Lecture 1: Introduction and Median Finding

Staff

Professors:

Danny Sleator

Elaine Shi

TAs:

Joel Manning

Abby Li

Aditya Sundaram

Efe Cekirage

Jonathan Liu

Yoseph Mak

Nick Grill

Summit Wei

Please call me "Elaine" or "Runting"

润婷

- Prof. Shi
- Respected Madam
- Respected Sir

Grading and Course Policies

- All available here: https://www.cs.cmu.edu/~15451-s23/policies.html

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

Homework

- **Written HW**: Each HW has 3-4 problems

- **Programming Problems**: Typically, one problem is a programming problem – submit via Autolab (languages accepted are Java, C, C++, Ocaml, SML)

- **Oral HW:** For oral HWs you can collaborate, but write the programming problem yourself (unless otherwise noted).

Homework submission

- Submit on Gradescope/Autolab **by 11:59PM on the Wednesday** following release

- Grace day policy: 2 for written and 2 for programming (see course webpage)

- HW1 posted today.

Office hour

Two options:

- **In person:** Danny and TAs’ OH, sign up using OHQ,
- **In person or remote:** Elaine’s OH, sign up for a 15-min slot on youcanbookme

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?

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

$$a_1, a_2, \dots, a_n$$

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

Example: Median Finding

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

$$a_1, a_2, \dots, a_n$$

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?*

How can we find the median?

Naive idea 1: check if each element is median.

Runtime? $O(n^2)$

How can we find the median?

Naive idea 1: check if each element is median.

Runtime? $O(n^2)$

Naive idea 2: sort the array (MergeSort or QuickSort)

Runtime? $O(n \log n)$

Can we do better?

Naive idea 1: check if each element is median.

Runtime?

Naive idea 2: sort the array (MergeSort or QuickSort)

Runtime?

QuickSelect Algorithm to Find the k-th Smallest Number

- Assume $a_1, a_2, \dots, a_n$ 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

recurse on LESS

recurse on GREATER

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 $k \leq |\mathrm{LESS}|$, find the k-th smallest element in LESS
- If $k = |\mathrm{LESS}| + 1$, 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

Incorrect Proof:

$$n + \frac{n}{2} + \frac{n}{4} + \cdots = O(n)$$

Bounding the Running Time

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

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!

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!
  - Let $\mathrm{T(n)} = \max_{\mathrm{k}} \mathrm{T(n,k)}$

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!
  - Let $T(n) = \max_{k} T(n,k)$

- T(n) is a non-decreasing function of n
  - Can show by induction

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!
  - Let $T(n) = \max_k T(n,k)$

- T(n) is a non-decreasing function of n
  - Can show by induction

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

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

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

Bounding the Running Time

- Suppose we have an array of length n. Assume n is even for the moment

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

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}

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 $T(i)$ is non-decreasing with i, to upper bound T(n) we can assume we recurse on larger half

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 $T(i)$ is non-decreasing with i, to upper bound T(n) we can assume we recurse on larger half
  - $T(n) \le n - 1 + \frac{2}{n} \sum_{i=\frac{n}{2},\dots,n-1} T(i)$

\# comp with pivot

n=6

prob	LESS	GREATER	larger half
1/6	0	5	5
⋮	1	4	4
⋮	2	3	3
⋮	3	2	3
⋮	4	1	4
1/6	5	0	5

n-1

$\frac{n}{2}$

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 $T(i)$ is non-decreasing with i, to upper bound T(n) we can assume we recurse on larger half
  - $T(n) \leq n - 1 + \frac{2}{n} \sum_{i=\frac{n}{2},\ldots,n-1} T(i)$

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

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 $T(i)$ is non-decreasing with i, to upper bound T(n) we can assume we recurse on larger half
  - $T(n) \leq n - 1 + \frac{2}{n} \sum_{i=\frac{n}{2},\ldots,n-1} T(i)$

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

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

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 $T(i)$ is non-decreasing with i, to upper bound T(n) we can assume we recurse on larger half
  - $T(n) \leq n - 1 + \frac{2}{n} \sum_{i=\frac{n}{2},\ldots,n-1} T(i)$

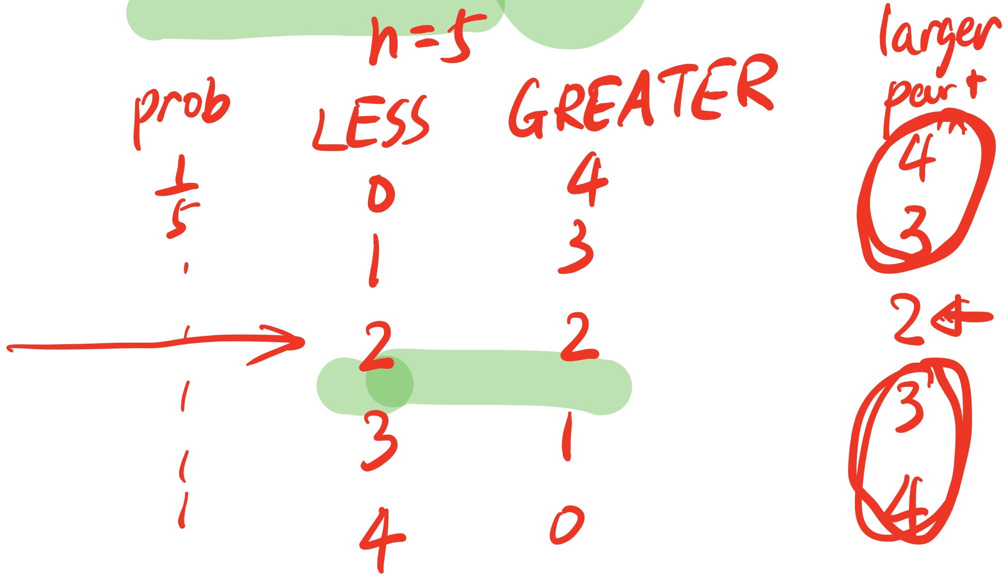
$\leq n - 1 + \frac{2}{n} \sum_{i=\frac{n}{2},\ldots,n-1} 4i$ by inductive hypothesis

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

$< 4n$ 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



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
  - $T(n) \leq n - 1 + \frac{1}{n} T\left(\frac{n-1}{2}\right) + \frac{2}{n} \sum_{i=\frac{n+1}{2},\ldots,n-1} T(i)$

#comp with pivot

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

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

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

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

What About **Deterministic** Algorithms?

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

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 $\lfloor \frac{n}{2} \rfloor$

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: Blum et. al.

  1. Group the array into n/5 groups of size 5 and find the median of each group → $\frac{n}{5}$ medians

  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

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

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

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

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| ≥ 3n/10-1 and |GREATER| ≥ 3n/10-1

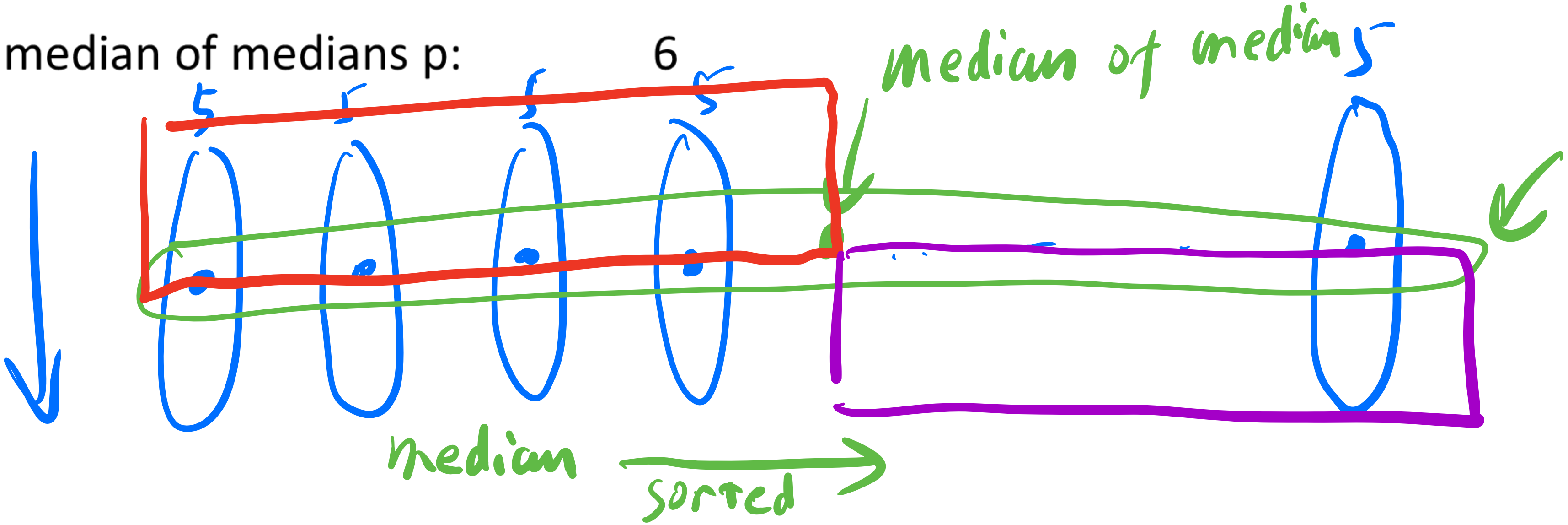
The pivot is reasonable

- **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



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| $\geq 3n/10\text{-}1$ and |GREATER| $\geq 3n/10\text{-}1$, Step 4 takes at most T(7n/10) time

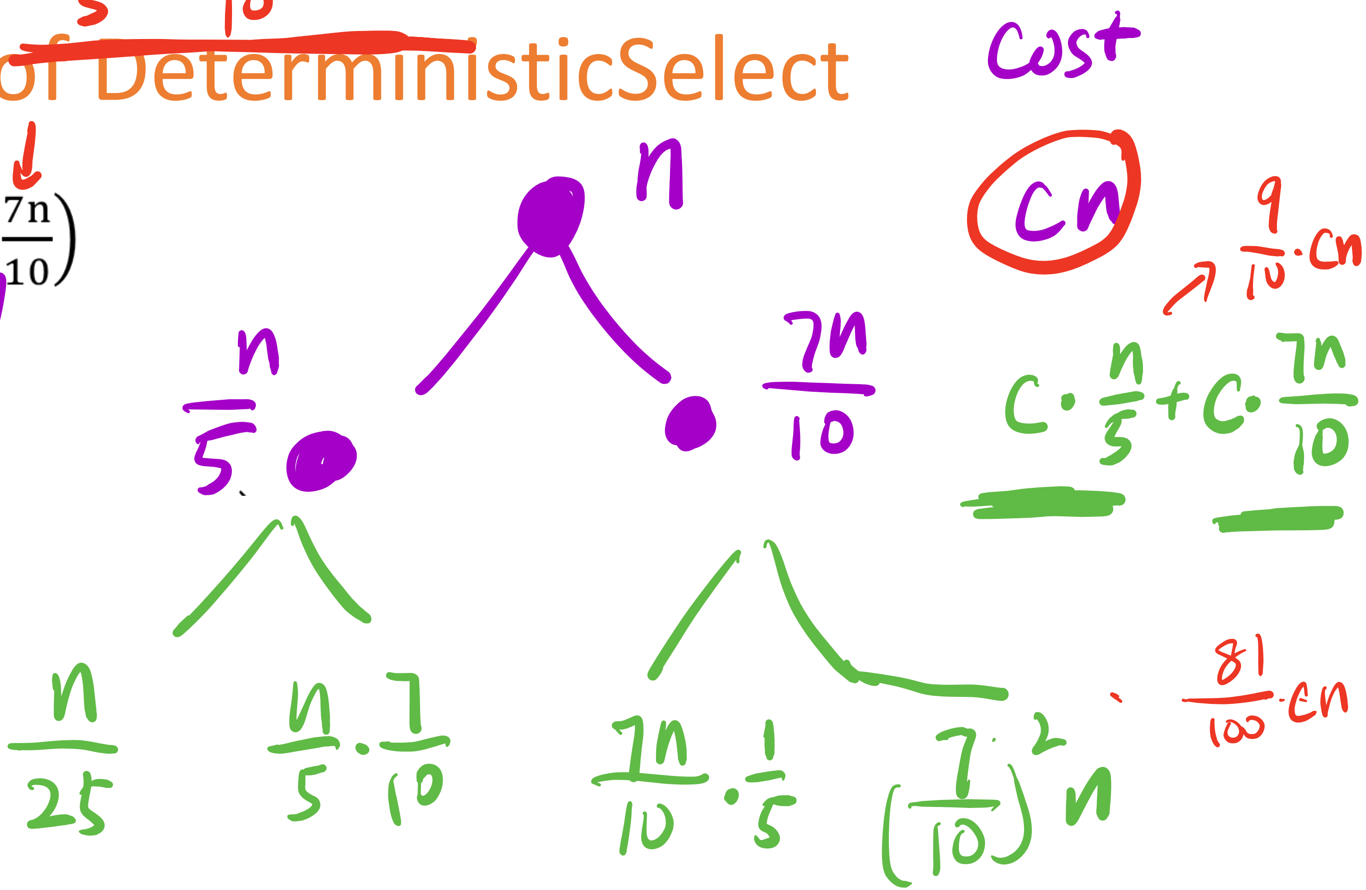
- So $T(n) \leq cn + T\left(\frac{n}{5}\right) + T\left(\frac{7n}{10}\right)$, for a constant c > 0

IMPORTANT: $\frac{1}{5} + \frac{7}{10} < 1$

Running Time of DeterministicSelect

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

$$cn + \frac{9}{10}cn + \frac{81}{100}\cdot cn + \ldots\ldots = O(n)$$



Running Time of DeterministicSelect

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

Running Time of DeterministicSelect

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

Thank you!

LESS / Greater 's at least

$\frac{n}{6} \cdot 2$ in size

$\frac{1}{3} n$

$T(n) = cn + T(\frac{1}{3} n) + T(\frac{2}{3} n)$

n