Design and Analysis of Algorithms

Mohammad GANJTABESH

mgtabesh@ut.ac.ir

School of Mathematics, Statistics and Computer Science,
University of Tehran,
Tehran, Iran.

Selecting the k-th smallest number

Suppose that A is an array of length n, containing nonnegative integers. The following problems may be asked:

Selecting the k-th smallest number

Suppose that A is an array of length n, containing nonnegative integers. The following problems may be asked:

- **Finding the Minimum or Maximum:** Require $O(n)$ time complexity to be performed.

Selecting the k-th smallest number

Suppose that A is an array of length n, containing nonnegative integers. The following problems may be asked:

- **Finding the Minimum or Maximum:** Require $O(n)$ time complexity to be performed.
- **Finding the Minimum and Maximum at the same time:** This can be also performed in $O(n)$ time complexity (how?).

Selecting the k-th smallest number

Suppose that A is an array of length n, containing nonnegative integers. The following problems may be asked:

- **Finding the Minimum or Maximum:** Require $O(n)$ time complexity to be performed.
- **Finding the Minimum and Maximum at the same time:** This can be also performed in $O(n)$ time complexity (how?).
- **Finding the median:**
  - This can be done by extracting minimums for $n/2$ times and the last one is the median. So it requires $O(n^2)$ time complexity (not good).
  - Another way is by sorting the array and then extracting the middle element of the sorted array. This requires $O(n.\log(n))$ (not bad).
  - Can we solve this problem in better way?

Selecting the k-th smallest number

Suppose that A is an array of length n, containing nonnegative integers. The following problems may be asked:

- **Finding the Minimum or Maximum:** Require $O(n)$ time complexity to be performed.
- **Finding the Minimum and Maximum at the same time:** This can be also performed in $O(n)$ time complexity (how?).
- **Finding the median:**
  - This can be done by extracting minimums for $n/2$ times and the last one is the median. So it requires $O(n^2)$ time complexity (not good).
  - Another way is by sorting the array and then extracting the middle element of the sorted array. This requires $O(n.\log(n))$ (not bad).
  - Can we solve this problem in better way?
- **Finding the k-th smallest element:** This is the generalization problem of finding the median (in median we set $k = n/2$). Now we try to design an algorithm for $\mathit{Select}(k,\ n)$.

Selecting the k-th smallest number

Suppose that $A[s..e]$ is an array of length n. We can use the partition algorithm to solve this problem.

Selecting the k-th smallest number

Suppose that $A[s..e]$ is an array of length n. We can use the partition algorithm to solve this problem.

$$
\begin{array}{l}
\textit{Select}(A,\ s,\ e,\ k)\{ \\
\quad \textit{if } s = e \textit{ then return}(A[s]); \\
\quad m \leftarrow \textit{Partition}(A,\ s,\ e); \\
\quad \textit{switch}(\textit{compare}(k,\ m))\{ \\
\qquad \textit{case } k = m\ : \\
\qquad\quad \textit{return}(A[m]); \\
\qquad \textit{case } k < m\ : \\
\qquad\quad \textit{return}(\textit{Select}(A,\ s,\ m-1,\ k)); \\
\qquad \textit{case } k > m\ : \\
\qquad\quad \textit{return}(\textit{Select}(A,\ m+1,\ e,\ k-m)); \\
\quad \} \\
\}
\end{array}
$$

Selecting the k-th smallest number

Suppose that $A[s..e]$ is an array of length n. We can use the partition algorithm to solve this problem.

```
Select(A, s, e, k){
    if s = e then return(A[s]);
    m ← Partition(A, s, e);
    switch(compare(k, m)){
        case k = m :
            return(A[m]);
        case k < m :
            return(Select(A, s, m − 1, k));
        case k > m :
            return(Select(A, m + 1, e, k − m));
    }
}
```

The analysis of this version of Select algorithm is similar to the Quicksort algorithm. The best, worse, and average case complexity are $O(n)$, $O(n^2)$, and $O(n)$, respectively. (why?)

Selecting the k-th smallest number

Now we try to design an algorithm with $O(n)$ time complexity in worse case.

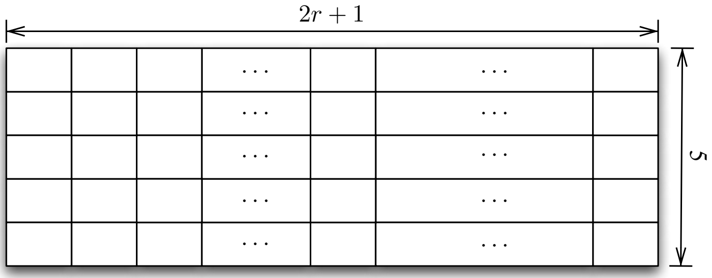
Selecting the k-th smallest number

Now we try to design an algorithm with $O(n)$ time complexity in worse case. Suppose that A is an array of length n, where $n = 5(2r+1)$ (if not, we can add some zeros).

Selecting the k-th smallest number

Now we try to design an algorithm with $O(n)$ time complexity in worse case. Suppose that A is an array of length n, where $n = 5(2r+1)$ (if not, we can add some zeros).

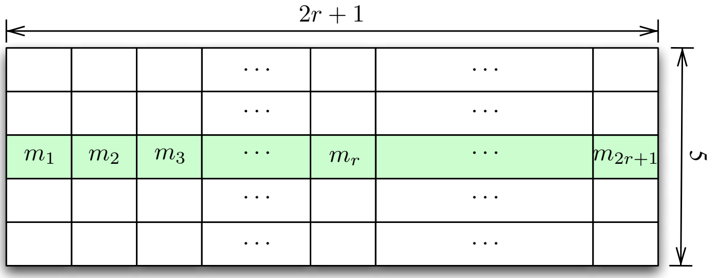
Step 1: Divide n elements into $2r+1$ groups, each of size 5 and arrange them as follows:



This step requires $O(1)$ time complexity.

Selecting the k-th smallest number

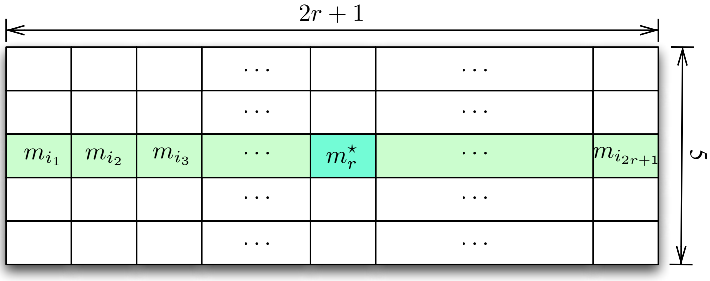
Step 2: Find the median in each column and place it in the middle as follows:



It requires 6 comparison in each column and so this step requires $6n/5 = 1.2n$.

Selecting the k-th smallest number

Step 3: Recursively call the algorithm to find the median of medians:



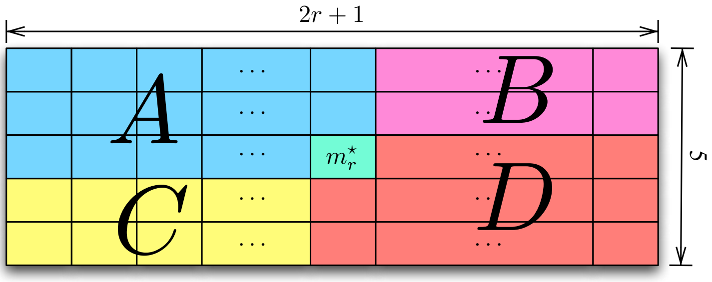
It requires $T(n/5) = T(0.2n)$ time complexity.

Selecting the k-th smallest number

Step 4: Constructing the following sets with respect to the value of $m_r^\star$ as follows:

$$L = A \cup \{x \mid x \in B \cup C \,\&\, x \leq m_r^\star\}$$

$$G = D \cup \{x \mid x \in B \cup C \,\&\, x \geq m_r^\star\}$$



This step requires $4r = 0.4n$ comparisons.

Selecting the k-th smallest number

Step 5:

- If $|L| = k - 1$ then $return(m_r^\star)$.
- Else if $|L| > k - 1$ then $return(Select(k, |L|))$.
- Else if $|L| < k - 1$ then $return(Select(k - |L| - 1, |G|)$.

Since the number of elements in L or G is at most $3r + 2 + 4r \simeq 7r$, so this step has $T(7r) = T(0.7n)$ time complexity. The overall time complexity of this algorithm is as follows:

$$T(n) = 1.6n + T(0.2n) + T(0.7n)$$

Selecting the k-th smallest number

$$T(n) = 1.6n + T(0.2n) + T(0.7n)$$

Selecting the k-th smallest number

$$T(n) = 1.6n + T(0.2n) + T(0.7n)$$

By using the induction, we show that $T(n) \leq 16n$.

- **Initiation:** $n = 5 \Longrightarrow T(5) \leq 16 \times 5. \surd$
- **Hypothesis:** $\forall i < n \Longrightarrow T(i) \leq 16i. \surd$

Selecting the k-th smallest number

$$T(n) = 1.6n + T(0.2n) + T(0.7n)$$

By using the induction, we show that $T(n) \leq 16n$.

- **Initiation:** $n = 5 \Longrightarrow T(5) \leq 16 \times 5. \surd$
- **Hypothesis:** $\forall i < n \Longrightarrow T(i) \leq 16i. \surd$
- **Induction step:** Prove the statement for n:

$$\begin{aligned} T(n) &= 1.6n + T(0.2n) + T(0.7n) \\ &\leq 1.6n + 16(0.2n) + 16(0.7n) \\ &= 1.6n + 3.2n + 11.2n \\ &= 16n. \end{aligned}$$

So $T(n) = O(n). \surd$

Exercises

1. Try to solve the select problem where each group contains j elements, instead of 5. The analyze you algorithm.
2. Let $X[1 \cdots n]$ and $Y[1 \cdots n]$ be two arrays, each containing n numbers already in sorted order. Give an $O(\log_2(n))$-time algorithm to find the median of all $2n$ elements in arrays X and Y .
3. Describe an $O(n)$-time algorithm that, given a set S of n distinct numbers and a positive integer $k \leq n$, determines the k numbers in S that are closest to the median of S.
4. For n distinct elements $x_1, x_2, \cdots, x_n$ with positive weights $w_1, w_2, \cdots, w_n$ such that $\sum_{i=1}^{n} w_i = 1$, the **weighted (lower) median** is the element x_k satisfying $\sum_{x_i < x_k} w_i < 1/2$ and $\sum_{x_i > x_k} w_i \leq 1/2$.
   a. Argue that the median of $x_1, x_2, \cdots, x_n$ is the weighted median of the x_i with weights $w_i = 1/n$ for $i = 1, 2, \cdots, n$.
   b. Show how to compute the weighted median of n elements in $O(n \log_2 n)$ worst-case time using sorting.
   c. Show how to compute the weighted median in $\Theta(n)$ worst-case time using a linear-time median algorithm.