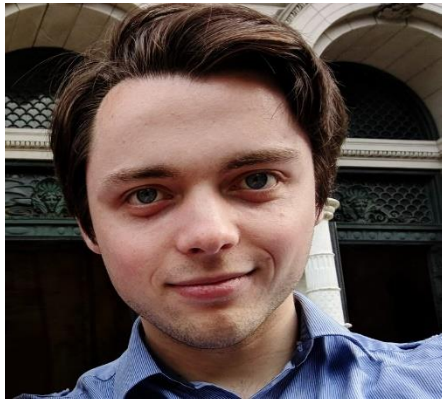
Lecture 1: Introduction and Median Finding

David Woodruff

Staff

Instructors:



Daniel Anderson

David Woodruff

TAs:

Aditya Sundaram

Bill Qin

Davis Wojnovich

Efe Cekirge

Joel Manning

Jonathan Liu

Yoseph Mak

Kunal Joshi

Emma Hu

Magdalen Dobson

Nicholas Grill

Suhas Kotha

Andrew Caosun

Grading and Course Policies

- All available here: https://www.cs.cmu.edu/~15451-f22

6 Written Homeworks	**30% (5% each)**
3 Oral Homeworks	**12% (4% each)**
Recitation Attendance	**3% (see below)**
Midterm exams (two in-class times)	**30% (15% each)**
Final exam	**25%**

- Solve written homeworks individually. Come to office hours or ask questions on Piazza! LaTeX solutions and submit on Gradescope
- Oral homeworks can be solved in groups of up to 3
- Recitation attendance contributes up to 3%. You may miss a small number of recitations and still get full points, but try to come to as many as possible

Homework

- Each HW has 3 problems
- About half the homeworks have a programming problem – submit via Autolab (languages accepted are Java, C, C++, Ocaml, Python, SML)
- For oral HWs you can collaborate, but write the programming problem yourself. Each team has 45 minutes to present the 3 problems. Feel free to use notes!
- Cite any reference material or webpage if you use it
- Late homeworks and "grace/mercy" days – please see the website for details!
- HW1 posted today. Due Sep 7

Goals of the Course

- Design and analyze algorithms!
- Algorithms: dynamic programming, divide-and-conquer, hashing and data structures, randomization, network flows, linear programming, approximation algorithms
- Analysis: recurrences, probabilistic analysis, amortized analysis, potential functions
- New Models: online algorithms, data streams

Guarantees on Algorithms

- Want provable guarantees on the running time of algorithms
- Why?
- Composability: if we know an algorithm runs in time at most T on any input, don't have to worry what kinds of inputs we run it on
- Scaling: how does the time grow as the input size grows?
- Designing better algorithms: what are the most time-consuming steps?

Example: Median Finding

- In the median-finding problem, we have an array of distinct numbers

$$a_1, a_2, \ldots, a_n$$

and want the index i for which there are exactly $\lfloor n/2 \rfloor$ numbers larger than a_i

- *How can we find the median?*
  - Check each item to see if it is the median: $\Theta(n^2)$ time
  - Sort items with MergeSort (deterministic) or QuickSort (randomized): $\Theta(n \log n)$ time
  - Can we find it faster? What about finding the k-th smallest number?

QuickSelect Algorithm to Find the k-th Smallest Number

- Assume $a_1, a_2, \ldots, a_n$ are all distinct for simplicity

- Choose a random element a_i in the list – call this the "pivot"

- Compare each a_j to a_i
  - Let LESS = $\{a_j$ such that $a_j < a_i\}$
  - Let GREATER = $\{a_j$ such that $a_j > a_i\}$

- If $k \leq |LESS|$, find the k-th smallest element in LESS
- If $k = |LESS| + 1$, output the pivot a_i
- Else find the (k-|LESS|-1)-th smallest item in GREATER

- Similar to Randomized QuickSort, but *only recurse on one side!*

Bounding the Running Time

- Theorem: the expected number of comparisons for QuickSelect is at most 4n

- T(n,k) is the expected number of comparisons to find k-th smallest item in an array of length n
    - T(n,k) is the same for any array! Can show by induction (algorithm does not depend on order)
    - Let $\mathrm{T}(\mathrm{n}) = \max_{\mathrm{k}} \mathrm{T}(\mathrm{n}, \mathrm{k})$

- T(n) is a non-decreasing function of n
    - Can show by induction (for any k and any pivot, size of recursive subarray does not decrease)

- Let's show T(n) < 4n by induction

- Base case: T(1) = 0 < 4

- Inductive hypothesis: T(i) < 4i for all $1 \leq \mathrm{i} \leq \mathrm{n} - 1$

Bounding the Running Time

- Suppose we have an array of length n. Assume n is even for the moment
- Pivot randomly partitions the array into two pieces, LESS and GREATER, with |LESS| + |GREATER| = n-1
  - |LESS| is uniform in the set {0, 1, 2, 3, …, n-1}
  - Since $\mathrm{T(i)}$ is non-decreasing with i, to upper bound T(n) we can assume we recurse on larger half
  - $\mathrm{T(n)} \leq \mathrm{n} - 1 + \frac{2}{\mathrm{n}} \sum_{\mathrm{i}=\frac{\mathrm{n}}{2},\ldots,\mathrm{n}-1} \mathrm{T(i)}$

$\leq \mathrm{n} - 1 + \frac{2}{\mathrm{n}} \sum_{\mathrm{i}=\frac{\mathrm{n}}{2},\ldots,\mathrm{n}-1} 4\mathrm{i}$ by inductive hypothesis

$< \mathrm{n} - 1 + 4\left(\frac{3\mathrm{n}}{4}\right)$ since the average $\frac{2}{\mathrm{n}} \sum_{\mathrm{i}=\frac{\mathrm{n}}{2},\ldots,\mathrm{n}-1} \mathrm{i}$ is at most $\frac{\frac{\mathrm{n}}{2}+(\mathrm{n}-1)}{2} < \frac{3\mathrm{n}}{4}$

$< 4\mathrm{n}$ completing the induction

Similar Analysis Holds for Odd n

- Suppose we have an array of length n. Assume n is odd now
- Pivot randomly partitions the array into two pieces, LESS and GREATER, with |LESS| + |GREATER| = n-1
  - The probability the larger of |LESS| and |GREATER| is (n-1)/2 is 1/n
  - The probability the larger of |LESS| and |GREATER| is in {(n+1)/2, …, n-1} is 2/n
  - $T(n) \leq n - 1 + \frac{1}{n} T\left(\frac{n-1}{2}\right) + \frac{2}{n} \sum_{i=\frac{n+1}{2},\ldots,n-1} T(i)$

$\leq n - 1 + \frac{1}{n} \cdot \frac{4(n-1)}{2} + \frac{2}{n} \sum_{i=\frac{n+1}{2},\ldots,n-1} 4i$ by inductive hypothesis

$\leq n - 1 + \frac{1}{n} \cdot \frac{4(n-1)}{2} + \frac{2}{n-1} \sum_{i=\frac{n+1}{2},\ldots,n-1} 4i$ there are (n-1)/2 terms to average so we can still upper bound by the average

$\leq n - 1 + 2 - \frac{2}{n} + 4((n-1) + \frac{n+1}{2})/2$

$< 4n$ completing the induction

What About Deterministic Algorithms?

- Can we get an algorithm which does not use randomness and always performs O(n) comparisons?

- Idea: suppose we could deterministically find a pivot which partitions the input into two pieces LESS and GREATER each of size $\lfloor\frac{n}{2}\rfloor$

- How to do that?

- Find the median and then partition around that
  - Um... finding the median is the original problem we want to solve....

Deterministically Finding a Pivot

- Idea: deterministically find a pivot with O(n) comparisons to partition the input into two pieces LESS and GREATER each of size at least 3n/10-1

- DeterministicSelect:
  1. Group the array into n/5 groups of size 5 and find the median of each group
  2. Recursively, find the median of medians. Call this p
  3. Use p as a pivot to split into subarrays LESS and GREATER
  4. Recurse on the appropriate piece

- Theorem: DeterministicSelect makes O(n) comparisons to find the k-th smallest item in an array of size n

Running Time of DeterministicSelect

- DeterministicSelect:
  1. Group the array into n/5 groups of size 5 and find the median of each group
  2. Recursively, find the median of medians. Call this p
  3. Use p as a pivot to split into subarrays LESS and GREATER
  4. Recurse on the appropriate piece

- Step 1 takes O(n) time since it takes O(1) time to find the median of 5 elements
- Step 2 takes T(n/5) time
- Step 3 takes O(n) time

Claim: |LESS| $\geq 3n/10$-1 and |GREATER| $\geq 3n/10$-1

Running Time of DeterministicSelect

- Claim: |LESS| $\geq 3n/10$-1 and |GREATER| $\geq 3n/10$-1

- **Example 1:** If n = 15, we have three groups of 5:

  {1, 2, 3, 10, 11}, {4, 5, 6, 12, 13}, {7,8,9,14,15}

  medians: 3 6 9

  median of medians p: 6

- There are g = n/5 groups, and at least $\lceil \frac{g}{2} \rceil$ of them have at least 3 elements at most p. The number of elements less than or equal to p is at least

$$3 \left\lceil \frac{g}{2} \right\rceil \geq \frac{3n}{10}$$

- Also at least 3n/10 elements greater than or equal to p

Running Time of DeterministicSelect

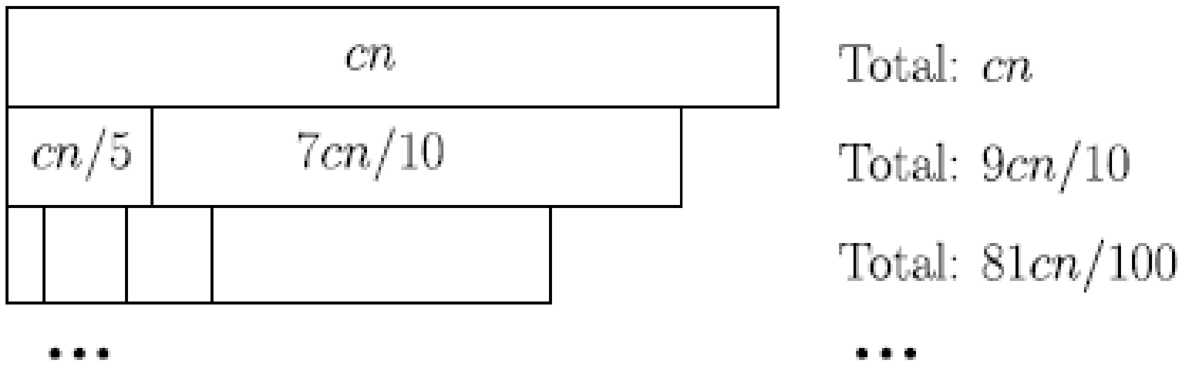
- DeterministicSelect:
  1. Group the array into n/5 groups of size 5 and find the median of each group
  2. Recursively, find the median of medians. Call this p
  3. Use p as a pivot to split into subarrays LESS and GREATER
  4. Recurse on the appropriate piece

- Steps 1-3 take O(n) + T(n/5) time
- Since |LESS| $\geq 3n/10$-1 and |GREATER| $\geq 3n/10$-1, Step 4 takes at most T(7n/10) time

- So $T(n) \leq cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$, for a constant c > 0

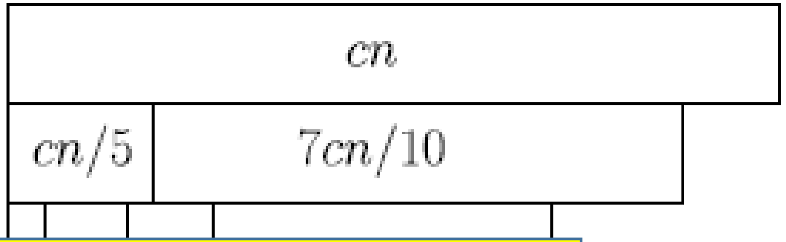
Running Time of DeterministicSelect

- $T(n) \leq cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$



Running Time of DeterministicSelect

- $T(n) \le cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$



Total: cn

Total: $9cn/10$

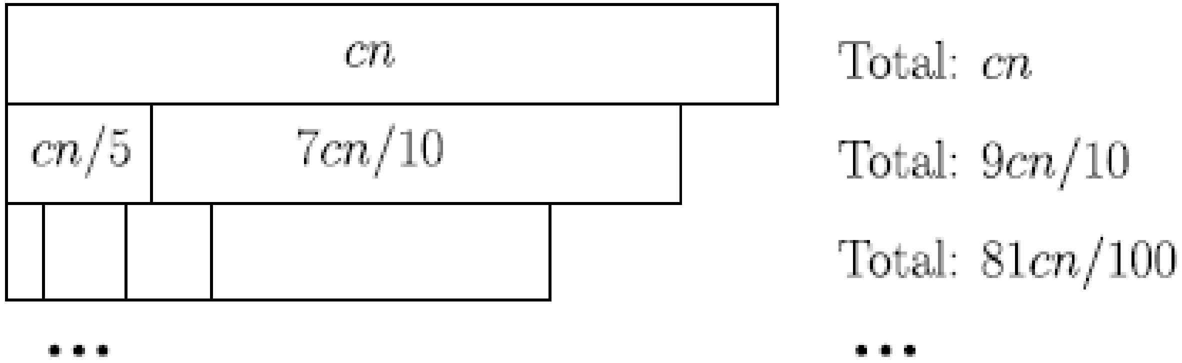
Total: $81cn/100$

$\cdots$

$$
\begin{aligned}
T(n) &\le cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right) \\
&\le cn + \frac{cn}{5} + \frac{c7n}{10} + T\left(\frac{n}{25}\right) + T\left(\frac{7n}{50}\right) + T\left(\frac{7n}{50}\right) + T\left(\frac{49n}{100}\right) \\
&\le cn + \frac{cn}{5} + \frac{c7n}{10} + \frac{cn}{25} + \frac{c7n}{50} + \frac{c7n}{50} + \frac{c49n}{100} + \cdots \\
&= cn + \frac{9cn}{10} + \frac{81cn}{100} + \cdots
\end{aligned}
$$

Running Time of DeterministicSelect

- $T(n) \le cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$



- Time is cn $\left(1 + \left(\frac{9}{10}\right) + \left(\frac{9}{10}\right)^2 + \ \dots\right) \le 10cn$
- Recurrence works because n/5 + 7n/10 < n

- For constants c and $a_1, a_2, \dots a_r$ with $a_1 + a_2 + \cdots a_r < 1$, the recurrence $T(n) \le T(a_1 n) + T(a_2 n) + \ \dots + T(a_r n) + cn$ solves to $T(n) = O(n)$
  - If instead $a_1 + a_2 + \ \dots + a_r = 1$, the recurrence solves to T(n) = O(n log n)
  - If we use median of 3 in DeterministicSelect instead of median of 5, what happens?