Topic 1: Introduction and Median Finding

David Woodruff

Course homepage:
http://www.cs.cmu.edu/~dwoodruf/teaching/15451-win20/cis.html

Grading and Course Policies

3 Written Homeworks 30% (10% each)
Class Participation 10%
1 Exam (in class) 20%
Research project 40%

Schedule of Lectures and Exams

• Fridays 7-10pm ET
  • On zoom
• First four weeks the lectures will cover theoretical background
• Remaining weeks the lectures will be project-oriented
• Exam: second half of lecture on 1/31

Homework

• HW1: out 1/10, due 1/20 at 11:59pm China time
• HW2: out 1/21, due at 1/27 at 11:59pm China time
• HW3: out 1/28, due 2/3 at 11:59pm China tie
• For the homework, you will be asked to design and/or analyze algorithms. Your solution should be written up formally – that is, you should prove your claims.
• You can work by yourself or with at most one other person - you must list your collaborator (if you have one) and write the solutions yourself. You’re allowed to read additional textbooks or online notes but must cite them.
Schedule of Topics

- 1/10: median-finding and concrete upper and lower bounds
- 1/17: hashing and streaming
- 1/24: fingerprinting and game theory
- 1/31: linear programming and Exam

Goals of the Course

- Design and analyze algorithms!
  - Algorithms: dynamic programming, divide-and-conquer, hashing and data structures, randomization, network flows, linear programming
  - Analysis: recurrences, probabilistic analysis, amortized analysis, potential functions
  - Dual to Algorithms: complexity theory and lower bounds
  - New Models: online algorithms, machine learning, data streams

Guarantees on Algorithms

- Want provable guarantees on the running time of algorithms
- Why?
  - Composability: if we know an algorithm runs in time at most T on any input, don’t have to worry what kinds of inputs we run it on
  - Scaling: how does the time grow as the input size grows?
  - Designing better algorithms: what are the most time-consuming steps?

Example: Median Finding

- In the median-finding problem, we have an array
  \[ a_1, a_2, \ldots, a_n \]
  and want the index \( i \) for which there are exactly \( \lfloor n/2 \rfloor \) numbers larger than \( a_i \)
- How can we find the median?
  - Check each item to see if it is the median: \( \Theta(n^2) \) time
  - Sort items with MergeSort (deterministic) or QuickSort (randomized): \( \Theta(n \log n) \) time
  - Can we find it faster? What about finding the k-th smallest number?
QuickSelect Algorithm to Find the k-th Smallest Number

• Assume \( a_1, a_2, \ldots, a_n \) are all distinct for simplicity

• Choose a random element \( a_i \) in the list – call this the “pivot”

• Compare each \( a_j \) to \( a_i \)
  • Let LESS = \{\( a_j \) such that \( a_j < a_i \)\}
  • Let GREATER = \{\( a_j \) such that \( a_j > a_i \)\}

• If \( k \leq |\text{LESS}| \), find the k-th smallest element in LESS
• If \( k = |\text{LESS}| + 1 \), output the pivot \( a_i \)
• Else find the \((k-|\text{LESS}|-1)\)-th smallest item in GREATER

• Similar to Randomized QuickSort, but only recurse on one side!

Bounding the Running Time

• Theorem: the expected number of comparisons for QuickSelect is at most 4n

• Let \( T(n) = \max T(n,k) \), where \( T(n,k) \) is the expected number of comparisons to find the k-th smallest item in an array of length n, maximized over all arrays

• \( T(n) \) is a non-decreasing function of n

• Let’s show \( T(n) < 4n \) by induction

• Base case: \( T(1) = 0 < 4 \)

• Inductive hypothesis: \( T(n-1) < 4(n-1) \)

What About Deterministic Algorithms?

• Can we get an algorithm which does not use randomness and always performs \( O(n) \) comparisons?

• Idea: suppose we could deterministically find a pivot which partitions the input into two pieces LESS and GREATER each of size \( \left\lfloor \frac{n}{2} \right\rfloor \)

• How to do that?

• Find the median and then partition around that
  • Um... finding the median is the original problem we want to solve....
Deterministically Finding a Pivot

• **Idea:** deterministically find a pivot with \( O(n) \) comparisons to partition the input into two pieces LESS and GREATER each of size at least \( 3n/10 \)

• **DeterministicSelect:**
  1. Group the array into \( n/5 \) groups of size 5 and find the median of each group
  2. Recursively, find the median of medians. Call this \( p \)
  3. Use \( p \) as a pivot to split into subarrays LESS and GREATER
  4. Recurse on the appropriate piece

• **Theorem:** DeterministicSelect makes \( O(n) \) comparisons to find the \( k \)-th smallest item in an array of size \( n \)

Running Time of DeterministicSelect

• **DeterministicSelect:**
  1. Group the array into \( n/5 \) groups of size 5 and find the median of each group
  2. Recursively, find the median of medians. Call this \( p \)
  3. Use \( p \) as a pivot to split into subarrays LESS and GREATER
  4. Recurse on the appropriate piece

• Step 1 takes \( O(n) \) time since it takes \( O(1) \) time to find the median of 5 elements
• Step 2 takes \( T(n/5) \) time
• Step 3 takes \( O(n) \) time

Claim: \(|LESS| \geq 3n/10 - 1 \) and \(|GREATER| \geq 3n/10 - 1 \)

Example 1: If \( n = 15 \), we have three groups of 5:
\( \{1, 2, 3, 10, 11\}, \{4, 5, 6, 12, 13\}, \{7, 8, 9, 14, 15\} \)

medians: \( 3, 6, 9 \)

median of medians \( p \): 6

There are \( g = n/5 \) groups, and at least \( \left\lceil \frac{g}{2} \right\rceil \) of them have at least 3 elements at most \( p \). The number of elements less than or equal to \( p \) is at least \( \frac{3}{2} \cdot \frac{n}{10} \)

Also at least \( 3n/10 \) elements greater than or equal to \( p \)

Running Time of DeterministicSelect

• **Claim:** \(|LESS| \geq 3n/10 - 1 \) and \(|GREATER| \geq 3n/10 - 1 \)

• **Example 1:** If \( n = 15 \), we have three groups of 5:
  \( \{1, 2, 3, 10, 11\}, \{4, 5, 6, 12, 13\}, \{7, 8, 9, 14, 15\} \)

medians: \( 3, 6, 9 \)

median of medians \( p \): 6

• Steps 1-3 take \( O(n) \) \( + \) \( T(n/5) \) time
• Since \(|LESS| \geq 3n/10 - 1 \) and \(|GREATER| \geq 3n/10 - 1 \), Step 4 takes at most \( T(7n/10) \) time

So \( T(n) \leq cn + T \left( \frac{n}{2} \right) + T \left( \frac{3n}{10} \right) \), for a constant \( c > 0 \)
Running Time of DeterministicSelect

- $T(n) \leq cn + T\left(\frac{n}{2}\right) + T\left(\frac{7n}{10}\right)$

<table>
<thead>
<tr>
<th></th>
<th>cn</th>
<th></th>
<th>cn/5</th>
<th>7cn/10</th>
<th>8cn/10</th>
<th>...</th>
<th>...</th>
<th>Total: cn</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9cn/10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9cn/10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8cn/10</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8cn/10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

- Time is $cn \left(1 + \left(\frac{9}{10}\right) + \left(\frac{7}{10}\right)^2 + \ldots\right) \leq 10cn$
- Recurrence works because $n/5 + 7n/10 < n$

- For constants $c$ and $a_1, a_2, \ldots, a_r$, with $a_1 + a_2 + \ldots + a_r < 1$, the recurrence $T(n) \leq T(a_1n) + T(a_2n) + \ldots + T(a_rn) + cn$ solves to $T(n) = O(n)$
  - If instead $a_1 + a_2 + \ldots + a_r = 1$, the recurrence solves to $T(n) = O(n \log n)$
  - If we use median of 3 in DeterministicSelect instead of median of 5, what happens?