

# 第2章 递归与分治策略(4)

线性时间选择

# 内容

## 2.1 求解递归式

## 2.2 递归

1. 阶乘函数
2. Fibonacci数列
3. 排列问题
4. 整数划分问题
5. Hanoi塔问题

## 2.3 分治

1. 二分搜索技术
2. 合并排序
3. 快速排序
4. 线性时间选择

## 2.3 分治

1. 二分搜索技术
2. 合并排序
3. 快速排序
4. 线性时间选择

## 4 线性时间选择

- 给定线性序集中 $n$ 个元素和一个整数 $i$ ,  $1 \leq i \leq n$ , 要求找出这 $n$ 个元素中第 $i$ 小的元素。
- 即如果将这 $n$ 个元素依线性序从小到大排列, 排在第 $i$ 个位置的元素即为要找的元素。
- 特殊情况:
  1. 当 $i=1$ 时, 找的是最小元素;
  2. 当 $i=n$ 时, 找的是最大元素;
  3. 当 $i=(n+1)/2$ 时, 找的是中位数。

# 线性时间选择两种算法

- 根据选取基准元素的方法：
  - a) **RandomizedSelect算法**：划分基准是随机产生的
  - b) **Select算法**：分组得到数组的中位数，以该元素作为划分基准

## a) RandomizedSelect算法

- 模仿快速排序算法，其基本思想是对输入数组进行递归划分，与快速排序算法不同的是，它只对划分出的子数组之一进行递归处理。

# RandomizedSelect算法

1. **分解**: 在 $A[p..r]$ 中随机选出一个元素作为划分基准, 数组 $A[p..r]$ 被划分成两个子数组 $A[p..q-1]$ 和 $A[q+1..r]$ , 使 $A[p..q-1] \leq A[q] \leq A[q+1..r]$ 。
2. **递归求解**:
  - ① 计算子数组 $A[p..q]$ 中元素个数 $k$ 。
  - ② 如果 $i=k$ , 则第 $i$ 小的元素为 **$A[q]$** ;
  - ③ 如果 $i < k$ , 则在子数组 **$A[p..q-1]$** 中寻找第 $i$ 小的元素;
  - ④ 如果 $i > k$ , 则第 $i$ 小的元素在子数组 **$A[q+1..r]$** 中, 且是第 $i-k$ 小元素。
3. **合并**

# RandomizedSelect算法描述

```
1 RANDOMIZED_SELECT(A, p, r, i)
2   if p==r
3       return A[p]
4   q = RANDOMIZED_PARTITION(A, p, r)
5   k = q-p+1
6   if i==k
7       return A[q]
8   else if i < k
9       return RANDOMIZED_SELECT(A, p, q-1, i)
10  else return RANDOMIZED_SELECT(A, q+1, r, i-k)
```



# RandomizedSelect算法分析

- 在最坏情况下，算法RandomizedSelect需要 $O(n^2)$ 计算时间
- 可以证明，在平均情况下，算法RandomizedSelect的时间复杂度为 $O(n)$

## b) Select算法

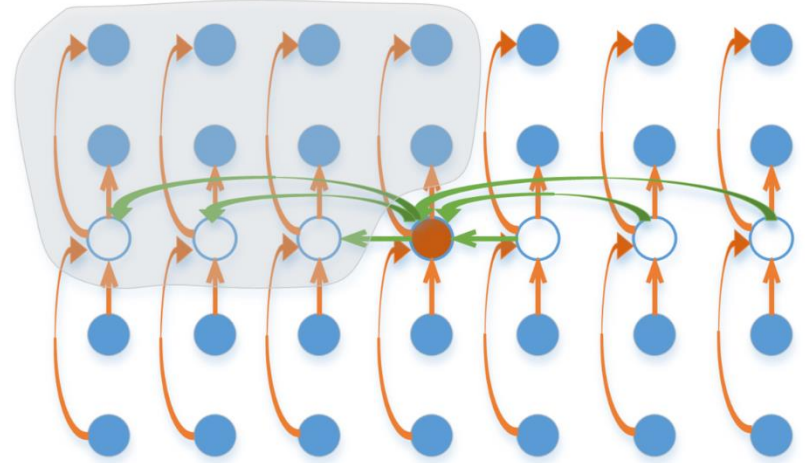
- 如果能在线性时间内找到一个划分基准，使得按这个基准所划分出的2个子数组的长度都至少为原数组长度的 $\varepsilon$ 倍( $0 < \varepsilon < 1$ 是某个正常数)，那么就可以在**最坏情况下**用 $O(n)$ 时间完成选择任务。
- 例如：
  - 若 $\varepsilon = 9/10$ ，算法递归调用所产生的子数组的长度至少缩短 $1/10$ 。所以，在最坏情况下，算法所需的计算时间 $T(n)$ 满足递归式：

$$T(n) \leq T(9n/10) + O(n) \quad \longrightarrow \quad T(n) = O(n)$$

# 选择划分基准

- 分组得到数组的中位数，以该元素作为划分基准
  1. 将 $n$ 个输入元素划分成 $\lfloor n/5 \rfloor$ 个组，每组5个元素。
  2. 用任何一种排序算法，将每组中的元素排好序，并取出每组的中位数，共 $\lfloor n/5 \rfloor$ 个。
  3. 递归调用Select来找出这 $\lfloor n/5 \rfloor$ 个元素的中位数，以这个元素作为划分基准。如果 $\lfloor n/5 \rfloor$ 是偶数，就找它的2个中位数中较小的一个。

# 选择划分基准分析



- 设所有元素互不相同
- 找出的基准 $x$ 至少比下式计算出的元素数大：

$$3 * \left\lfloor \left\lfloor \frac{n}{5} \right\rfloor / 2 \right\rfloor = 3 \left\lfloor \frac{n}{10} \right\rfloor$$

- 同理，基准 $x$ 也至少比该式计算出的元素数小。
- 而当 $n \geq 50$ 时， $3 \left\lfloor \frac{n}{10} \right\rfloor \geq \frac{n}{4}$

所以按此基准划分所得的2个子数组的长度都至少缩短  
 $1/4$ 。

# Select算法描述和分析(1)

**SELECT**(A, p, r, i)

$n = r - p + 1$

if  $n < 50$   在作业中, 这个条件改为  $n \leq 5$

**SORT**(A, p, r)

return A[p+i-1]

向下取整得到组数

Divide the  $n$  elements of the input array into  $\lfloor n/5 \rfloor$  groups of 5 elements each and find the median of each of the  $\lfloor n/5 \rfloor$  groups.

Use **SELECT** recursively to find the **median**  $x$  of the  $\lfloor n/5 \rfloor$  medians.


...

当组数为偶数时, 比如8, 则  
中位数为第4小元素

# Select算法描述(2)

**SELECT**(A, p, r, i)

...

exchange A[r] with x 

$q \leftarrow \text{PARTITION}(A, p, r)$

$k = q - p + 1$

if  $i == k$

    return A[q]

elseif  $i < k$

    return **SELECT**(A, p, q-1, i)

else return **SELECT**(A, q+1, r, i-k)

**SELECT()**函数，不仅要返回第i小元素，还要返回这个第i小元素在A中的位置。

# Select算法复杂度分析(1)

$T(n)$     **SELECT**(A, p, r, i)

$\Theta(1)$      $\left\{ \begin{array}{l} n = p - r + 1 \\ \text{if } n < 50 \\ \quad \text{SORT}(A, p, r) \\ \quad \text{return } A[p+i-1] \end{array} \right.$

$\Theta(n)$  — Divide the  $n$  elements of the input array into  $\lfloor n/5 \rfloor$  groups of 5 elements each and find the median of each of the  $\lfloor n/5 \rfloor$  groups.

$T(n/5)$  — Use **SELECT** recursively to find the median  $x$  of the  $\lfloor n/5 \rfloor$  medians.

...

# Select算法复杂度分析(2)

**SELECT**(A, p, r, i)

...

exchange A[r] with x

$\Theta(n)$

$q \leftarrow \mathbf{PARTITION}(A, p, r)$

$k = q - p + 1$

if  $i == k$

return A[q]

$T(3n/4)$

elseif  $i < k$

return **SELECT**(A, p, q-1, i)

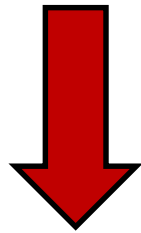
else return **SELECT**(A, q+1, r, i-k)



# Select算法复杂度分析(3)

- 递归式:

$$T(n) = \begin{cases} \Theta(1) & \text{if } n < 50 \\ T(n/5) + T(3n/4) + \Theta(n) & \text{if } n \geq 50 \end{cases}$$



$$T(n) = O(n)$$

# Select算法思考

- 在select算法中，将每一组的大小定为5，并选取50作为是否进行递归调用的分界点，这两点保证了select算法的时间复杂度为 $O(n)$ 。
- 是否有其他选择？

# 快速排序改进

- 如何才能使快速排序在最坏情况下以  $O(n \log n)$  时间运行？
- 每次对待排序的子数组  $A[p..r]$  进行分解时，都使用 Select 算法，选择待排序的  $(r-p+1)$  个元素的中位数作为基准元素。

# 快速排序改进算法描述

```
1 BEST_CASE_QUICKSORT(A, p, r)
2   if p < r
3       i = floor((r-p+1)/2)
4       x = SELECT(A, p, r, i) // x是A[p..r]中位数
5       exchange A[r] with x
6       q=PARTITION(A, p, r)
7       BEST_CASE_QUICKSORT(A, p, q-1)
8       BEST_CASE_QUICKSORT(A, q+1, r)
```

End