CS3000: Algorithms & Data
Jonathan Ullman

Lecture 3:
• Divide and Conquer: Karatsuba
• Solving Recurrences

Jan 13, 2020
The “Master Theorem”

• Recipe for recurrences of the form:
  • \( T(n) = a \cdot T(n/b) + Cn^d \)

• Three cases:
  • \( \left( \frac{a}{b^d} \right) > 1 : T(n) = \Theta(n^{\log_b a}) \)
  
  • \( \left( \frac{a}{b^d} \right) = 1 : T(n) = \Theta(n^d \log n) \)
  
  • \( \left( \frac{a}{b^d} \right) < 1 : T(n) = \Theta(n^d) \)
• Use the Master Theorem to Solve:

  • \( T(n) = 16 \cdot T \left( \frac{n}{4} \right) + Cn^2 \)

  • \( T(n) = 21 \cdot T \left( \frac{n}{5} \right) + Cn^2 \)

  • \( T(n) = 2 \cdot T \left( \frac{n}{2} \right) + C \)

  • \( T(n) = 1 \cdot T \left( \frac{n}{2} \right) + C \)
Divide and Conquer: Selection (Median)
Selection

- Given an array of numbers $A[1: n]$, how quickly can I find the:
  - Smallest number?
  - Second smallest?
  - $k$-th smallest?
## Selection

- **Fact:** can select the $k$-th smallest in $O(nk)$ time
- **Fact:** can select the $k$-th smallest in $O(n \log n)$ time
  - Sort the list, then return $A[k]$  

<table>
<thead>
<tr>
<th>11</th>
<th>3</th>
<th>42</th>
<th>28</th>
<th>17</th>
<th>8</th>
<th>2</th>
<th>15</th>
</tr>
</thead>
</table>

- **Today:** select the $k$-th smallest in $O(n)$ time
Warmup

• You have 25 horses and want to find the 3 fastest
• You have a racetrack where you can race 5 at a time
  • In: {1, 5, 6, 18, 22}  Out: (6 > 5 > 18 > 22 > 1)
• **Problem:** find the 3 fastest with only seven races
Median Algorithm: Take I

Select(A[1:n],k):
  If(n = 1): return A[1]

Choose a {\color{red}pivot} \( p = A[1] \)
Partition around the pivot, let \( p = A[r] \)

If(k = r): return A[r]
ElseIf(k < r): return Select(A[1:r-1],k)
ElseIf(k > r): return Select(A[r+1:n],k-r)
Median Algorithm: Take I
Median Algorithm: Take II

• **Problem:** we need to find a good pivot element
Median of Medians

\[ \text{MOM}(A[1:n]) : \]
\[
\text{Let } m \leftarrow \left\lceil \frac{n}{5} \right\rceil \\
\text{For } i = 1, \ldots, m: \\
\quad \text{Meds}[i] = \text{median}\{A[5i-4], A[5i-3], \ldots, A[5i]\} \\
\quad \text{Let } p \leftarrow \text{Select}(\text{Meds}[1:m], \left\lceil \frac{m}{2} \right\rceil) \\
\]
Median of Medians

- **Claim**: For every $A$ here are at least $3n/10$ items that are smaller than $\text{MOM}(A)$ and at least $3n/10$ items that are larger.

Visualizing the median of medians
Median Algorithm: Take II

\[
\begin{array}{cccccccccc}
17 & 3 & 42 & 11 & 28 & 8 & 2 & 15 & 13 & A \\
\end{array}
\]

\[
\begin{array}{cccccccccc}
11 & 3 & 5 & 13 & 2 & 8 & 17 & 28 & 42 & \\
\end{array}
\]

\[
\text{MOMSelect}(A[1:n], k):
\]
\[
\text{If}(n \leq 25): \text{return} \text{ median}\{A\}
\]

\[
\text{Let } p = \text{ MOM}(A)
\]
\[
\text{Partition around the pivot, let } p = A[r]
\]

\[
\text{If}(k = r): \text{return} A[r]
\]
\[
\text{ElseIf}(k < r): \text{return} \text{ MOMSelect}(A[1:r-1], k)
\]
\[
\text{ElseIf}(k > r): \text{return} \text{ MOMSelect}(A[r+1:n], k-r)
\]
Running Time Analysis
Recursion Tree

\[ T(n) = T \left( \frac{7n}{10} \right) + T \left( \frac{n}{5} \right) + Cn \]

\[ T(1) = C \]
Proof by Induction

• **Claim:** $T(n) = O(n)$

\[
T(n) = T\left(\frac{7n}{10}\right) + T\left(\frac{n}{5}\right) + Cn
\]

$T(1) = C$
Ask the Audience

• If we change MOM so that it uses n/3 blocks of size 3, would Select still run in O(n) time?
Selection Wrapup

• Find the $k$-th largest element in $O(n)$ time
  • Selection is strictly easier than sorting!
• Divide-and-conquer approach
  • Find a pivot element that splits the list roughly in half
  • Key Fact: median-of-medians-of-five is a good pivot
• Can sort in $O(n \log n)$ time using same technique
  • Algorithm is called Quicksort
• Analyze running time via recurrence
  • Master Theorem does not apply
• Fun Fact: a random pivot is also a good pivot!
Divide and Conquer: Binary Search
Binary Search

Is 28 in this list?

2  3  8  11  15  17  28  42  A
Binary Search

StartSearch(A,t):
  // A[1:n] sorted in ascending order
  Return Search(A,1,n,t)

Search(A,ℓ,r,t):
  If(ℓ > r): return FALSE

  \[ m \leftarrow \ell + \left\lfloor \frac{r-\ell}{2} \right\rfloor \]

  If(A[m] = t): return m
  ElseIf(A[m] > t): return Search(A,ℓ,m-1,t)
  Else: return Search(A,m+1,r,t)
Running Time Analysis

\[
T(n) = T(n/2) + C \\
T(1) = C
\]
Binary Search Wrapup

• Search a sorted array in time $O(\log n)$
• Divide-and-conquer approach
  • Find the middle of the list, recursively search half the list
  • **Key Fact:** eliminate half the list each time
• Prove correctness via induction
• Analyze running time via recurrence
  • $T(n) = T(n/2) + C$