CSE515 Advanced Algorithms
Lecture 22
Randomized Selection

Antoine Vigneron
antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

July 27, 2021

1. Introduction
2. Selection in expected linear time
3. Analysis: intuition
4. Modified algorithm
5. Analysis of SELECT
6. Conclusion

Course Organization

- Assignment 2 due next Monday (April 8).
- In this lecture, we present a randomized algorithm for the selection problem, and show how to analyze it.
- References:
  - Section 9 of Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein. (Available online from the UNIST library website.)
  - Section 13.5 of Algorithm Design by Kleinberg and Tardos.

Problem Statement

Problem

Given a set S of n distinct numbers and an integer k, the selection problem is to find the kth smallest number in S.

Example

Given $S = \{8, 4, 5, 6, 12, 9, 7, 1\}$ and $k = 3$, then the answer is 5 because S sorted in an array is $A = [1, 4, 5, 6, 7, 8, 9, 12]$ and $A[3] = 5$.

Special cases

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

Naive Algorithm

Pseudocode

1: **procedure** NaiveSelect(S, k)
2: $A[1 \ldots n] \leftarrow$MergeSort(S)
3: **return** $A[k]$

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

Computing the Minimum or the Maximum

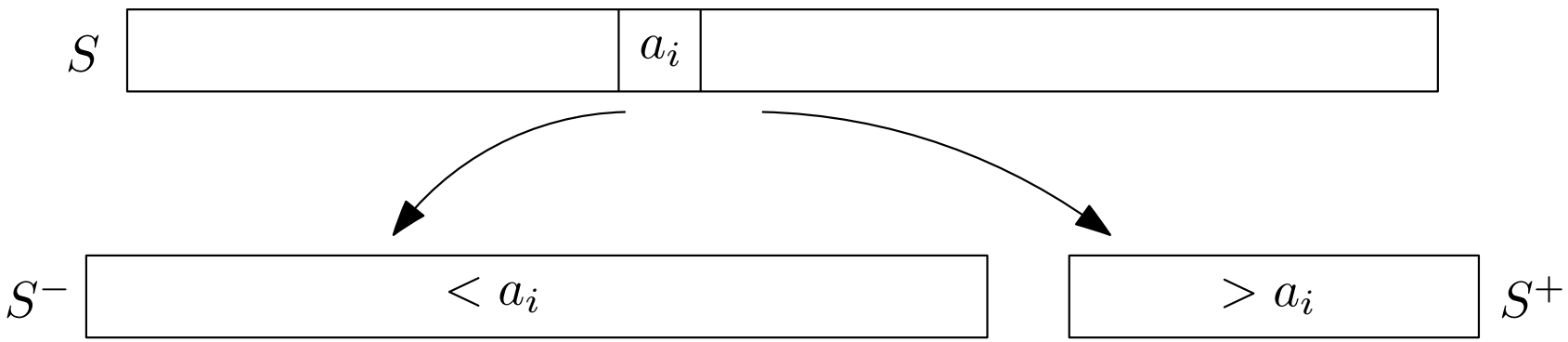
Pseudocode

1: **procedure** MINIMUM($S = \{a_1, \ldots, a_n\}$)
2: result $\leftarrow a_1$
3: **for** $i \leftarrow 2, n$ **do**
4: result $\leftarrow \min(\text{result}, a_i)$
5: **return** result

- 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 an expected $\Theta(n)$-time randomized algorithm for *every* k, not just the minimum ($k = 1$) or the maximum ($k = n$).
- In particular, it shows that the median can be computed in expected linear time.

Selection in Expected Linear Time

- Assume that the elements of S are distinct, i.e. $a_i \neq a_j$ for all $i \neq j$.
- Pick $a_i \in S$ at random. (a_i is called the *pivot*.)
- Split S into S^- and S^+, where S^- and S^+ contain the elements less than a_i and more than a_i, respectively.



- If $k \leqslant |S^-|$ then return the kth element of S^-.
- If $k = |S^-| + 1$ then return a_i
- If $k > |S^-| + 1$ then return the $(k - |S^-| - 1)$th element of S^+.

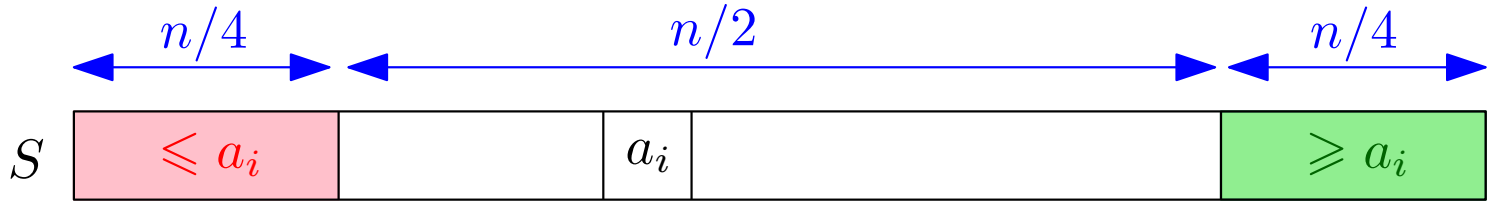
Selection in Expected Linear Time

Pseudocode

1: **procedure** $\textsc{Select}(S, k)$
2: pick $a_i \in S$ at random
3: **for** each $a_j \in S$ **do**
4: **if** $a_j < a_i$ **then** insert a_j into S^-
5: **if** $a_j > a_i$ **then** insert a_j into S^+
6: $\ell \leftarrow |S^-|$
7: **if** $k \leqslant \ell$ **then**
8: **return** $\textsc{Select}(S^-, k)$
9: **if** $k = \ell + 1$ **then**
10: **return** a_i
11: **return** $\textsc{Select}(S^+, k - \ell - 1)$

Analysis of randomized SELECT: Intuition

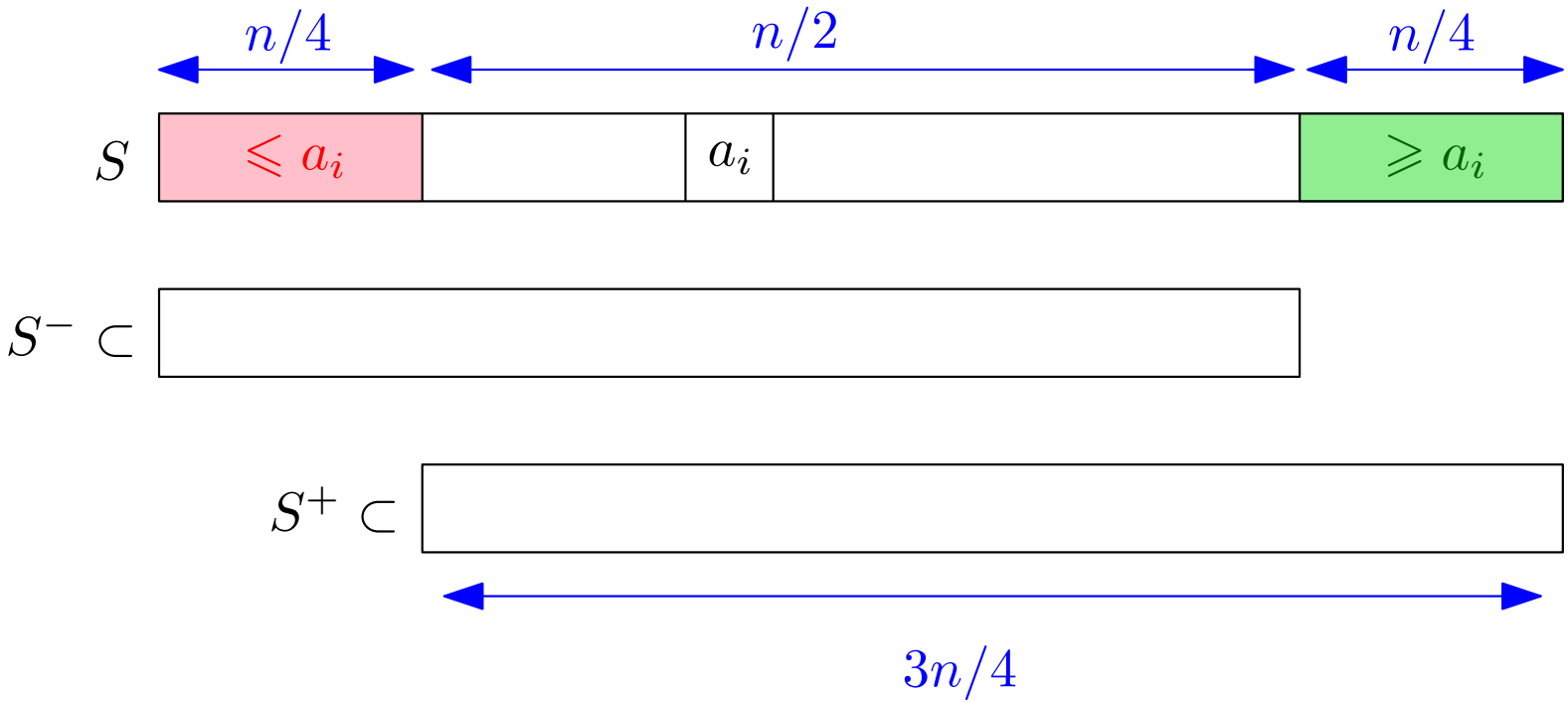
- We say that the pivot a_i is *central* if at least one quarter of the elements are $\leqslant a_i$ and at least one quarter are $\geqslant a_i$.



- What is the probability that the pivot is central?
  - Answer: $1/2$.
  - More precisely, it can be slightly more, i.e. between $1/2$ and $1/2 + 2/n$, but it will not affect our proofs, so we will ignore this fact.

Analysis of randomized SELECT: Intuition

- What can we say about the sizes of S^- and S^+ when the pivot is central?
  - Answer: $|S^-| \leqslant 3n/4$ and $|S^+| \leqslant 3n/4$.



Analysis of randomized SELECT: Intuition

- Each call to SELECT takes linear time, say time $c|S|$ for some constant c.
- Suppose all pivots are central. Then $|S|$ shrinks by a factor at least $4/3$ at each recursive call, hence

$$T(n) \leqslant cn + \frac{3}{4}cn + \left(\frac{3}{4}\right)^2 cn + \ldots$$
$$= c\frac{1}{1-3/4}n = 4cn$$

- As only half of the pivots are central, we should get $T(n) \leqslant 8cn$ so $T(n) = O(n)$.
- We now want to make this analysis rigorous.

Modified Algorithm

Pseudocode

1: **procedure** ModifiedSelect(S, k)
2: **repeat**
3: pick $a_i \in S$ at random
4: **for** each $a_j \in S$ **do**
5: **if** $a_j < a_i$ **then** insert a_j into S^-
6: **if** $a_j > a_i$ **then** insert a_j into S^+
7: $\ell \leftarrow |S^-|$
8: **until** $n/4 \leqslant \ell \leqslant 3n/4$
9: **if** $k \leqslant \ell$ **then**
10: **return** ModifiedSelect(S^-, k)
11: **if** $k = \ell + 1$ **then**
12: **return** a_i
13: **return** ModifiedSelect($S^+, k - \ell - 1$)

Analysis

- What does this modification mean?
- We now make sure that the pivot is central before each recursive call.
- So the size of the subproblem shrinks by a factor 4/3 at each recursive call.
- It will make it easier to analyze ModifiedSelect.

Analysis

- At each call to ModifiedSelect, what is the expected number of times we iterate the repeat ... until loop?
- As the pivot is central with probability $1/2$, by the waiting time bounds, the repeat ... until loop is iterated on average *twice*.

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 $\frac{1}{\mu}$.

- So ignoring recursive calls, ModifiedSelect runs in expected linear time, say $c'|S|$ for some constant c'.

Analysis

- So the running time of ModifiedSelect can be easily bounded:

$$\begin{aligned} M(n) &\leqslant c'n + \frac{3}{4}c'n + \left(\frac{3}{4}\right)^2 c'n + \ldots \\ &= c'\frac{1}{1-3/4}n \\ &= 4c'n \\ &= O(n) \end{aligned}$$

Analysis of Randomized SELECT

- We say that the algorithm is in *phase* j if the size $|S|$ of the subset under consideration satisfies

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

  where n is the input size.
- 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 Y denote the total running time, and Y_j the running time in phase j.

Analysis of Randomized SELECT

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

Concluding Remarks

- So randomized SELECT runs in expected linear time.
- This is faster than sorting with any comparison-based algorithm (see CSE331).
- So if we only want to compute the median of an array, it is better not to sort it.
- There is also a linear-time *deterministic* algorithm.
- It is more complicated, and slower in practice. (See CSE331.)
- Remark: SELECT runs in expected linear time on *worst-case* input.
- So it still runs in linear time if the input is very skewed.
- Randomization is only done internally by the algorithm.
- This is different from some algorithms whose expected running time is good if the input is random. See for instance BUCKET SORT.