} \times \frac{n}{2}$的矩阵：

	1	2	3	4	5	6	7
$A:$	a	$a+b$	$c+d$	d	$a+d$	$b-d$	$a-c$
$B:$	$f-h$	h	e	$g-e$	$e+h$	$g+h$	$e+f$

- 递归调用计算：

$$P_1 = A_1B_1 = af - ah,\ P_2 = A_2B_2 = ah + bh,\ P_3 = A_3B_3 = ce + de$$

$$P_4 = A_4B_4 = dg - de,\ P_5 = A_5B_5 = ae + ah + de + dh$$

$$P_6 = A_6B_6 = bg + bh - dg - dh,\ P_7 = A_7B_7 = ae + af - ce - cf$$

- 计算r, s, t, u如下：

$$s = P_1 + P_2,\ t = P_3 + P_4$$
$$r = P_5 + P_4 - P_2 + P_6$$
$$u = P_5 + P_1 - P_3 - P_7$$

- 复杂度分析：

$$T(n) = 7T(\frac{n}{2}) + \Theta(n^2) = \Theta(n^{\log 7}) = \Theta(n^{2.81})$$

- 目前复杂度最低的算法$\Theta(n^{2.376})$

最近点对问题

Problem: (最近点对)

实例： 给定平面上的n个点及其坐标。
问题： 寻找其中距离最接近的一对点。

- 这里距离的定义为：

$$p_1 = (x_1, y_1), \quad p_2 = (x_2, y_2)$$
$$d(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

分治算法