CSE331 Introduction to Algorithm
Lecture 14: Deterministic Algorithm for The Selection Problem

Antoine Vigneron
antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

July 23, 2021

1. Introduction

2. Selection in worst-case linear time

3. Concluding remarks

Course Organization

- In the previous lecture, I gave a randomized algorithm for the selection problem that runs in expected linear time.
- Today, I present a deterministic algorithm that also runs in linear time.
- Reference: Section 9 of the textbook Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein. (Available online from the UNIST library website.)

Problem Statement

Problem

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

Example

Given $A = [8, 4, 5, 6, 12, 9, 7, 1]$ and $i = 3$, then the answer is 5 because A in sorted order is $B = [1, 4, 5, 6, 7, 8, 9, 12]$ 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:
  - $i = \lfloor (n + 1)/2 \rfloor$ gives the *lower median*.
  - $i = \lceil (n + 1)/2 \rceil$ gives the *upper median*.

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.
- As we did in the lectures on QUICKSORT and randomized selection, we assume that no two input numbers are equal.

Finding the Pivot

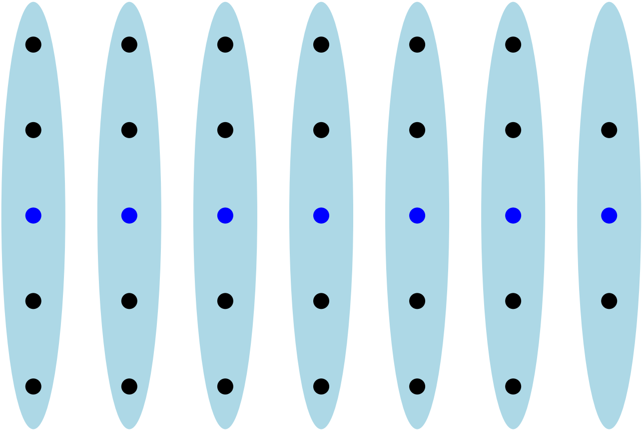


Figure: Group the elements of the array by 5, and compute the median of each group of 5 by brute force (blue).

Finding the Pivot

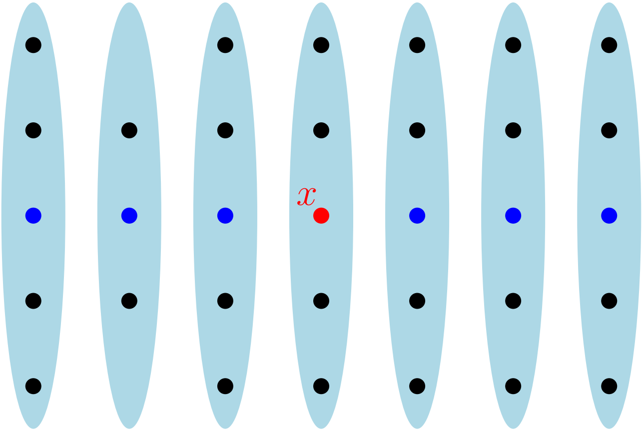


Figure: Recursively compute the median x of the $\lceil n/5 \rceil$ medians (red). Imagine they are sorted from left to right

Finding the Pivot

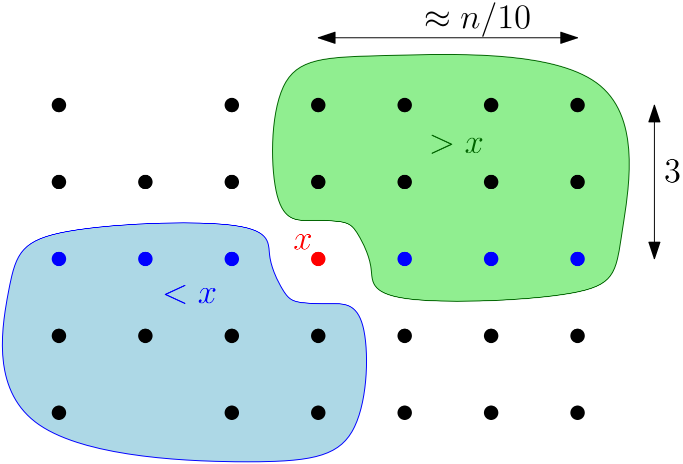


Figure: The keys in the green region are larger than x. There are about $3n/10$ of them. Similarly, there are about $3n/10$ keys in the blue regions, and they are smaller than x.

Selection in Worst-Case Linear Time

Pseudocode

SELECT(A, p, r, i)

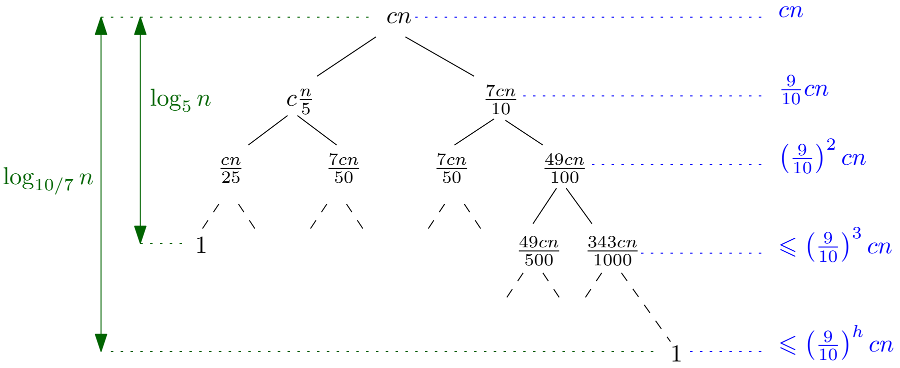
1. Divide the keys into $\lceil n/5 \rceil$ groups of size 5 each. (More precisely, one group may have size < 5.)
2. Find the median of each group using INSERTION SORT.
3. Find the median x of these $\lceil n/5 \rceil$ medians by calling SELECT recursively.
4. Partition the array using x as the pivot. 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 at 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 8, 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 decreases by a factor (at least) 10/9 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:

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

- 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. In other words, the constant factor hidden in the notation $\Theta(n)$ is larger for the deterministic version.
- This is similar to sorting algorithms: $\textsc{Quicksort}$ is a simple randomized algorithm that outperforms deterministic algorithms with a high probability.