CSE331 Introduction to Algorithm
Lecture 13: The Selection Problem

Antoine Vigneron
antoine@unist.ac.kr

Ulsan National Institute of Science and Technology

July 23, 2021

Course Organization

- In this lecture, I present a randomized algorithm for the selection problem.
- As a byproduct, we also obtain an analysis of randomized quicksort.
- In the next lecture, I will present a deterministic algorithm.
- 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 $A[1 \dots n]$ 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 $i = 3$, 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* of A.
- $i = n$ gives the *maximum* of 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*.

Naive Algorithm

Pseudocode

```
procedure NaiveSelect(A[1...n], i)
    mergesort(A)
    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

```
procedure MINIMUM(A[1 ... n])
    result ← A[1]
    for i ← 2, n do
        result ← min(result, A[i])
    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 a $\Theta(n)$ 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 $A' = A[1 \dots q-1]$ with $i' = i$.
- If $i > q$ recurse on $A[q+1 \dots n]$ with $i' = i - q$,

Selection in Expected Linear Time

Pseudocode

```
procedure RandomizedSelect(A, p, r, i)
    if p = r then
        return A[p]                          ▷ array of size 1
    r′ ← Random(p, r)                        ▷ random integer in [p, r]
    Exchange A[r] with A[r′]
    q ← Partition(A[p . . . r])              ▷ random partition
    k ← q − p + 1
    if i = k then
        return A[q]                          ▷ the result is the pivot
    if i < k then
        return RandomizedSelect(A, p, q − 1, i)
    return RandomizedSelect(A, q + 1, r, i − k)
```

Analysis of RANDOMIZED SELECT: Intuition

- Same as QUICKSORT, at most iterations, the array shrinks by a factor at least $\rho = 10/9$.
- So the running time is of the form $\sum_j T(n/\rho^j)$ where $T(n) = \Theta(n)$ is the running time of PARTITION.
- Therefore it is

$$\left(\sum_j \frac{1}{\rho^j}\right)\Theta(n) = \frac{1}{1-\frac{1}{\rho}}\Theta(n) = 10 \cdot \Theta(n),$$

  which 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 until 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 $x \in (0,1)$:

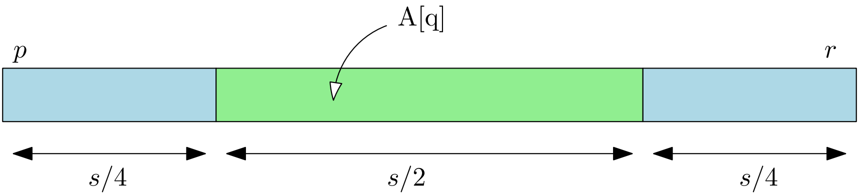
$$\sum_{j=1}^{\infty} jx^{j-1} = \left(\sum_{j=0}^{\infty} x^j\right)' = \left(\frac{1}{1-x}\right)' = \frac{1}{(1-x)^2}$$

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 Y denote the total running time, and Y_j the running time in phase j.
- The running time of Partition is $\leqslant cn$ for some constant $c > 0$.

Analysis of RANDOMIZED SELECT

$$
\begin{aligned}
E[Y] &= E\left[\sum_j Y_j\right] \\
&= \sum_j E[Y_j] && \text{by linearity of expectation} \\
&\leqslant \sum_j 2cn \left(\frac{3}{4}\right)^j && \text{on average} \leqslant 2 \text{ iterations,} \leqslant cn\left(\frac{3}{4}\right)^j \text{ each} \\
&= 2cn \sum_j \left(\frac{3}{4}\right)^j \\
&\leqslant 2cn \times \frac{1}{1-\frac{3}{4}} \\
&= 8cn \\
&= \Theta(n)
\end{aligned}
$$

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.
- Next lecture: We will give a *deterministic* linear-time algorithm.