LECTURE 6
More Divide and Conquer
• Binary Search
Review questions

• How long does Merge Sort take on a sorted input?
  
  (Answer: $\Theta(n \log n)$. The exact input only affects the constants in the running time of the merge operation.)

• In a full tree of depth $d$, where every node above the bottom level has three children, what is the number of leaves?
  – If there are $n$ leaves, what is $d$?
    
    (Answer: # of leaves = $3^d$; depth = $\log_3(n)$ )
Divide-and-conquer design

1. **Divide** the problem (instance) into subproblems.

2. **Conquer** the subproblems by solving them recursively.

3. **Combine** subproblem solutions.
Binary search

Find an element in a sorted array:

1. **Divide**: Check middle element.
2. **Conquer**: Recursively search 1 subarray.
3. **Combine**: Trivial.

**Example**: Find 9

```
3  5  7  8  9  12  15
```
Binary search

Find an element in a sorted array:

1. **Divide**: Check middle element.
2. **Conquer**: Recursively search 1 subarray.
3. **Combine**: Trivial.

**Example**: Find 9

3  5  7  **8**  9  12  15
Binary search

Find an element in a sorted array:

1. **Divide**: Check middle element.
2. **Conquer**: Recursively search 1 subarray.
3. **Combine**: Trivial.

**Example**: Find 9

3 5 7 8 9 12 15
Binary search

Find an element in a sorted array:

1. **Divide**: Check middle element.
2. **Conquer**: Recursively search 1 subarray.
3. **Combine**: Trivial.

**Example**: Find 9

3 5 7 8 9 12 15
Binary search

Find an element in a sorted array:

1. **Divide**: Check middle element.
2. **Conquer**: Recursively search 1 subarray.
3. **Combine**: Trivial.

**Example**: Find 9

3 5 7 8 9 12 15
Binary search

Find an element in a sorted array:

1. Divide: Check middle element.
2. Conquer: Recursively search 1 subarray.

Example: Find 9

3  5  7  8  9  12  15
Binary Search

\[ \text{\textsc{BinarySearch}}(b, A[1 \ldots n]) \triangleright \text{find } b \text{ in sorted array } A \]

1. If \( n=0 \) then return “not found”
2. If \( A[\lceil n/2 \rceil] = b \) then return \( \lfloor n/2 \rfloor \)
3. If \( A[\lceil n/2 \rceil] < b \) then
4. \hspace{1em} return \text{\textsc{BinarySearch}}(A[1 \ldots \lceil n/2 \rceil])
5. Else
6. \hspace{1em} return \( \lceil n/2 \rceil + \text{\textsc{BinarySearch}}(A[\lceil n/2 \rceil+1 \ldots n]) \)
Proof of correctness

• Omitted here, similar to Merge-Sort

9/11/09

S. Raskhodnikova and A. Smith. Based on notes by E. Demaine and C. Leiserson
Recurrence for binary search

\[ T(n) = 1 + T(n/2) + \Theta(1) \]

- # subproblems
- subproblem size
- work dividing and combining
Recurrence for binary search

\[
T(n) = T(n/2) + \Theta(1)
\]

- # subproblems
- subproblem size
- work dividing and combining

\[
\Rightarrow T(n) = T(n/2) + c = T(n/4) + 2c \\
\ldots \\
= c \left\lceil \log n \right\rceil = \Theta(\lg n).
\]