

# CMPSCI 311: Introduction to Algorithms

## Intro to Divide and Conquer

Dan Sheldon

University of Massachusetts

Last Compiled: March 13, 2017

## Algorithm Design Techniques

- ▶ Greedy
- ▶ **Divide and Conquer**
- ▶ Dynamic Programming
- ▶ Network Flows

## Divide and Conquer: Recipe

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

## Comparison

	Greedy	Divide and Conquer
Formulate problem	?	?
Design algorithm	easy	hard
Prove correctness	hard	easy
Analyze running time	easy	hard

## 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**

$$A[i] + A[i + 1] + \dots + A[j]$$

- ▶ (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( $A$ )

Initialize all entries of  $n \times n$  array  $B$  to zero

```
for  $i = 1$  to  $n$  do  
    sum = 0  
    for  $j = i$  to  $n$  do  
        sum +=  $A[j]$   
         $B[i, j] =$  sum  
    end for
```

**end for**

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

$$A = \underbrace{4, -3, 5}_{L=6}, \underbrace{-2, -1, 2, 6}_{R=8}, -2 \quad M=11$$

- ▶ Return  $\max(L, R, M)$

MSS( $A$ , left, right)

```
if left == right then
    return max(A[left], 0)
end if
```

▷ Base case

```
mid = ⌊(left+right)/2⌋
L = MSS(A, left, mid)
R = MSS(A, mid+1, right)
```

▷ Recurse on left and right halves

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

▷ Compute  $L'$  (left part of  $M$ )

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

▷ Compute  $R'$  (right part of  $M$ )

```
M = L' + R'
```

▷ Compute  $M$

```
return max(L, R, M)
```

▷ Return max

MSS( $A$ , left, right)

```
if left == right then
    return max(A[left], 0)
end if
```

```
mid = ⌊(left+right)/2⌋
L = MSS(A, left, mid)
R = MSS(A, mid+1, right)
```

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

```
Set sum = 0 and R' = 0
for i = mid+1 to right do
    sum += A[i]
    R' = max(R', 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

$$\begin{aligned} T(n) &\leq 2T(n/2) + cn \\ &\leq 2[2T(n/4) + c(n/2)] + cn \\ &\leq 2[2(2T(n/8) + c(n/4)) + c(n/2)] + cn \end{aligned}$$

- ▶ 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

$$1 + r + r^2 + \dots + r^d = \frac{1 - r^{d+1}}{1 - r} = \frac{r^{d+1} - 1}{r - 1}$$

## 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)