

Divide and Conquer Method

Dr. Raghunath Reddy M



BENNETT
UNIVERSITY
TIMES OF INDIA GROUP

Dept. of Computer Science Engineering,
Bennett University, Greater Noida

January 28, 2020

The Maximum Sub-array Sum Problem

Problem Definition

Input: Let A be an array of n elements, namely, $A[1], A[2], \dots, A[n]$.

Output: Find a sub-array $A[i..j]$ i.e., $A[i], A[i+1], \dots, A[j]$ such that the sum of the sub-array is maximum.

10	-12	14	-10	12	14	10	-7	10	-15	20	-5
----	-----	----	-----	----	----	----	----	----	-----	----	----

Figure: An instance of array A .

The Maximum Subarray Sum Problem

Problem Definition

Input: Let A be an array of n elements, namely, $A[1], A[2], \dots, A[n]$.

Output: Find a sub-array $A[i..j]$ i.e., $A[i], A[i+1], \dots, A[j]$ such that the sum of the sub-array is maximum.

10	-12	14	-10	12	14	10	-7	10	-15	20	-5
----	-----	----	-----	----	----	----	----	----	-----	----	----

Figure: An instance of array A and the colored sub-array is the maximum sum sub-array for the given instance.

Question ?

How to obtain the solution in $O(n \log n)$ -time ?

Divide and Conquer Approach



Where can the optimal solution lie ?

- ❶ Can lie only in the left half $A[low..mid]$
- ❷ Can lie only in the right half $A[mid + 1..high]$
- ❸ Can cross the mid location of the array.

Special case: The optimal solution crosses the mid

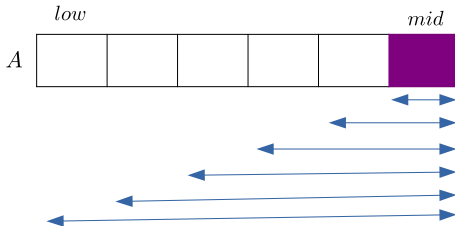
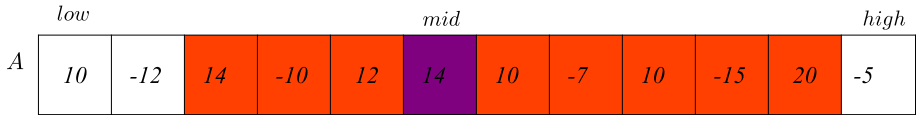
Special case: The optimal solution crosses the mid

low *mid* *high*

A

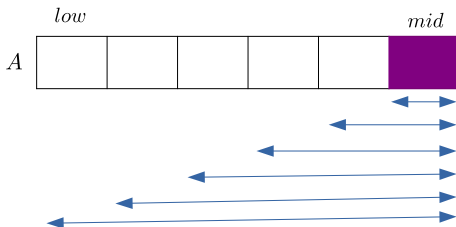
10	-12	14	-10	12	14	10	-7	10	-15	20	-5
----	-----	----	-----	----	----	----	----	----	-----	----	----

Special case: The optimal solution crosses the mid



Special case: The optimal solution crosses the mid

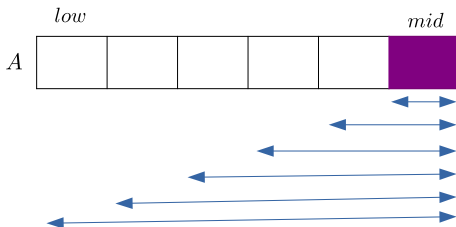
	<i>low</i>		<i>mid</i>							<i>high</i>		
<i>A</i>	10	-12	14	-10	12	14	10	-7	10	-15	20	-5



- 1: $left_sum = -\infty$
- 2: $sum = 0$
- 3: **for** $i = mid$ down to low **do**
- 4: $sum = sum + A[i]$
- 5: **if** $sum > left_sum$ **then**
- 6: $left_sum = sum$
- 7: $left_max = i$
- 8: **end if**
- 9: **end for**

Special case: The optimal solution crosses the mid

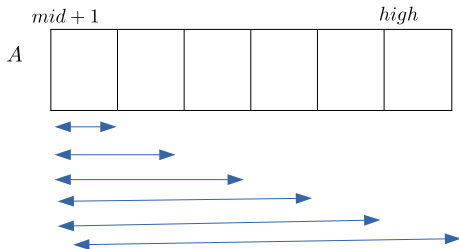
	<i>low</i>		<i>mid</i>							<i>high</i>		
<i>A</i>	10	-12	14	-10	12	14	10	-7	10	-15	20	-5



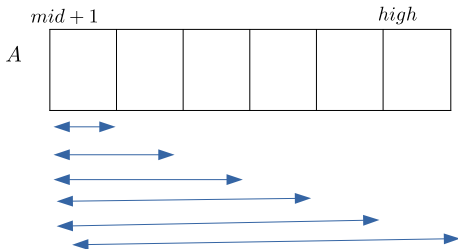
```
1: left_sum =  $-\infty$ 
2: sum = 0
3: for i = mid down to low do
4:   sum = sum + A[i]
5:   if sum > left_sum then
6:     left_sum = sum
7:     left_max = i
8:   end if
9: end for
```

- Similarly, we can extend the solution on the right side of *mid*.

Finish the code for the right of mid !!!!



Finish the code for the right of mid !!!!



```
1: right_sum =  $-\infty$ 
2: sum = 0
3: for i = mid + 1 to high do
4:     sum = sum + A[i]
5:     if sum > right_sum then
6:         right_sum = sum
7:         right_max = i
8:     end if
9: end for
```

The complete algorithm for case (3)

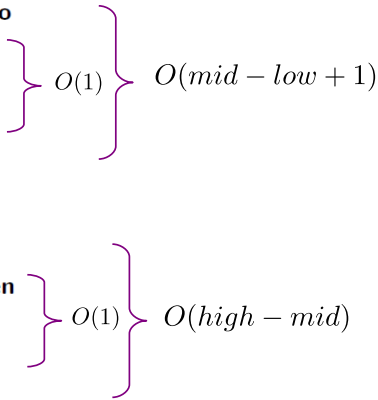
In the next slide ...

Algorithm 1 FIND_MAX_CROSSING_SUBARRAY($A, low, mid, high$)

```
1:  $left\_sum = -\infty$ 
2:  $sum = 0$ 
3: for  $i = mid$  down to  $low$  do
4:    $sum = sum + A[i]$ 
5:   if  $sum > left\_sum$  then
6:      $left\_sum = sum$ 
7:      $left\_max = i$ 
8:   end if
9: end for
10:  $right\_sum = -\infty$ 
11:  $sum = 0$ 
12: for  $i = mid + 1$  to  $high$  do
13:    $sum = sum + A[i]$ 
14:   if  $sum > right\_sum$  then
15:      $right\_sum = sum$ 
16:      $right\_max = i$ 
17:   end if
18: end for
19: return ( $left\_max, right\_max, left\_sum + right\_sum$ )
```

Algorithm 1 FIND-MAX-CROSSING-SUBARRAY($A, low, mid, high$)

```
1:  $left\_sum = -\infty$ 
2:  $sum = 0$ 
3: for  $i = mid$  down to  $low$  do
4:   if  $sum > left\_sum$  then
5:      $left\_sum = sum$ 
6:      $left\_max = i$ 
7:   end if
8: end for
9:  $right\_sum = -\infty$ 
10:  $sum = 0$ 
11: for  $i = mid + 1$  to  $high$  do
12:   if  $sum > right\_sum$  then
13:      $right\_sum = sum$ 
14:      $right\_max = i$ 
15:   end if
16: end for
17: return ( $left\_max, right\_max, left\_sum + right\_sum$ )
```

The diagram shows two purple curly braces on the right side of the algorithm. The first brace groups lines 4 through 7, with a label $O(1)$ next to it. A larger brace groups this first brace and line 8, with a label $O(mid - low + 1)$ next to it. The second brace groups lines 12 through 15, with a label $O(1)$ next to it. A larger brace groups this second brace and line 16, with a label $O(high - mid)$ next to it.

Algorithm 1 FIND-MAX-CROSSING-SUBARRAY($A, low, mid, high$)

```
1:  $left\_sum = -\infty$ 
2:  $sum = 0$ 
3: for  $i = mid$  down to  $low$  do
4:   if  $sum > left\_sum$  then
5:      $left\_sum = sum$ 
6:      $left\_max = i$ 
7:   end if
8: end for
9:  $right\_sum = -\infty$ 
10:  $sum = 0$ 
11: for  $i = mid + 1$  to  $high$  do
12:   if  $sum > right\_sum$  then
13:      $right\_sum = sum$ 
14:      $right\_max = i$ 
15:   end if
16: end for
17: return ( $left\_max, right\_max, left\_sum + right\_sum$ )
```

Complexity analysis:

- Lines 3-8: $O(1)$ per iteration, $O(mid - low + 1)$ total.
- Lines 11-16: $O(1)$ per iteration, $O(high - mid)$ total.

- Total time =

$$O(mid - low + 1) + O(high - mid) = O(high - low + 1) = O(n)$$

General case:

Algorithm 2 *FIND_MAXIMUM_SUBARRAY*(*A*, *low*, *high*)

```
1: if high == low then
2:   return (low, high, A[low]) //base case
3: else mid =  $\lfloor (\textit{low} + \textit{high})/2 \rfloor$ 
4:   (llow, lhigh, lsum) = FIND_MAXIMUM_SUBARRAY(A, low, mid)
5:
6:   (rlow, rhigh, rsum) = FIND_MAXIMUM_SUBARRAY(A, mid + 1, high)
7:
8:   (clow, chigh, csum) = FIND_MAX_CROSSING_SUBARRAY(A, low, mid, high)
9:
10:  if lsum ≥ rsum and lsum ≥ csum then return (llow, lhigh, lsum)
11:
12:  else if rsum ≥ lsum and rsum ≥ csum then return (rlow, rhigh, rsum)
13:
14:  else return (clow, chigh, csum)
15:
16:  end if
17: end if
```

General case:

- Total time $T(n) = 2T(n/2) + O(n) = O(n \log n)$

The maximum sub-array Problem

Problem Definition

Input: Let A be an array of n elements, namely, $A[1], A[2], \dots, A[n]$.

Output: Find a sub-array two indices i and j of A such that (i) $i < j$ and (ii) $A[j] - A[i]$ is the maximum.

85	105	102	86	63	81	101	94	106	101	79	94
----	-----	-----	----	----	----	-----	----	-----	-----	----	----

Figure: An instance of the maximum sub-array problem.

Reduction to The maximum sub-array sum problem

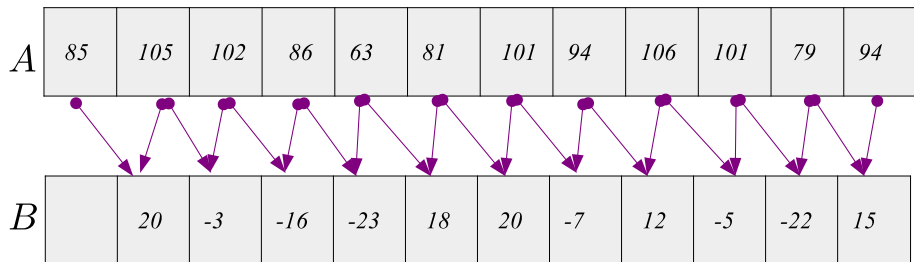


Figure: $B[i] = A[i] - A[i - 1]$ for all $i \geq 2$ and $B[1] = \text{NULL}$.

- A is an instance of the maximum sub-array problem.
- B is an instance of the maximum sub-array sum problem.

Relation between the solutions of A and B

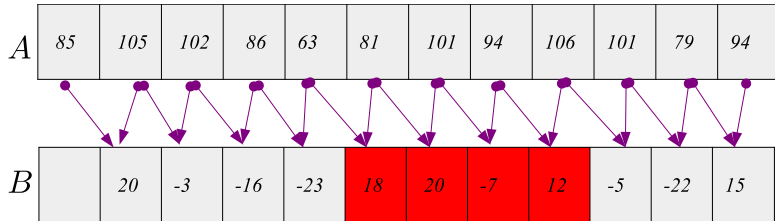


Figure: The red colored sub-array of B is the maximum sum sub-array.

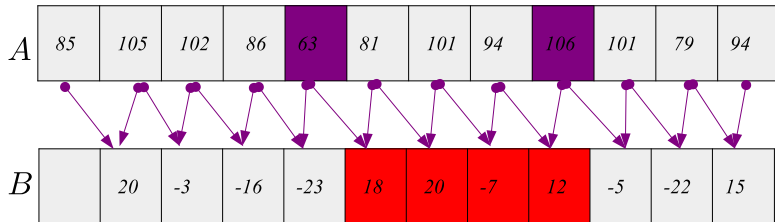


Figure: The colored cells of A denote the solution to the problem.

Finding the Closest Pair of Points in the Plane

Closest pair of points

Problem Definition

Input: Let $P = \{p_1, p_2, \dots, p_n\}$ be a set of points in the plane where each $p_i = (x_i, y_i)$ for real numbers x_i and y_i .

Output: Find a pair of points p_i and p_j such that the Euclidean distance between p_i and p_j , $d(p_i, p_j)$ is minimum.

Brute-force algorithm

Check all possible nC_2 pairs and pick the best one.

Time : $O(n^2)$

Question

Can we do it in better time, like $O(n \log n)$ -time ?