CSE 221: Algorithms

Order statistics

Mumit Khan

Computer Science and Engineering
BRAC University

References

1. T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*. The MIT Press, September 2001.
2. Erik Demaine and Charles Leiserson, *6.046J Introduction to Algorithms*. MIT OpenCourseWare, Fall 2005. Available from: `ocw.mit.edu/OcwWeb/Electrical-Engineering-and-Computer-Science/6-046JFall-2005/CourseHome/index.htm`

Last modified: June 20, 2009

Contents

Contents

Order statistics

Definition

Select the i^{th} smallest of n elements, also known as the element with rank i.

Order statistics

Definition

Select the i^{th} smallest of n elements, also known as the element with rank i.

- The minimum element has rank $i = 1$

Order statistics

Definition

Select the i^{th} smallest of n elements, also known as the element with rank i.

- The minimum element has rank $i = 1$
- The maximum element has rank $i = n$

Order statistics

Definition

Select the i^{th} smallest of n elements, also known as the element with rank i.

- The minimum element has rank $i = 1$
- The maximum element has rank $i = n$
- The median element has rank $\lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$

Order statistics

Definition

Select the i^{th} smallest of n elements, also known as the element with rank i.

- The minimum element has rank $i = 1$
- The maximum element has rank $i = n$
- The median element has rank $\lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$

Naïve algorithm

- Sort, and then get the i^{th} element.

Order statistics

Definition

Select the i^{th} smallest of n elements, also known as the element with rank i.

- The minimum element has rank $i = 1$
- The maximum element has rank $i = n$
- The median element has rank $\lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$

Naïve algorithm

- Sort, and then get the i^{th} element.
- Using $O(n \lg n)$ algorithm (such as heapsort or mergesort, but *not* quicksort), it runs in $O(n \lg n) + \Theta(1)$ or $\boxed{\Theta(n \lg n)}$ time in the worst case.

Order statistics

Definition

Select the i^{th} smallest of n elements, also known as the element with rank i.

- The minimum element has rank $i = 1$
- The maximum element has rank $i = n$
- The median element has rank $\lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$

Naïve algorithm

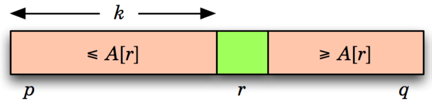
- Sort, and then get the i^{th} element.
- Using $O(n \lg n)$ algorithm (such as heapsort or mergesort, but *not* quicksort), it runs in $O(n \lg n) + \Theta(1)$ or $\boxed{\Theta(n \lg n)}$ time in the worst case.
- Can we do better?

Contents

Idea behind the selection algorithm

Basic idea

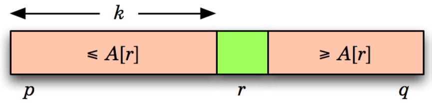
1. Partition the sequence around a pivot, which returns the location of the pivot k after partitioning.



Idea behind the selection algorithm

Basic idea

1. Partition the sequence around a pivot, which returns the location of the pivot k after partitioning.

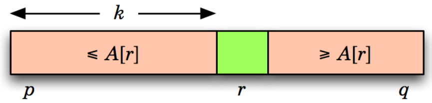


2. If $k = i$, then return the value at that index.

Idea behind the selection algorithm

Basic idea

1. Partition the sequence around a pivot, which returns the location of the pivot k after partitioning.

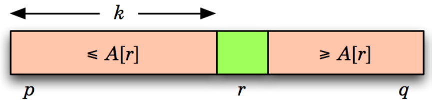


2. If $k = i$, then return the value at that index.
3. Otherwise, if $k > i$, then the i^{th} value must be *left* of k, so recursively partition the subarray left of k until $k = i$.

Idea behind the selection algorithm

Basic idea

1. Partition the sequence around a pivot, which returns the location of the pivot k after partitioning.



2. If $k = i$, then return the value at that index.
3. Otherwise, if $k > i$, then the i^{th} value must be *left* of k, so recursively partition the subarray left of k until $k = i$.
4. Otherwise, if $k > i$, then the i^{th} value must be *right* of k, so recursively partition the subarray left of k until $k = i - k$.

Partition based selection example

Given the following sequence (and using the first element as the *pivot*):

6	10	13	5	8	3	2	11

Partition based selection example

Given the following sequence (and using the first element as the *pivot*):

6	10	13	5	8	3	2	11

Partitioning around the pivot gives ($k = 4$):

2	5	3	6	8	13	10	11

Partition based selection example

Given the following sequence (and using the first element as the *pivot*):

6	10	13	5	8	3	2	11

Partitioning around the pivot gives ($k = 4$):

2	5	3	6	8	13	10	11

Selecting i^{th} element when $i < k$:

2	5	3	6	8	13	10	11

Partition based selection example

Given the following sequence (and using the first element as the *pivot*):

6	10	13	5	8	3	2	11

Partitioning around the pivot gives ($k = 4$):

2	5	3	6	8	13	10	11

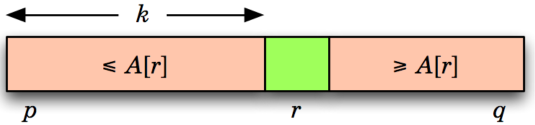
Selecting i^{th} element when $i > k$:

2	5	3	6	8	13	10	11

The selection algorithm

RANDOMIZED-SELECT(A, p, q, i) $\triangleright$ i^{th} smallest of $A[p\,..\,q]$

if $p = q$
  then return $A[q]$
$r \leftarrow$ RANDOMIZED-PARTITION(A, p, q)
$k \leftarrow r - p + 1$ $\triangleright$ $k = rank(A[r])$
if $i = k$
  then return $A[r]$
if $i < k$
  then return RANDOMIZED-SELECT$(A, p, r - 1, i)$
  else return RANDOMIZED-SELECT$(A, r + 1, q, i - k)$



Conclusions

- Choosing a fixed pivot (e.g., the 1^{st} element) leads to a $\Theta(n^2)$ algorithm in the worst case!

Conclusions

- Choosing a fixed pivot (e.g., the 1^{st} element) leads to a $\Theta(n^2)$ algorithm in the worst case! Better to sort and index the i^{th} element then . . .

Conclusions

- Choosing a fixed pivot (e.g., the 1^{st} element) leads to a $\Theta(n^2)$ algorithm in the worst case! Better to sort and index the i^{th} element then . . .
- Choosing a random pivot makes it a $\Theta(n)$ on the average, just like in the case of Quicksort.

Conclusions

- Choosing a fixed pivot (e.g., the 1^{st} element) leads to a $\Theta(n^2)$ algorithm in the worst case! Better to sort and index the i^{th} element then ...
- Choosing a random pivot makes it a $\Theta(n)$ on the average, just like in the case of Quicksort.
- There is however a $\Theta(n)$ algorithm in the worst case, but it has a large hidden constant (*by Blum, Floyd, Pratt, Rivest and Tarjan in 1973*).

Conclusions

- Choosing a fixed pivot (e.g., the 1^{st} element) leads to a $\Theta(n^2)$ algorithm in the worst case! Better to sort and index the i^{th} element then ...
- Choosing a random pivot makes it a $\Theta(n)$ on the average, just like in the case of Quicksort.
- There is however a $\Theta(n)$ algorithm in the worst case, but it has a large hidden constant (*by Blum, Floyd, Pratt, Rivest and Tarjan in 1973*).
- Given this large constant before the n in the linear-time algorithm, the randomized algorithm works better in practice.