Divide and Conquer: Recipe

- Divide problem into several parts
- Solve each part recursively
- Combine solutions to sub-problems into overall solution

Comparison

<table>
<thead>
<tr>
<th></th>
<th>Greedy</th>
<th>Divide and Conquer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formulate problem</td>
<td>?</td>
<td>?</td>
</tr>
<tr>
<td>Design algorithm</td>
<td>easy</td>
<td>hard</td>
</tr>
<tr>
<td>Prove correctness</td>
<td>hard</td>
<td>easy</td>
</tr>
<tr>
<td>Analyze running time</td>
<td>easy</td>
<td>hard</td>
</tr>
</tbody>
</table>

Motivating Problem: Maximum Subsequence Sum (MSS)

- **Input:** array $A$ of $n$ numbers, e.g.
  
  $$A = [4, -3, 5, -2, 1, 2, 6, -2]$$

- **Find:** value of the largest subsequence sum
  

  (empty subsequence allowed and has sum zero)

- **MSS in example?** 11 (first 7 elements)

What is a simple algorithm for MSS?

Anyone remember HW2?

MSS($A$)

1. Initialize all entries of $n \times n$ array $B$ to zero
2. for $i = 1$ to $n$
   1. sum = 0
   2. for $j = i$ to $n$
      1. sum += $A[j]$
      2. $B[i, j] = $ sum
   end for
3. end for
4. Return maximum entry of $B[i, j]$

**Running time?** $O(n^2)$. Can we do better?
Divide-and-conquer for MSS

- Recursive solution for MSS
  - **Idea:**
    - Find MSS \( L \) in left half of array
    - Find MSS \( R \) in right half of array
    - Find MSS \( M \) for sequence that crosses the midpoint

\[
A = \frac{4, -3, 5}{-2, 6, -2}, \quad L = 6, \quad R = 8
\]

\[
M = \frac{11, 4, -3, 5}{-2, 2, 6, -2}
\]

- Return \( \max(L, R, M) \)

MSS(\( A, \) left, right)
if \( \text{left} == \text{right} \) then
return \( \max(A[\text{left}], 0) \)
end if

\[
\text{mid} = \lfloor \frac{\text{left} + \text{right}}{2} \rfloor
\]

\[
L = \text{MSS}(A, \text{left}, \text{mid})
\]

\[
R = \text{MSS}(A, \text{mid}+1, \text{right})
\]

Set \( \text{sum} = 0 \) and
for \( i = \text{mid} \) down to 1 do
\[
\text{sum} += A[i]
\]
\[
L' = \max(L', \text{sum})
\]
end for

Set \( \text{sum} = 0 \) and \( R' = 0 \)
for \( i = \text{mid}+1 \) to right do
\[
\text{sum} += A[i]
\]
\[
R' = \max(R', \text{sum})
\]
end for

\[
M = L' + R'
\]
return \( \max(L, R, M) \)

Running time?
- Let \( T(n) \) be running time of MSS on array of size \( n \)
- Two recursive calls on arrays of size \( n/2 \): \( 2T(n/2) \)
- Work outside of recursive calls: \( O(n) \)
- Running time
\[
T(n) = 2T(n/2) + O(n)
\]

Recurrence

- Recurrence with convenient base case
  \[
  T(n) = 2T(n/2) + O(n)
  \]
  \[
  T(2) = O(1)
  \]
- How do we solve the recurrence to find a simple expression for \( T(n) \)?
  First, let’s use definition of Big-O:
  \[
  T(n) \leq 2T(n/2) + cn
  \]
  \[
  T(2) \leq c
  \]
- What next?

Solving a Recurrence

- **Idea 1:** “unroll” the recurrence
  \[
  T(n) \leq 2T(n/2) + cn
  \]
  \[
  \leq 2 \left[ 2T(n/4) + c(n/2) \right] + cn
  \]
  \[
  \leq 2 \left[ 2 \left( 2T(n/8) + c(n/4) \right) + c(n/2) \right] + cn
  \]
- This will work, but can get messy in a hurry…

Solving a Recurrence

- **Idea 2:** recursion tree (same idea, different organization)
  - Board work
  - **Conclusion:** \( T(n) \leq cn \log n \)
Solving a Recurrence

- Idea 3: “guess and verify”
  - Guess solution
  - Prove by (strong) induction
  - Board work

A More General Recurrence

\[ T(n) \leq q \cdot T(n/2) + cn \]

- What does the algorithm look like?
  - \( q \) recursive calls to itself on problems of half the size
  - \( O(n) \) work outside of the recursive calls
- Exercises: \( q = 1, q > 2 \)
- Useful fact (geometric sum): if \( r \neq 1 \) then
  \[
  1 + r + r^2 + \ldots + r^d = \frac{1 - r^{d+1}}{1 - r} = \frac{r^{d+1} - 1}{r - 1}
  \]

Summary

Useful general recurrence and its solutions:

\[ T(n) \leq q \cdot T(n/2) + cn \]

1. \( q = 1 \): \( T(n) = O(n) \)
2. \( q = 2 \): \( T(n) = O(n \log n) \)
3. \( q > 2 \): \( T(n) = O(n \log_2 q) \)

Algorithms with these running times?

1. ???
2. MSS, Mergesort
3. Integer multiplication (next time)