Graduate Algorithms

CS673-2016F-06

Selection Problem

David Galles

Department of Computer Science
University of San Francisco
What is the smallest exact number of comparisons required to find the maximum element of a list with $n$ elements?

What is the smallest exact number of comparisons required to find the minimum element of a list of $n$ elements?

What is the smallest number of comparisons required to find the maximum and minimum elements of a list?
06-1: Finding Max & Min

- What is the smallest number of comparisons required to find the maximum element of a list? $(n - 1)$
- What is the smallest number of comparisons required to find the minimum element of a list? $(n - 1)$
- What is the smallest number of comparisons required to find the maximum and minimum elements of a list?
  - Compare pairs: then compare the largest to the current largest, and smallest to the current smallest \( \left\lceil \frac{n}{2} \right\rceil + (n - 2) \)
What if we want to find the $k$th smallest element?

- Median: $k = \left\lceil \frac{n}{2} \right\rceil$th smallest element, for odd $n$

What is the obvious method?

Can we do better?
What if we want to find the $k$th smallest element?

- Median: $k = \left\lceil \frac{n}{2} \right\rceil$th smallest element, for odd $n$

What is the obvious method?

- Sort the list, select the element at index $k$

Can we do better?
Selection Problem

Quicksort(A, low, high)
  if (low < high) then
    pivotindex ← Partition(A, low, high)
    Quicksort(A, low, pivotindex 1)
    Quicksort(A, pivotindex + 1, high)

Partition(A, low, high)
  pivot = A[high]
  i ← low - 1
  for j ← low to high - 1 do
    if (A[j] ≤ pivot) then
      i ← i + 1
  return i+1
06-5: Selection Problem

Select(A,low,high,k)
  if (low = high)
    return A[low]
  pivot = Partition(A, low, high)
  adj_pivot = piv - low + 1
  if (k = adj_pivot) then
    return A[pivot]
  if (k < adj_pivot) then
    return Select(A,low,pivot-1, k)
  else
    return Select(A,pivot+1, high, k-adj_pivot)

Running time (Best and worst case)?
06-6: Selection Problem

- Best case time:
  \[ T(n) = T(n/2) + c \cdot n \in \Theta(n) \]

- Worst case time:
  \[ T(n) = T(n - 1) + c \cdot n \in \Theta(n^2) \]

- Average case time turns out to be \( \Theta(n) \), but we’d like to get the worst-case time down.
06-7: Selection Problem

- Improving worst-case time for selection
  - We need to guarantee a “good” pivot to get $\Theta(n)$ time for selection
  - How much time can we spend to find a good pivot, and still get $\Theta(n)$ time for selection?
Improving worst-case time for selection

- We need to guarantee a “good” pivot to get $\Theta(n)$ time for selection
- How much time can we spend to find a good pivot, and still get $\Theta(n)$ time for selection?
- $O(n)$!
Finding a “Good” pivot (one that is near the median) in linear time:

• Split the list into $\frac{n}{5}$ list of length 5
• Do an insertion sort on each of the $\frac{n}{5}$ lists to find the median of each of these lists
• Call select recursively to find the median of the $\frac{n}{5}$ medians
## Selection Problem

<p>| | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
06-11: Selection Problem
### 06-12: Selection Problem

<p>| | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- The red circle highlights the selected elements.

---

This diagram illustrates the selection problem, where certain elements are marked as selected.
How good is the pivot chosen by this method? How many elements are guaranteed to be less than the pivot?

- Each row has 5 elements
- Half of the rows will have a median less than the pivot
- Each of these rows will have 3 elements less than the pivot

\[ 3 \times \left\lfloor \frac{n}{5} \right\rfloor \frac{1}{2} \]
06-14: Selection Problem

3 \times \left\lceil \left\lceil \frac{n}{5} \right\rceil \frac{1}{2} \right\rceil

- Not all of those rows have exactly 3 elements less than the pivot:
  - The total number of elements might not be divisible by 5 (so one row would have < 5 elements)
  - The row containing the pivot itself only has 2 elements less than the pivot (not 3)
- So, we will omit those two rows, leaving:
  \[ 3 \times \left( \left\lceil \left\lceil \frac{n}{5} \right\rceil \frac{1}{2} \right\rceil - 2 \right) \geq \frac{3n}{10} - 6 \]
06-15: Selection Problem

- Worst case time for selection for a problem of size \( n \):
  - \( \Theta(n) \) time to do partition, \( n/5 \) insertion sorts
  - Time to find the median of medians (looking at \( n/5 \) elements)
  - Time to make the recursive call (to a problem of no more than size \( 7n/10 + 6 \))
Selection Problem

\[ T(n) \leq \begin{cases} 
C_1 \\
T(\lceil \frac{n}{5} \rceil) + T(\frac{7n}{10} + 6) + C_2 \times n 
\end{cases} \]

if \( n < 140 \)

otherwise
06-17: Selection Problem

\[ T(n) \leq T\left(\left\lfloor \frac{n}{5} \right\rfloor \right) + T\left(\frac{7n}{10} + 6\right) + C_2 \cdot n \]

\[ \leq \frac{C \cdot n}{5} + C + \frac{7 \cdot C \cdot n}{10} + 6 \cdot C + C_2 \cdot n \]

\[ = \frac{9 \cdot C \cdot n}{10} + 7 \cdot C + C_2 \cdot n \]

\[ = C \cdot n + \left(7 \cdot C + C_2 \cdot n - \frac{C \cdot n}{10}\right) \]
06-18: Selection Problem

\[7 \times C + C_2 \times n - \frac{C \times n}{10} \leq 0\]

\[C \times \left(7 - \frac{n}{10}\right) \leq -C_2 \times n\]

\[C \times \left(\frac{n}{10} - 7\right) \geq C_2 \times n\]

\[C \geq C_2 \times \frac{n}{(n/10 - 7)}\]

\[C \geq 10 \times C_2 \times \frac{n}{(n - 70)}\]

Note that we must insist that \(n > 70\). If \(n \geq 140\), then this is true if \(C > 20 \times C_2\).
Selection Problem

- Selection takes time $O(n)$
  - in fact, $\Theta(n)$, since each recursion step takes time $\Omega(n)$

- So, we can use Selection to make Quicksort take time $\Theta(n \log n)$ worst case
  - Would that be a good idea?