COMP 3170 - Analysis of Algorithms & Data Structures

Shahin Kamali

Lecture 7 - Jan. 17, 2018
CLRS 7.1, 7-4, 9.1, 9.3
University of Manitoba
QuickSelect Review

\[\text{quick-select1}(A, i)\]
\[A: \text{array of size } n, \quad i: \text{integer s.t. } 0 \leq i < n\]

1. \(p \leftarrow \text{choose-pivot1}(A)\)
2. \(j \leftarrow \text{partition}(A, p)\)
3. \(\text{if } j = i \text{ then}\)
   4. \(\text{return } A[j]\)
5. \(\text{else if } j > i \text{ then}\)
   6. \(\text{return } \text{quick-select1}(A[0, 1, \ldots, j - 1], i)\)
7. \(\text{else if } j < i \text{ then}\)
   8. \(\text{return } \text{quick-select1}(A[j + 1, j + 2, \ldots, n - 1], i - j - 1)\)

- If pivot is at position \(j\), the cost of recursive call parameters will be:
  - None if \(j = i\).
  - \((j, i)\) if \(j > i\) (recursing on the left subarray).
  - \((n - j - 1, i - j - 1)\) if \(j < i\) (recursing on the right subarray).
Average-case analysis of quick-select1

Assume all \( n! \) permutations are equally likely.

Define \( T(n, i) \) as average cost for selecting \( i \)th item from size-\( n \) array:

The cost for recursive calls (RC) is

\[
RC = \begin{cases} 
0 & j = i \\
T(j, i), & j > i \\
T(n - j - 1, i - j - 1) & j < i 
\end{cases}
\]
Average-case analysis of quick-select

Assume all $n!$ permutations are equally likely.

Define $T(n, i)$ as average cost for selecting $i$th item from size-$n$ array:

The cost for recursive calls (RC) is

$$RC = \begin{cases} 
0 & j = i \\
T(j, i), & j > i \\
T(n - j - 1, i - j - 1), & j < i 
\end{cases}$$

Shuffled input $\rightarrow$ it is equally likely for the pivot to be at any position:

$$T(n, i) = \left\lfloor cn \right\rfloor + \frac{1}{n} \left( (\text{RC if } j=0) + (\text{RC if } j=1) + \ldots + (\text{RC if } j=n-1) \right)$$

$$= \left\lfloor cn \right\rfloor + \frac{1}{n} \left( \sum_{j=0}^{i-1} T(n - j - 1, i - j - 1) + \sum_{j=i+1}^{n-1} T(j, i) \right)$$

For simplicity, define $T(n) = \max_{0 \leq k \leq n} T(n, k)$. 
Average-case analysis of quick-select

\[ T(n) \leq \underbrace{cn}_{\text{partition}} + \frac{1}{n} \left( \sum_{j=0}^{i-1} T(n-j-1) + \sum_{j=i+1}^{n-1} T(j) \right) \]

- We say that a pivot is **good** if the arrays on both sides have size at least \( n/4 \)
  - This happens when pivot index \( j \) is in \([n/4, 3n/4)\).
  - Half of possible pivots are good and the rest are bad.
- The recursive cost for a good pivot is at most \( T(3n/4) \).
- The recursive cost for a bad pivot is at most \( T(n) \).

The average cost is then given by:

\[ T(n) \leq \begin{cases} 
  cn + \frac{1}{2} \left( T(n) + T([3n/4]) \right), & n \geq 2 \\
  d, & n = 1 
\end{cases} \]
Average-case analysis of quick-select

The average cost is then given by:

\[ T(n) \leq \begin{cases} 
  cn + \frac{1}{2} \left( T(n) + T(\lfloor 3n/4 \rfloor) \right), & n \geq 2 \\
  d, & n = 1 
\end{cases} \]
Average-case analysis of quick-select1

The average cost is then given by:

\[ T(n) \leq \begin{cases} 
  cn + \frac{1}{2} \left( T(n) + T(\lfloor 3n/4 \rfloor) \right), & n \geq 2 \\
  d, & n = 1 
\end{cases} \]

Rearranging gives:

\[ T(n) \leq 2cn + T(\lfloor 3n/4 \rfloor) \leq 2cn + 2c(3n/4) + 2c(9n/16) + \cdots + d \]

\[ \leq d + 2cn \sum_{i=0}^{\infty} \left( \frac{3}{4} \right)^i \in O(n) \]

Since \( T(n) \) must be \( \Omega(n) \) (why?), \( T(n) \in \Theta(n) \).
Linear-time selection

- Although Quick-select runs in $O(n)$ on average, in the worst-case it is still super-linear.

- Is there any selection algorithm that runs in $O(n)$ in the worst-case?

  - The answer is Yes; **Median of medians** algorithms!
  - It is a twist to Quick-select in which the pivot is selected a bit smarter!
Median of five algorithm

- A variant of Quick-select in which the pivot is selected more carefully.
- The input is an array $A$ of $n$ objects (assume $n$ is divisible by 5).
- Divide $A$ into $n/5$ blocks of size 5.
- Recursively find the median of the medians; denote it by $x$.
  - $x$ will be the pivot for quick-select
- Partition the whole array using $x$ as the pivot
- Recurs on the corresponding subarray as in Quick-select
### Median of five example

<table>
<thead>
<tr>
<th></th>
<th>2</th>
<th>54</th>
<th>44</th>
<th>4</th>
<th>25</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>32</td>
<td>18</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>87</td>
<td>21</td>
<td>26</td>
<td>47</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>9</td>
<td>13</td>
<td>16</td>
<td>56</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>10</td>
<td>2</td>
<td>19</td>
<td>71</td>
</tr>
</tbody>
</table>
Median of five example

\[
\begin{array}{cccccc}
2 & 5 & 2 & 4 & 25 \\ 
5 & 9 & 13 & 16 & 39 \\ 
9 & 10 & 21 & 18 & 47 \\ 
19 & 54 & 32 & 19 & 56 \\ 
24 & 87 & 44 & 26 & 71 \\
\end{array}
\]

Median of each group
Median of five example

Find X, the median of medians
Median of five algorithm

- Pivot $x$ is median of medians $\rightarrow$ half of blocks have median $< x$.
  - This implies half of blocks include at least 3 elements $< x$.
  - So, there will be at least $n/5 \cdot 1/2 \cdot 3 = 3n/10$ elements smaller than $x$

- Similarly, there will be at least $3n/10$ elements larger than $x$.

- We assume distinct items; when pivot is equal to multiple items, you can update the partition algorithm so that the pivot is the ‘best’ among items with the same key

- Hence, the size of recursive call is always in the range $(3n/10, 7n/10)$.
  - $x$ is always a ‘good’ pivot
Median of five algorithm

- Pivot $x$ is median of medians $\Rightarrow$ half of blocks have median $< x$.
  - This implies half of blocks include at least 3 elements $< x$.
  - So, there will be at least $n/5 \cdot 1/2 \cdot 3 = 3n/10$ elements smaller than $x$
- Similarly, there will be at least $3n/10$ elements larger than $x$.
- We assume distinct items; when pivot is equal to multiple items, you can update the partition algorithm so that the pivot is the ‘best’ among items with the same key.
- Hence, the size of recursive call is always in the range $(3n/10, 7n/10)$.
  - $x$ is always a ‘good’ pivot.
- In the worst case, the size of recursive call is always $7n/10$.

$$T(n) \leq \begin{cases} 
  T(n/5) + \underbrace{cn}_{\text{partition around } x} + T(7n/10), & n \geq 2 \\
  \underbrace{T(n/5)}_{\text{find } x} + \underbrace{d}_{\text{recursive call}}, & n = 1 
\end{cases}$$
Median of five algorithm

\[ T(n) \leq \begin{cases} 
T(n/5) + cn + T(7n/10), & n \geq 2 \\
\text{find } x, \text{ partition around } x, \text{ recursive call} & n = 1 
\end{cases} \]

- We guess that \( T(n) \in O(n) \) and use strong induction to prove it.
- We prove there is a value \( M \) s.t. \( T(n) \leq Mn \) for all \( n \geq 1 \).
- For the base we have \( T(1) = d \leq M \) as long as \( M \geq d \).
- For any value of \( n \) we can state:

\[
T(n) \leq T(n/5) + T(7n/10) + cn \quad \text{(definition)}
\leq M \cdot n/5 + M \cdot 7n/10 + cn \quad \text{(induction hypothesis)}
= (9M/10 + c)n
\leq M \cdot n \quad \text{as long as } M \geq 9M/10 + c, \ i.e., \ M \geq 10c
\]

so, we showed for \( M = \max\{10c, d\} \) we have \( T(n) \leq M \cdot n \) for \( n \geq 1 \). So, \( T(n) \in O(n) \).
Quick-sort revisit

**Theorem**

*It is possible to select the i’th smallest item in a list of n numbers in time $\Theta(n)$*

- Quick-sort in $O(n \log n)$ time:
  - Using select algorithm to choose the pivot as the **median** of $n$ items in $O(n)$ time
  - Partition around pivot in $O(n)$ time (selecting pivot as $n/c$’th smallest item for constant $c$ gives the same result)
  - Sort the two sides of pivot recursively in time $2T(n/2)$.

  The cost will be $T(n) = 2T(n/2) + \Theta(n)$, which gives $T(n) = \Theta(n \log n)$ [case II of Master theorem]

**Theorem**

*A smart selection of pivot, using linear-time select, results in quick-sort running in $\Theta(n \log n)$*