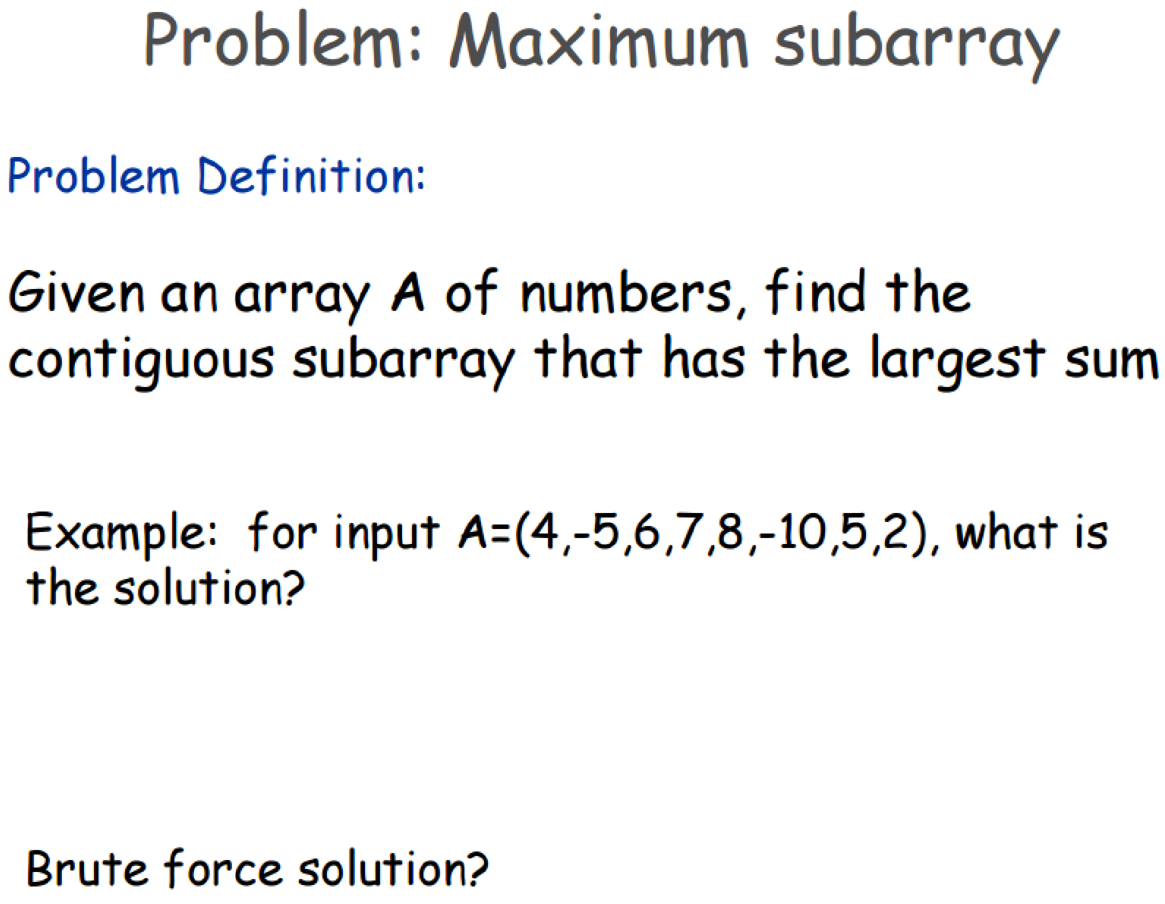
Week 3 Thursday Lecture Notes

Prep For Class:

- Review Recitation Questions for Quiz 2 Today.
- Creating and Solving Algorithms for Runtime Complexity.

Lecture 5: Maximum Subarray



Problem: Maximum subarray

Problem Definition:

Given an array A of numbers, find the contiguous subarray that has the largest sum

Example: for input A=(4,-5,6,7,8,-10,5,2), what is the solution?

Brute force solution?

- Given an Array A, find the contiguous subarray that has the largest sum.
- Example Array A= (4,-5,6,7,8,-10,5,2).
- Maximum Contagious Subarray is [6,7,8] = 21.

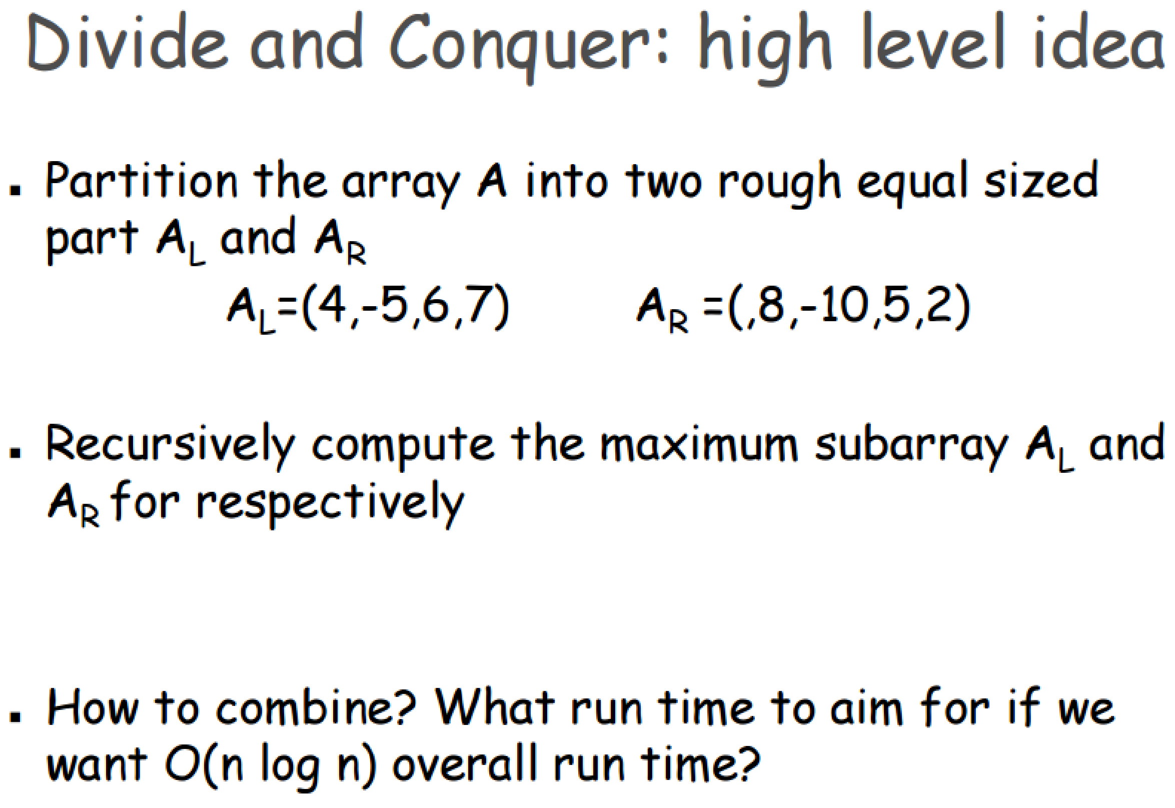
Brute Force: For all I = 1 … n For j = i+1… n. **O(n^2)**

- We can Brute Force this problem by comparing every possible combination with another, running at O(n^2)

Divide and Conquer.

- Recursively compute the Max subarray for the left and right sides respectively.
- How do we combine the cross sections? We're aiming for O (n log n) runtime.

Looking at the middle between the two Subarrays



Divide and Conquer: high level idea

- Partition the array A into two rough equal sized part A_L and A_R

  A_L=(4,-5,6,7) A_R =(,8,-10,5,2)

- Recursively compute the maximum subarray A_L and A_R for respectively

- How to combine? What run time to aim for if we want O(n log n) overall run time?

- Start at the middle point. Scanning the whole array once to find the cross solution maximum.
- Combine left side [6,7] with right side [8].
- Compare this cross solution with the best solution of the left and right side.

Example: Overall [4,-5,6,7,8,-10,5,2]

Now we split the array in half into:

- **Left [4,-5,6,7]**
- **Right side [8,-10,5,2]**

Now we find the Maximum contagious array of both left and right.

- Left side Contiguous Maximum (6,7) = 13
- Right Contiguous Maximum (8) = 8

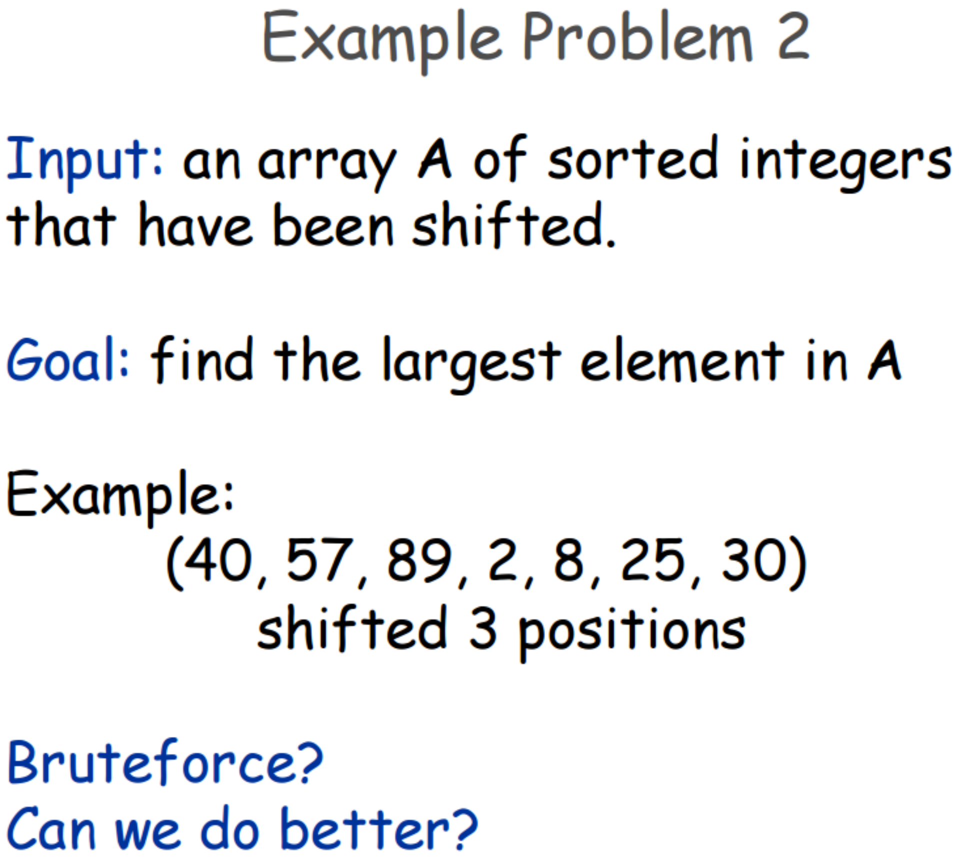
Find the Maximum of the cross section (It must require the first node on the edges (7,8)

- Cross Solution Maximum [6,7,8] = 21

Combined Best is better than the left or right side maximum.

Overall Largest Contagious Array Solution = Cross Solution = 21!

Example Problem 2: Array of Sorted Integers that have been shifted by a number of positions



Example Problem 2

Input: an array A of sorted integers that have been shifted.

Goal: find the largest element in A

Example:
(40, 57, 89, 2, 8, 25, 30)
shifted 3 positions

Bruteforce?
Can we do better?

- How do we achieve O(log n) ?
- We can use a Binary Search!

How to find the Largest Element in Shifted Array A? *Formula*

1: Let A_M be the Middle Element of Array A.

2: Let A_L be the First Element in Array A.

3: Let A_R be the Last Element in Array A.

3: If $A_M < A_L$ than The Maximum point is between $[A_L, A_M]$

4: else the Max point is Between $[A_M, A_R]$

Example: Let Array A contain elements:; (5,10,15,20,25,3,4)

1. Let **A_M** be the **Middle** element in position 4 = 20.
2. Let **A_L** be the **Left** (First position) element = 5.
3. Let **A_R** be the **Right** (Last Position) element = 4
4. Compare **if A_M < A_L**. 20 is Not Less than 4 so this ss FALSE. So we go to 5.
5. The Max Point is between **A_M – A_R** [20,***25***,3,4]

This holds to be true since the max point 25 is between the middle and the last element in the array.

Next Time:

- **Review 2nd Quiz results.**
- **Dynamic Programming (Lecture 6)**
- **Reading 1 and Reading 2 for Dynamic Programming.**
- **Work on Implementation 1 Assignment (Due Feb. 2nd).**

Week 3 Thursday Lecture Notes

~Information composed by Notetaker Scott Russell for CS 325 **DAS** student