Lecture 5:
• Divide-and-Conquer: more examples
Divide-and-Conquer: Binary Search
Binary Search

Is 28 in this list? If so, where.

| 2 | 3 | 8 | 11 | 15 | 17 | 28 | 42 |

↑ Is this 28? No.

Is it bigger or smaller? Smaller.

| 15 | 17 | 28 | 42 |

↑

| 28 | 42 |

↑ Found it!

Binary Search

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

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

   m ← ℓ + \left\lfloor \frac{r-ℓ}{2} \right\rfloor

   If(A[m] = t): Return m
   ElseIf(A[m] > t): Return BS(A, ℓ, m-1, t)
   Else: Return BS(A, m+1, r, t)

T(n) = time to search list of size n

T(n) = T(\frac{n}{2}) + C

T(1) = C
Correctness of Binary Search

Claim: Binary Search is correct

\[
\forall n \in \mathbb{N} \quad \forall \text{arrays } A \text{ of size } n \quad \forall t \exists i \text{ s.t. } A[i] = t
\]

\[
\text{Search}(A, t) = \begin{cases} i & \text{if } t \in A \\ \bot & \text{otherwise} \end{cases}
\]

\[
\forall \text{arrays } A \quad \forall n \in \mathbb{N} \quad \forall l, r \text{ s.t. } r - l \leq n \quad \forall t
\]

\[
\text{BS}(A, l, r, t) = \begin{cases} i & \text{if } t \in A[l..r] \\ \bot & \text{otherwise} \end{cases}
\]

\[H(n) \text{ Inductive Hypothesis}\]

Base Case: \(H(1)\) is correct
Inductive Step: Assume $H(n)$ let $l, r$ s.t. $r-l=n+1$

$m \leftarrow l + \left\lfloor \frac{r-l}{2} \right\rfloor$

case 1: $A[m] = t$, $BS(A, l, r, t) = m$ which is correct

case 2: $A[m] < t$, we output $\boxed{BS(A, m+1, r, t)}$

  * if we get $i$ s.t. $A[i] = t$
  * if we get $l$, then $t \not\in A[m+1 : r]$

  we also know $t \not\in A[l : m]$ because $A$ is sorted and $A[m] < t$

  $\Rightarrow t \in A[l : r]$ so we are correct

case 3: $A[m] > t$ is symmetric
Ask the Audience

- What is the running time of binary search?
  - What is the recurrence?
  - What is the solution to the recurrence?

\[
T(n) = T\left(\frac{n}{2}\right) + C
\]

\[
T(1) = C
\]

\[
T(n) = a \cdot T\left(\frac{n}{2}\right) + Cn^d
\]

\[
a = 1, \quad b = 2, \quad d = 0
\]

\[
\frac{a}{b^d} = \frac{1}{2^0} = 1
\]

\[
T(n) = \Theta(n^d \log n) = \Theta(\log n)
\]
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
  • $T(n) = T(n/2) + C$
Selection (Median)
Selection

• Given an array of numbers $A[1, ..., n]$, how quickly can I find the:
  • Smallest number? $O(n)$ time
  • Second smallest? $O(n)$
  • $k$-th smallest? $O(nk)$
  • median? $\left\lceil \frac{n}{2} \right\rceil$th smallest number $\Theta(n^2)$
Selection

\[ O(nk) \text{ or } O(n \log n) \]
\[ O(n \log k) \]

- **Fact:** can select the \( k \)-th smallest in \( O(n \log n) \) time
  - Sort the list and look up \( A[k] \)

<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>

\[ A \]

\[ \text{sort} \]

| 2  | 3  | 8  | 11 | 15 | 17 | 28 | 42 |

- **Today:** select the \( k \)-th smallest in \( O(n) \) time
Median Algorithm: Take I

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

  Choose a 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)
```

\[ A = \begin{bmatrix}
  17 & 3 & 42 & 11 & 28 & 8 & 2 & 15 & 13 \\
  11 & 3 & 5 & 13 & 2 & 8 & 17 & 28 & 42 \\
\end{bmatrix} \]

\[ r = 7, A[r] = p \]

*partitioning: splitting into* \[ \langle p \rangle \]
\[ T(n) = T\left(\frac{n}{2}\right) + Cn \]
\[ T(n) = O(n) \]

\[ T(n) = T\left(\frac{3n}{4}\right) + Cn \]
\[ = O(n) \]
Median Algorithm: Take I

\[ \begin{array}{cccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\
\end{array} \]

\[ \begin{array}{cccccc}
2 & \text{---} & \text{---} & \text{---} & 9 \\
\end{array} \]

\[ \begin{array}{cccccc}
3 & \text{---} & \text{---} & \text{---} & 9 \\
\end{array} \]

time to find \( k \)th smallest

\[ \sum_{i=0}^{k} n - i \approx nk - k^{2/2} = \Theta(nk) \]

\[ T(n) = T(n-1) + Cn \]
Median Algorithm: Take II

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

  • Perfect pivot elt is the median

  • Enough to have an elt in the middle half of the list

    ![Diagram](image_url)

    ... as long as we can find it quickly.
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 of Medians

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

• **Claim:** For every $A$ here are at least $\frac{3n}{10}$ items that are smaller than MOM($A$)

\[
(3 \text{ rows of elts}) \times (\frac{n}{10} \text{ columns}) = \frac{3n}{10}
\]

• Also $\frac{3n}{10}$ are large than the MOM
Median Algorithm: Take II

```plaintext
MOMSelect(A[1:n],k):
    If(n ≤ 25): return median{A}

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

    If(k = r): return A[r]
    ElseIf(k < r): return MOMSelect(A[1:r-1],k)
    ElseIf(k > r): return MOMSelect(A[r+1:n],k-r)
```
Running Time Analysis

\[ T(n) = \text{time to select with } n \text{ elts} \]

\[ T(n) = T\left(\frac{7n}{10}\right) + \left[ \text{time to find the pivot} \right] + C_n \]

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

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

\[ T(n) = T\left(\frac{7n}{10}\right) + T\left(\frac{2n}{10}\right) + Cn \]

\[ T(1) = C \]
Ask the Audience

• If we change MOM so that it uses \( \frac{n}{3} \) blocks of size 3, how many items can we eliminate?

\[
2 \times \left( \frac{n}{6} \right) = \frac{n}{3} \text{ elts}
\]

• What is the new running time of the algorithm?

\[
T(n) = T\left( \frac{n}{3} \right) + T\left( \frac{2n}{3} \right) + Cn
\]

\[
T(n) = \Theta(n \log n)
\]
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!