Algorithm Design & Analysis (CS3383)[1]

Unit 1 Cont.: Randomized D&C

Rasoul Shahsavarifar

January 28, 2019

[1]Thanks to Dr. Ptricia Evans and Dr. David Bremner at UNB, Dr. Erik Demaine at MIT for sharing the teaching stuffs

Outline[2]

Even More Divide and Conquer

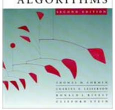
- Quicksort
- Randomized Quicksort
- Randomized median finding

---

[2] Reading:

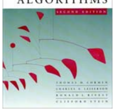
- Main textbook (DPV), Divide and conquer algorithms, Chapter 2 mainly 2.4.
- Algorithms(Cormen): Chapter 5 (5.2, 5.3, and 5.4), Chapter 7, and Chapter 9.
- Recursive algorithms from Jeff Ericson's Algorithm page `http://jeffe.cs.illinois.edu/teaching/algorithms/notes/99-recurrences.pdf`

Contents



Quicksort

- Proposed by C.A.R. Hoare in 1962.
- Divide-and-conquer algorithm.
- Sorts "in place" (like insertion sort, but not like merge sort).
- Very practical (with tuning).



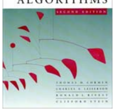
Divide and conquer

Quicksort an n-element array:

1. ***Divide:*** Partition the array into two subarrays around a ***pivot*** x such that elements in lower subarray $\leq x \leq$ elements in upper subarray.

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

Key: *Linear-time partitioning subroutine.*



Partitioning subroutine

PARTITION(A, p, q) $\rhd$ $A[p \,.\,.\, q]$

   $x \leftarrow A[p]$ $\rhd$ pivot $= A[p]$

   $i \leftarrow p$

   for $j \leftarrow p + 1$ **to** q

      do if $A[j] \leq x$

         then $i \leftarrow i + 1$

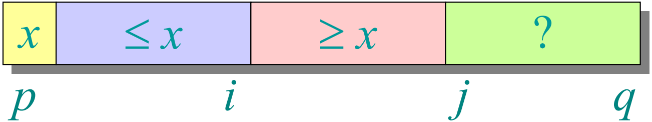
            exchange $A[i] \leftrightarrow A[j]$

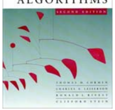
   exchange $A[p] \leftrightarrow A[i]$

   return i

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

Invariant:





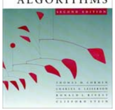
Example of partitioning

6	10	13	5	8	3	2	11
i	j						

Example of partitioning

6	10	13	5	8	3	2	11

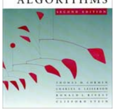
$i \qquad \longrightarrow j$



Example of partitioning

6	10	13	5	8	3	2	11

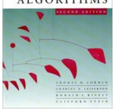
i $\longrightarrow j$



Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11

i $\qquad\qquad j$

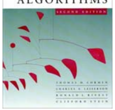


Example of partitioning

6	10	13	5	8	3	2	11

6	5	13	10	8	3	2	11

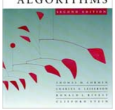
i j



Example of partitioning

6	10	13	5	8	3	2	11

6	5	13	10	8	3	2	11
	i				j		



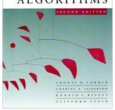
Example of partitioning

6	10	13	5	8	3	2	11

6	5	13	10	8	3	2	11

6	5	3	10	8	13	2	11

$\longrightarrow i$ $\qquad\qquad$ j



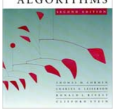
Example of partitioning

6	10	13	5	8	3	2	11

6	5	13	10	8	3	2	11

6	5	3	10	8	13	2	11

i $\quad\quad\quad \longrightarrow j$



Example of partitioning

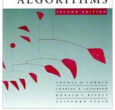
6	10	13	5	8	3	2	11

6	5	13	10	8	3	2	11

6	5	3	10	8	13	2	11

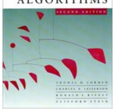
6	5	3	2	8	13	10	11

$\longrightarrow i$ $\qquad\qquad\qquad j$



Example of partitioning

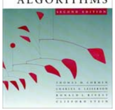
6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11
			i			$\longrightarrow$	j



Example of partitioning

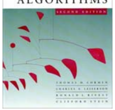
6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11

i $\longrightarrow j$



Example of partitioning

6	10	13	5	8	3	2	11
6	5	13	10	8	3	2	11
6	5	3	10	8	13	2	11
6	5	3	2	8	13	10	11
2	5	3	6	8	13	10	11
			i				



Pseudocode for quicksort

QUICKSORT(A, p, r)

  if $p < r$

    then $q \leftarrow$ PARTITION(A, p, r)

      QUICKSORT$(A, p, q-1)$

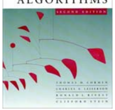
      QUICKSORT$(A, q+1, r)$

Initial call: QUICKSORT$(A, 1, n)$

Analysis of quicksort

- Quicksort is $\Theta(n^2)$ in the worst case. What kind of input is bad? Sorted ($\uparrow$ / $\downarrow$), Array of Same Elements? Why?
- Quicksort is supposed to be fast "in practice".
- We can choose a better pivot in $O(n)$ time, but we'll see it's a bit complicated.
- What if we choose a random element as pivot?

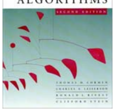
Contents



Randomized quicksort

IDEA: Partition around a ***random*** element.

- Running time is independent of the input order.
- No assumptions need to be made about the input distribution.
- No specific input elicits the worst-case behavior.
- The worst case is determined only by the output of a random-number generator.



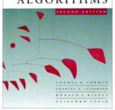
Randomized quicksort analysis

Let $T(n)$ = the random variable for the running time of randomized quicksort on an input of size n, assuming random numbers are independent.

For $k = 0, 1, \ldots, n-1$, define the ***indicator random variable***

$$X_k = \begin{cases} 1 & \text{if PARTITION generates a } k : n-k-1 \text{ split,} \\ 0 & \text{otherwise.} \end{cases}$$

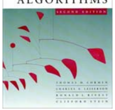
$E[X_k] = \Pr\{X_k = 1\} = 1/n$, since all splits are equally likely, assuming elements are distinct.



Analysis (continued)

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

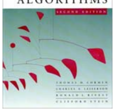
$$= \sum_{k=0}^{n-1} X_k \big(T(k) + T(n-k-1) + \Theta(n)\big)$$



Calculating expectation

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

Take expectations of both sides.

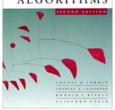


Calculating expectation

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

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

Linearity of expectation.



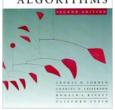
Calculating expectation

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

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

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

Independence of X_k from other random choices.



Calculating expectation

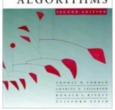
$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k\left(T(k) + T(n-k-1) + \Theta(n)\right)\right]$$

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

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

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

Linearity of expectation; $E[X_k] = 1/n$.



Calculating expectation

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

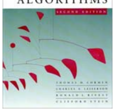
$$= \sum_{k=0}^{n-1} E\left[X_k\left(T(k) + T(n-k-1) + \Theta(n)\right)\right]$$

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

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

$$= \frac{2}{n}\sum_{k=1}^{n-1} E[T(k)] + \Theta(n)$$

Summations have identical terms.



Hairy recurrence

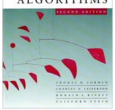
$$E[T(n)] = \frac{2}{n} \sum_{k=2}^{n-1} E[T(k)] + \Theta(n)$$

(The $k = 0, 1$ terms can be absorbed in the $\Theta(n)$.)

Prove: $E[T(n)] \le a n \lg n$ for constant $a > 0$.

- Choose a large enough so that $a n \lg n$ dominates $E[T(n)]$ for sufficiently small $n \ge 2$.

Use fact: $\sum_{k=2}^{n-1} k \lg k \le \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$ (exercise).

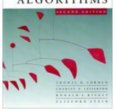


Substitution method

$$E[T(n)] \le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n)$$

$$\le \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n)$$

Use fact.



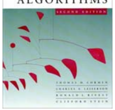
Substitution method

$$E[T(n)] \le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n)$$

$$\le \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n)$$

$$= an\lg n - \left(\frac{an}{4} - \Theta(n)\right)$$

Express as ***desired*** – ***residual***.

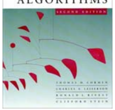


Substitution method

$$
\begin{aligned}
E[T(n)] &\le \frac{2}{n}\sum_{k=2}^{n-1} ak\lg k + \Theta(n) \\
&= \frac{2a}{n}\left(\frac{1}{2}n^2\lg n - \frac{1}{8}n^2\right) + \Theta(n) \\
&= an\lg n - \left(\frac{an}{4} - \Theta(n)\right) \\
&\le an\lg n\,,
\end{aligned}
$$

if a is chosen large enough so that $an/4$ dominates the $\Theta(n)$.

Contents



Order statistics

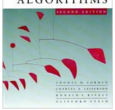
Select the ith smallest of n elements (the element with ***rank*** i).

- $i = 1$: ***minimum***;
- $i = n$: ***maximum***;
- $i = \lfloor (n+1)/2 \rfloor$ or $\lceil (n+1)/2 \rceil$: ***median***.

Naive algorithm: Sort and index ith element.

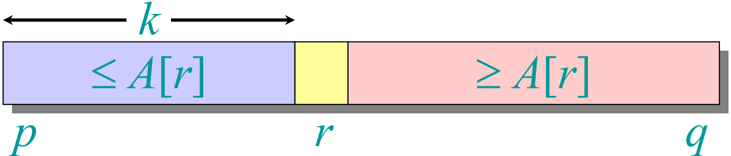
$$\text{Worst-case running time} = \Theta(n \lg n) + \Theta(1) \\ = \Theta(n \lg n),$$

using merge sort or heapsort (*not* quicksort).



Randomized divide-and-conquer algorithm

RAND-SELECT(A, p, q, i) $\triangleright$ ith smallest of $A[p \,.\,.\, q]$
    if $p = q$ **then return** $A[p]$
    $r \leftarrow$ RAND-PARTITION(A, p, q)
    $k \leftarrow r - p + 1$ $\triangleright$ $k = \text{rank}(A[r])$
    if $i = k$ **then return** $A[r]$
    if $i < k$
        then return RAND-SELECT$(A, p, r - 1, i)$
        else return RAND-SELECT$(A, r + 1, q, i - k)$



Example

Select the $i = 7$th smallest:

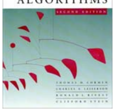
6	10	13	5	8	3	2	11	$i = 7$

pivot

Partition:

2	5	3	6	8	13	10	11	$k = 4$

Select the $7 - 4 = 3$rd smallest recursively.



Intuition for analysis

(All our analyses today assume that all elements are distinct.)

Lucky:

$$T(n) = T(9n/10) + \Theta(n) \qquad n^{\log_{10/9} 1} = n^0 = 1$$

$$= \Theta(n) \qquad \text{Case 3}$$

Unlucky:

$$T(n) = T(n-1) + \Theta(n) \qquad \text{arithmetic series}$$

$$= \Theta(n^2)$$

Worse than sorting!

Randomized median finding

```
Select2(A, p, q, i)
    n <- q - p + 1
    do {
        r <- RandPartition(A,p,q)
        k <- r - p + 1
        if i = k then return A[r]
    } while ((k < n/4) or (k > 3n/4));
    if i < k
        then return Select2 (A, p, r - 1, i )
        else return Select2 (A, r + 1, q, i - k )
```

Exercise: Analyze the randomized median finding.[3]

[3] There is a bonus to do this exercise before I post the solution on D2L (by next lecture)