CSE331 Introduction to Algorithm
Lecture 8: The Selection Problem

Antoine Vigneron
antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

July 11, 2017

1. Introduction
2. Selection in expected linear time
3. Analysis of Randomized Select
4. Selection in worst-case linear time
5. Concluding remarks

Introduction

- In this lecture, we present two algorithms for the selection problem.
- The first algorithm is randomized.
  - As a byproduct, we also present an analysis of randomized quicksort.
- The second algorithm is deterministic.
- References:
  - Section 9 of the textbook Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein. (Available online from the UNIST library website.)
  - Analysis taken from the textbook Algorithm Design by Kleinberg and Tardos.

Problem Statement

Problem

Given an array $A[1 \ldots n]$ of n distinct numbers and an integer i, the selection problem is to find the ith smallest number in A.

Example

Given $A = [8, 4, 5, 6, 7, 9, 7, 1]$ and $i = 3$, then the answer is 5 because A in sorted order is $B = [1, 4, 5, 6, 7, 7, 8, 9]$ and $B[3] = 5$.

Special cases

- $i = 1$ gives the *minimum* in A.
- $i = n$ gives the *maximum* in A.
- The middle element is the *median*. More precisely:
  - $i = \lfloor (n+1)/2 \rfloor$ is the *lower median*.
  - $i = \lceil (n+1)/2 \rceil$ is the *upper median*.

Naive Algorithm

Pseudocode

1: **procedure** NAIVE SELECT($A[1 \ldots n]$, i)
2: MERGE SORT(A)
3: **return** $A[i]$

- This algorithm runs in $\Theta(n \log n)$ time.
- But it is easy to do better for $i = 1$ or $i = n$. (See next slide.)

Computing the Minimum or the Maximum

Pseudocode

```
1: procedure MINIMUM(A[1 ... n])
2:     min ← A[1]
3:     for i ← 2, n do
4:         if A[i] < min then
5:             min ← A[i]
6:     return min
```

- This runs in time $\Theta(n)$, so it is not a good idea to run the algorithm from the previous slide in this case.
- This lecture: We give $\Theta(n)$ time algorithm for *every* i, not just the minimum ($i = 1$) or the maximum ($i = n$).
- In particular, it shows that the median can be computed in linear time.

Selection in Expected Linear Time

- First run the randomized partition procedure of QUICKSORT, which splits A at a random element $A[q]$ in linear time.

- If $i = q$ return $A[q]$.
- If $i < q$ recurse on $A' = A[1 \ldots q-1]$ with $i' = i$.
- IF $i > q$ recurse on $A[q+1 \ldots n]$ with $i' = i - q$,

Selection in Expected Linear Time

Pseudocode

```
1:  procedure RANDOMIZED SELECT(A, p, r, i)
2:      if p = r then
3:          return A[p]                                  ▷ array of size 1
4:      r' ← RANDOM(p, r)
5:      Exchange A[r] with A[r']
6:      q ← PARTITION(A[p ... r])                        ▷ random partition
7:      k ← q − p + 1
8:      if i = k then
9:          return A[q]                                  ▷ the result is the pivot
10:     if i < k then
11:         return RANDOMIZED SELECT(A, p, q − 1, i)
12:     return RANDOMIZED SELECT(A, q + 1, r, i − k)
```

Analysis of Randomized Select

Intuition:

- Same as Quicksort, at most iterations, the array shrinks by a constant factor.
- So the running time is of the form $\sum_j T(\rho^j n)$ where $0 < \rho < 1$ and $T(n) = \Theta(n)$ is the running time of Partition.
- Therefore it is $(\sum_j \rho^j)\Theta(n) = \frac{1}{1-\rho}\Theta(n)$, which is this is $\Theta(n)$.
- We now give a rigorous proof.

Analysis of Randomized Select

- On average, how many times do you need to roll a dice before you get a 6?
- Answer: 6 times.

Theorem (Waiting-time bound)

If we repeatedly perform independent trials of an experiment, each of which succeeds with probability $\mu > 0$, then the expected number of trials we need to perform until the first success is $\frac{1}{\mu}$.

Proof:

- Let X denote the number of trials until the first success.
- $\Pr[X = j] = (1 - \mu)^{j-1}\mu$ for every $j > 0$,

Analysis of Randomized Select

- So

$$E[X] = \sum_{j=1}^{\infty} j \cdot \Pr[X = j] = \sum_{j=1}^{\infty} j(1-\mu)^{j-1}\mu = \mu \sum_{j=1}^{\infty} j(1-\mu)^{j-1}$$

- To complete the proof of the waiting-time bound, we need to prove that

$$\sum_{j=1}^{\infty} j(1-\mu)^{j-1} = \frac{1}{\mu^2}$$

- It is true because for any $x \in (0,1)$:

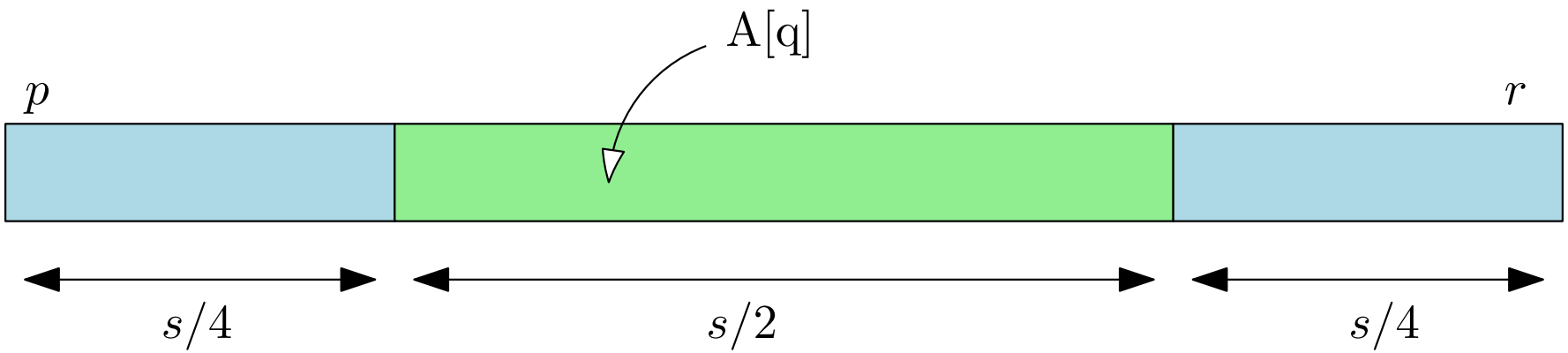
$$\sum_{j=1}^{\infty} j x^{j-1} = \left(\sum_{j=0}^{\infty} x^j\right)' = \left(\frac{1}{1-x}\right)' = \frac{1}{(1-x)^2}$$

Analysis of Randomized Select

- We say that the algorithm is in *phase* j if the size $s = r - p + 1$ of the subarray under consideration satisfies

$$n\left(\frac{3}{4}\right)^{j+1} < s \leqslant n\left(\frac{3}{4}\right)^{j}.$$

- We say that the pivot $A[q]$ is *central* if at least a quarter of the elements are larger and at least a quarter are smaller.



Analysis of Randomized Select

- A pivot is central with probability $1/2$.
- A central pivot allows to discard at least one quarter of the elements, so it takes us from phase j to phase $\geqslant j+1$.
- Thus, by the waiting-time bound, the algorithm stays in phase j for at most 2 iterations on average.
- Let X denote the total running time, and X_j the running time in phase j.
- The running time of Partition is $\leqslant cn$ for some constant $c > 0$.

Analysis of Randomized Select

$$
\begin{aligned}
E[X] &= E\left[\sum_j X_j\right] \\
&= \sum_j E[X_j] && \text{by linearity of expectation} \\
&\leqslant \sum_j 2cn\left(\frac{3}{4}\right)^j && \text{on average} \leqslant 2 \text{ iterations, } cn\left(\frac{3}{4}\right)^j \text{ each} \\
&= 2cn\sum_j \left(\frac{3}{4}\right)^j \\
&\leqslant 8cn \\
&= \Theta(n)
\end{aligned}
$$

Analysis of Randomized Select

- So Randomized Select runs in expected linear time.
- This is faster than Randomized Quicksort, which runs in expected $\Theta(n \log n)$ time.
- Reason: Randomized Select just follows one path from the root to a leaf of the recursion tree of Randomized Quicksort.
- Remainder of this lecture: We give a *deterministic* linear-time algorithm.
- It is more complicated, and has larger constants in its running time, so the deterministic algorithm is probably not a good choice in practice.
- But it is useful in theory.

Selection in Worst-Case Linear Time

- The randomized selection algorithm picks a pivot at random.
- It gives a linear expected running time because, with probability $1/2$, the pivot is *central*, i.e. it splits the array into two parts, each part being of size at least $n/4$.
- Here we give a *deterministic* algorithm, that does not need randomization to find a central pivot.

Finding the Pivot

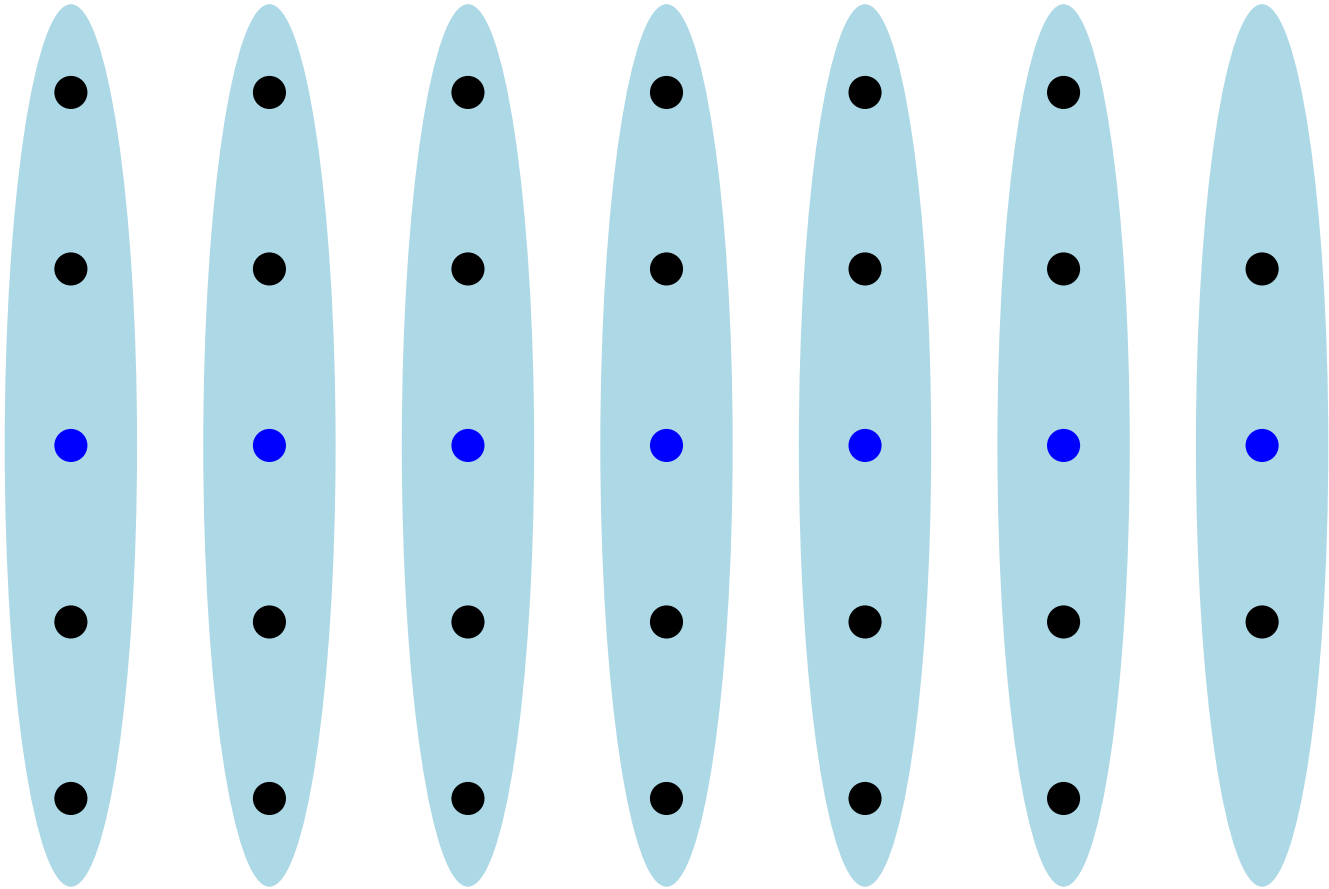


Figure: Group the elements of the array by 5, and compute the median of each group (blue).

Finding the Pivot

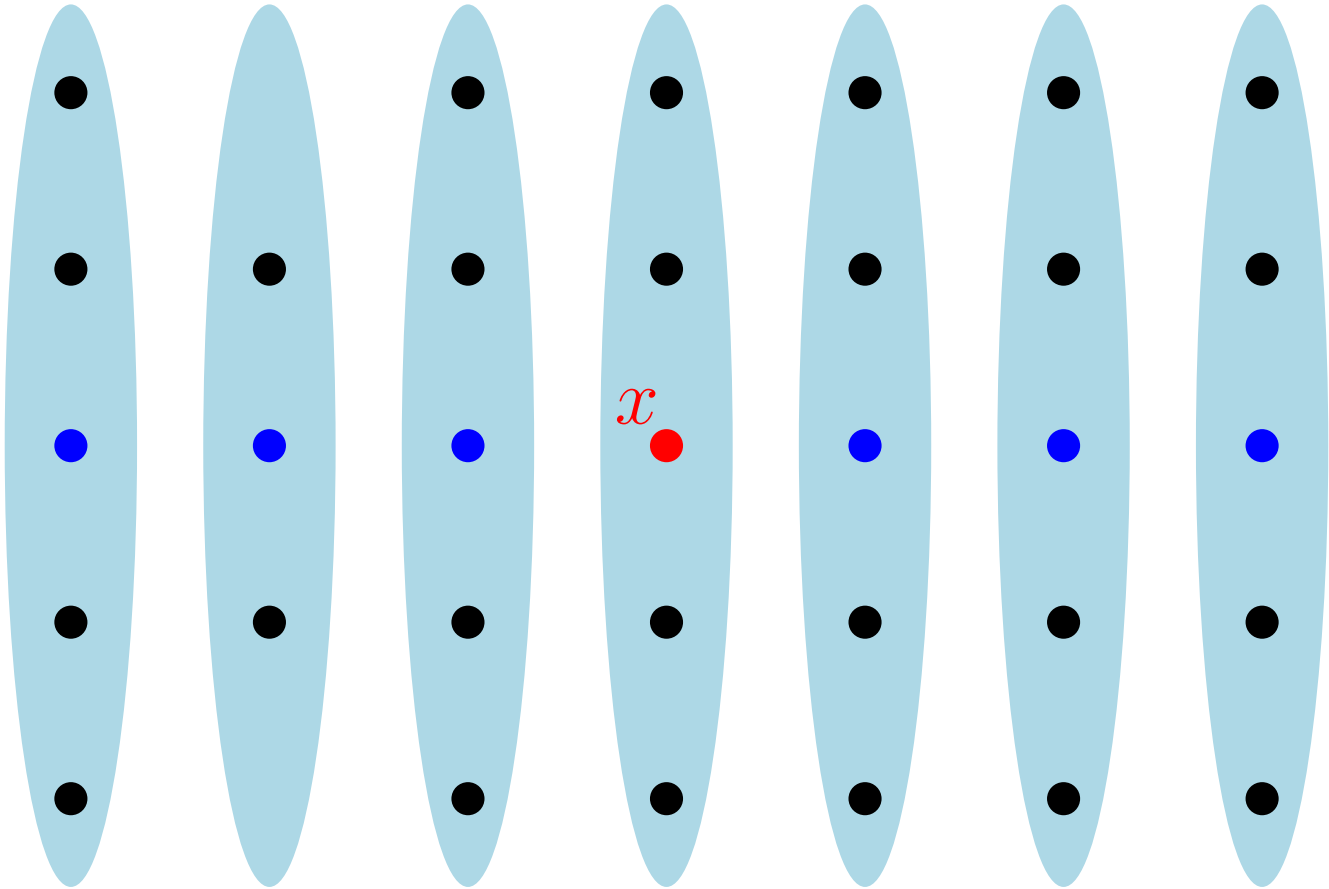


Figure: Compute the median x of the $\lceil n/5 \rceil$ medians (red). Imagine they are sorted from left to right

Finding the Pivot

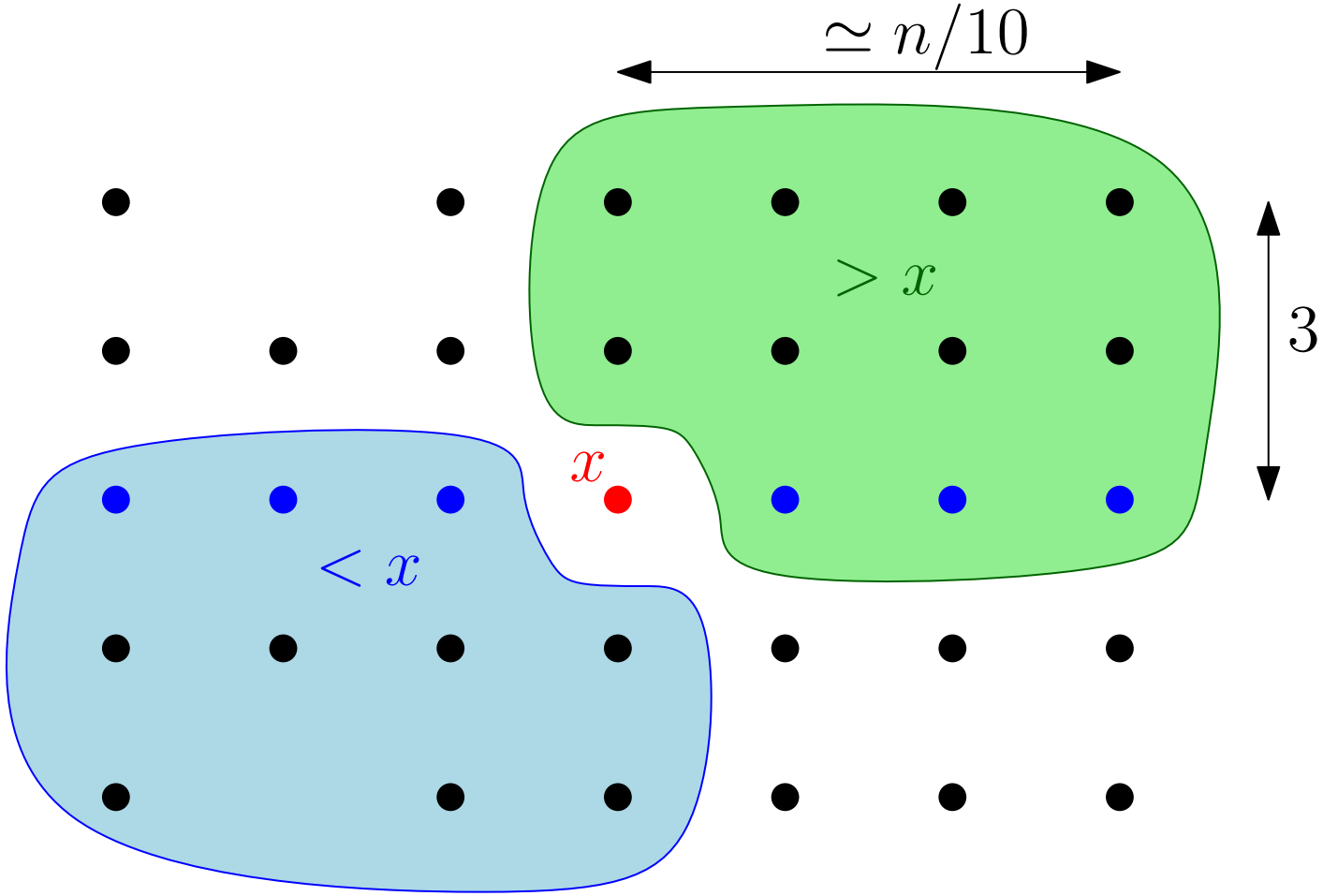


Figure: The keys in the green region are larger than x. There are about $3n/10$ of them. Similarly, there are about $3n/10$ keys in the blue regions, and they are smaller than x.

Selection in Worst-Case Linear Time

Pseudocode

SELECT(A, p, r, i)

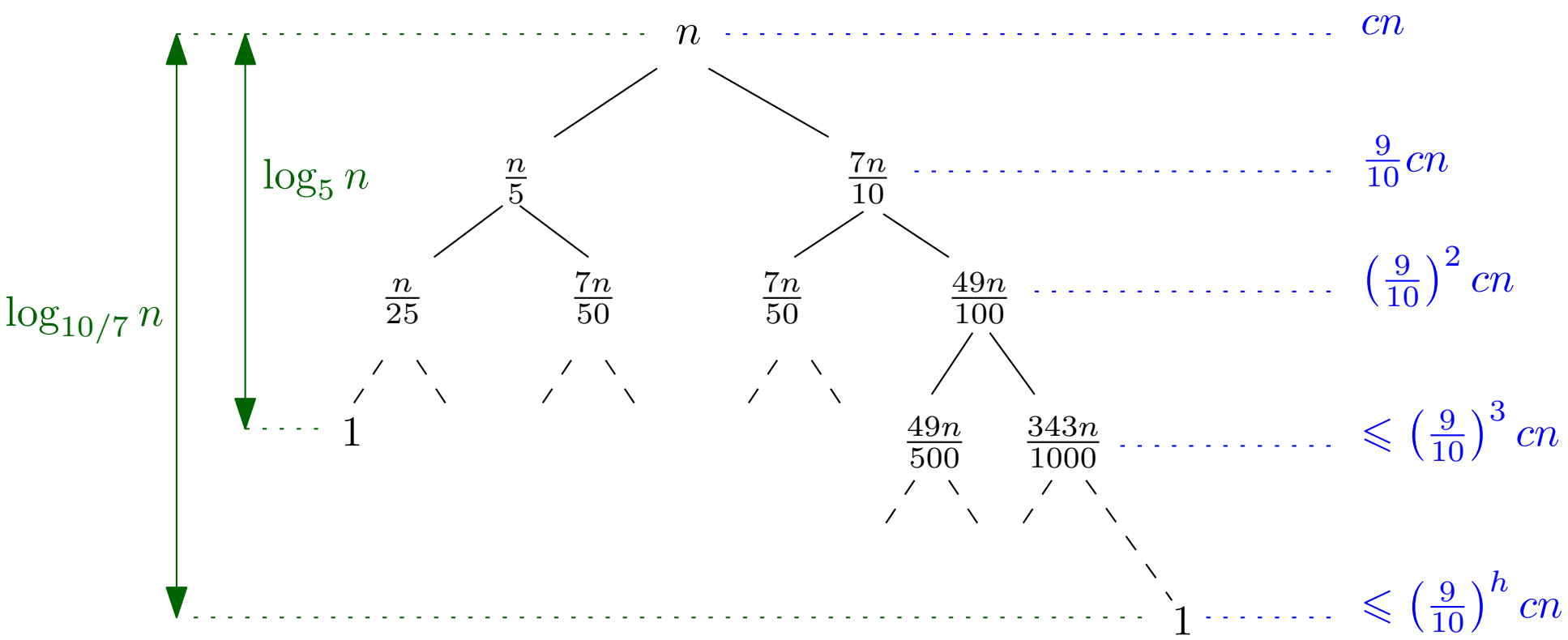
1. Divide the keys into $\lfloor n/5 \rfloor$ groups of 5, and at most one group of size less than 5.
2. Find the median of each group using INSERTION SORT.
3. Find the median x of these $\lceil n/5 \rceil$ medians by calling SELECT recursively.
4. Partition the array using x as the pivot.
   Assume that now $x = A[q]$, and let $k = q - p + 1$.
5. If $i = k$, then return x.
6. If $i < k$, then return SELECT$(A, p, q - 1, i)$.
7. If $i > k$, then return SELECT$(A, q + 1, r, i - k)$.

- The only difference with RANDOMIZED SELECT is the choice of the pivot in Lines 1–3.

Analysis

- Rough analysis: Let $T(n)$ be the running time of SELECT.
  - Line 1: $\Theta(n)$.
  - Line 2: Insertion sort on an array of size 5 takes $O(1)$ time, so Line 2 takes time $\Theta(n)$
  - Line 3: $T(n/5)$
  - Line 4: $\Theta(n)$
  - Line 5: $\Theta(1)$
  - From Slide 19, after partitioning an array of size n, the resulting subarrays have size $\leqslant 7n/10$
  - So Line 6 and 7 are at most $T(7n/10)$
- We obtain $T(n) \leqslant T(n/5) + T(7n/10) + \Theta(n)$.
- The master theorem does not apply.
- Next slide: We guess a bound on $T(n)$ using the recursion tree method.

Analysis: Recursion Tree



Total: $\leqslant \frac{1}{1-\frac{9}{10}} cn$

$= 10cn$

Analysis: Recursion Tree

- The sum of the size of the subproblems is decreases by a factor (at least) 9/10 at each level.
- So an upper bound on the running time should be

$$\begin{aligned} T(n) &\leqslant \sum_{j=0}^{\infty} \left(\frac{9}{10}\right)^j cn \\ &= cn \sum_{j=0}^{\infty} \left(\frac{9}{10}\right)^j \\ &= cn \frac{1}{1-\frac{9}{10}} \\ &= 10cn \end{aligned}$$

- So our conjecture is that $T(n) = \Theta(n)$.

Analysis: Substitution Method

- We now give a rigorous proof that $T(n) = \Theta(n)$ using the substitution method.
- The proof can be found in the textbook. (Page 221.)

Lemma

The number of elements $\leqslant x$ is at least $\frac{3n}{10} - 6$. Similarly, the number of elements $> x$ is at least $\frac{3n}{10} - 6$.

(Proof done in class.)

Remark: These are the numbers from the textbook. The alternate proof I made during the lecture also works, and gives $3n/10 - 2$ and $3n/10 - 3$, which is slightly better. But in the end the time bound remains $\Theta(n)$.

Analysis: Substitution Method

- Our analysis above shows that for all $n \geqslant 1$

$$T(n) \leqslant T(\lceil n/5 \rceil) + \max_{1 \leqslant i \leqslant 7n/10+6} \{T(i)\} + O(n).$$

- So there are positive constants a, b such that:

$$T(n) \leqslant \begin{cases} T(\lceil n/5 \rceil) + \max_{1 \leqslant i \leqslant 7n/10+6} \{T(i)\} + an & \text{if } n \geqslant 140 \\ b & \text{if } n < 140 \end{cases} \quad (1)$$

- We make the inductive hypothesis that $T(m) \leqslant cm$ for all $m < n$.
- **Basis step**: For $n = 140$, and thus $m < 140$, it suffices to choose $c \geqslant b$.

Analysis: Substitution Method

Inductive step:

- We assume that $n \geqslant 140$ and that, for all $m < n$, we have $T(m) \leqslant cm$.
- We need to prove that $T(n) \leqslant cn$.
- By Equation (1), we obtain

$$T(n) \leqslant T(\lceil n/5 \rceil) + c\left(\frac{7n}{10} + 6\right) + an.$$

- Rest of the proof done it class. It suffices to choose $c \geqslant 20a$.

Concluding Remarks

- The deterministic algorithm for selection is interesting from a theoretical standpoint, but in practice it should be slower than the randomized version.
- This is similar to sorting algorithms: QUICKSORT is a simple randomized algorithm that outperforms deterministic algorithms with a high probability.
- The alternate analysis of randomized QUICKSORT is posted on blackboard as *Notes on Lecture 8*.