Open Topic: Randomized Selection 快速选择算法

韩博

计算机科学与技术系

2018-05-07

目录

什么是快速选择

- 顾名思义，Randomized(用随机化方法) Selection(选择)。

什么是快速选择

- 顾名思义，Randomized(用随机化方法) Selection(选择)。
- 那你为什么不翻译成：随机选择？

什么是快速选择

- 顾名思义，Randomized(用随机化方法) Selection(选择)。
- 那你为什么不翻译成：随机选择？
- 因为这和快速排序完全类似。

什么是快速选择

- 顾名思义，Randomized(用随机化方法) Selection(选择)。
- 那你为什么不翻译成：随机选择？
- 因为这和快速排序完全类似。
- 而且维基百科也这么命名

中文维基百科Facebook粉丝专页正式上线，邀请大家一同关注。

快速选择 [编辑]

维基百科，自由的百科全书

在计算机科学中，**快速选择**（英语：Quickselect）是一种从无序列表找到第k小元素的选择算法。它从原理上来说与快速排序有关。与快速排序一样都由托尼·霍尔提出的，因而也被称为**霍尔选择算法**。[1] 同样地，它在实际应用是一种高效的算法，具有很好的平均时间复杂度，然而最坏时间复杂度则不理想。快速选择及其变种是实际应用中最常使用的高效选择算法。

快速选择的总体思路与快速排序一致，选择一个元素作为基准来对元素进行分区，将小于和大于基准的元素分在基准左边和右边的两个区域。不同的是，快速选择并不递归访问双边，而是只递归进入一边的元素中继续寻找。这降低了平均时间复杂度，从$O(n \log n)$至$O(n)$，不过最坏情况仍然是$O(n^2)$。

与快速排序一样，快速选择一般是以原地算法的方式实现，除了选出第k小的元素，数据也得到了部分地排序。

算法介绍

- 完整的说法是
  对于一个数组中的n个两两不同(distinct)元素

$$A[1..n]$$

  请找出第k大的。$(1 \leq k \leq n)$

算法介绍

- 常言道：不要重复发明轮子

算法介绍

- 常言道：不要重复发明轮子
- 直接上ITA截图

```
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)
```

算法介绍

- 我假设各位对快速排序了如指掌

算法介绍

- 我假设各位对快速排序了如指掌
- 不再详细介绍Randomized-Partition内部了

期望复杂度分析

- 下面开始分析期望复杂度

一个显然的事实

这个算法中，p和r对时间开销的影响不是本征的
而$n = r - p$才是本征的

```
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)
```

开始数数

第1, 2行，边界：$n = 1$时耗费为$\Theta(1)$.

```
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)
```

开始数数

第3行，随机划分：耗费为严格线性，即$\Theta(n)$

```
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)
```

开始数数

第5,6行，第一种递归情况：恰好找到，递归直接结束
因为Randomized-Partition是均匀的随机
这种情况发生的概率是

$$p_1 = \frac{1}{n}$$

```
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)
```

开始数数

我们合并分析第7, 8, 9行，第二类递归情况：须继续
首先这种情况发生的概率是

$$p_2 = 1 - \frac{1}{n}$$

```
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)
```

开始数数

前面已经说到，一个本征影响时间耗费的量是n
我们来看递归后n'的情况

```
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)
```

开始数数

暂且只认为时间耗费是n的函数$T(n)$
当$i<q$时，下一层递归是

$$T(n-i)$$

```
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)
```

开始数数

当$i > q$时，下一层递归是

$$T(i-1)$$

```
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)
```

麻烦来了...

- 我们看到以下事实

麻烦来了...

- 我们看到以下事实
- n'不仅受到n的影响，还受到i的影响

麻烦来了...

- 我们看到以下事实
- n'不仅受到n的影响，还受到i的影响
- 所以认为时间耗费可以用$T(n)$表征是错误的

麻烦来了...

- 我们看到以下事实
- n'不仅受到n的影响，还受到i的影响
- 所以认为时间耗费可以用$T(n)$表征是错误的
- 须用$T(n,i)$表达

必须吗？

- 必须用$T(n,i)$吗？

必须吗？

- 必须用$T(n,i)$吗？
- （那不要分析了，两维递归式我解不出来）

必须吗？

- 必须用$T(n,i)$吗？
- （那不要分析了，两维递归式我解不出来）
- 我们需要再来一个“注意到...”

注意到

- 注意到算法的时间复杂度指的是:

注意到

- 注意到算法的时间复杂度指的是：
- 最坏输入下的最好时间

注意到

- 注意到算法的时间复杂度指的是：
- 最坏输入下的最好时间
- emmmmm. . . .

注意到

- 注意到算法的时间复杂度指的是：
- 最坏输入下的最好时间
- emmmmm. . . .
- 能不能给一个最“坏”的i?

吃个栗子

当$n=5$时

$$\begin{array}{ll}
 & q \quad 1 \quad ,\ 2 \quad ,\ 3 \quad ,\ 4 \quad ,\ 5 \\
i=1\text{时} & \Theta(1),T(1),T(2),T(3),T(4) \\
i=2\text{时} & T(4),\Theta(1),T(2),T(3),T(4) \\
i=3\text{时} & T(4),T(3),\Theta(1),T(3),T(4) \\
i=4\text{时} & T(4),T(3),T(2),\Theta(1),T(4) \\
i=5\text{时} & T(4),T(3),T(2),T(1),\Theta(1)
\end{array}$$

显然$i=3$时耗费最大，类似的可知$i=\lfloor\frac{n}{2}\rfloor$时耗费最大。

一点也不显然

- 谁说$T(3) \geq T(2)$的？

一点也不显然

- 谁说$T(3) \geq T(2)$的？
- emmmmm. . . .

一点也不显然

- 谁说$T(3) \geq T(2)$的?
- emmmmm. . . .
- 但是直觉上没有问题啊

一点也不显然

- 谁说$T(3) \geq T(2)$的？
- emmmmm. . . .
- 但是直觉上没有问题啊
- 规范性假设：$T(n+1) \geq T(n)$

一点也不显然

- 谁说$T(3) \geq T(2)$的？
- emmmmm. . . .
- 但是直觉上没有问题啊
- 规范性假设：$T(n+1) \geq T(n)$
- 承认这个假设可以带来许多方便

麻烦解决了

当$i<q$时，$T(n,i)$的下一层递归是

$$T(n-i,i-q)$$

当$i>q$时，$T(n,i)$的下一层递归是

$$T(i-1,i)$$

上述两个式子统一为：$T(n)$的下一层递归是

$$T(\max\{n-q,q-1\})\leq T(\lfloor\frac{n}{2}\rfloor)$$

递推一下

$\max\{n-q, q-1\}$的取值是$\{\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor + 1, \cdots, n-1\}$
每个取值对应两个q的取值，而随机分割算法等概率
由全期望公式、期望的线性性质可以得到

$$E[T(n)] = \frac{2}{n} \sum_{j=\lfloor n/2 \rfloor}^{n-1} E[T(j)] + O(n)$$

猜猜猜

- 先解决下界

猜猜猜

- 先解决下界
- 至少要随机划分一次

猜猜猜

- 先解决下界
- 至少要随机划分一次
- 而随机划分是稳妥的$\Theta(n)$

猜猜猜

- 先解决下界
- 至少要随机划分一次
- 而随机划分是稳妥的$\Theta(n)$
- 所以快速选择是$\Omega(n)$

如果下界等于上界，你说多好

- 猜猜猜：快速选择比二分查找更好

如果下界等于上界，你说多好

- 猜猜猜：快速选择比二分查找更好
- （不然干嘛叫快速选择）

如果下界等于上界，你说多好

- 猜猜猜：快速选择比二分查找更好
- （不然干嘛叫快速选择）
- 那么快速选择就是$\Omega(n)$和$o(n\log n)$

如果下界等于上界，你说多好

- 猜猜猜：快速选择比二分查找更好
- （不然干嘛叫快速选择）
- 那么快速选择就是$\Omega(n)$和$o(n\log n)$
- 所以快速选择是$\Theta(n)$吗？

猜出答案后，证明很容易

令n是偶数，干掉下取整
令递归式中$O(n)$渐进不超过dn，代入$E[T(n)] \le cn$.

$$\begin{aligned} E[T(n)] &\le \frac{2}{n}\sum_{j=n/2}^{n-1} cj + dn \\ &= \frac{2}{n}\cdot c\cdot \frac{3n(n/2+1)}{4} + dn \\ &= \left(\frac{3}{4}c + d\right)n + \frac{3}{2}c \\ &= cn - \left(d - \frac{c}{4}\right)n - \frac{3}{2}c \end{aligned}$$

大功告成

只要$c > 4d$，我们就立刻得到

$$E[T(n)] \le cn$$

上界下界都有，证明得

$$E[T(n)] = \Theta(n)$$

另外因为我们假定$T(n+1) \ge T(n)$
根据马老师之前讲到的类似夹逼的证明方法
我们可以证明n是奇数时也成立。

另外，请原谅

我根本不知道这是怎么“精确”算出来的

算法变体 [编辑]

最简单的快速排序变化是每次随机选择基准值，这样可以达到近乎线性的复杂度。更为确定的做法是采用“取三者中位数”[2]的基准值选择策略，这样对已部分排序的数据依然能够达到线性复杂度。但是，特定人为设置的数组在此方法下仍然会导致最差时间复杂度，如大卫·穆塞尔所描述的“取三者中位数杀手”数列，这成为他发表反省式选择算法的动机。

利用中位数的中位数算法，可以在最坏情形下依然保证线性时间复杂度。但是这一方法中的基准值计算十分复杂，实际应用中并不常见。改进方法是在快速选择算法的基础上，使用“中位数的中位数”算法处理极端特例，这样可以保证平均状态与最差情形下的时间复杂度都是线性的，这也是反省式选择算法的做法。

精确计算下，随机选择基准值策略最差会导致$n(2+2\log 2+o(1))\leq 3.4n+o(n)$复杂度。采用Floyd–Rivest算法可以使这一常数进一步减少，在最坏情形下达到 $1.5n+O(n^{1/2})$。

参考文献 [编辑]

1. ^ Hoare, C. A. R. Algorithm 65: Find. Comm. ACM. 1961, **4** (7): 321–322. doi:10.1145/366622.366647.
2. ^ http://stackoverflow.com/questions/7559608/median-of-three-values-strategy

分类： 选择算法

那，见好就收吧

That’s all. Appreciations.