DESIGN AND ANALYSIS OF ALGORITHMS

CS 4120/5120

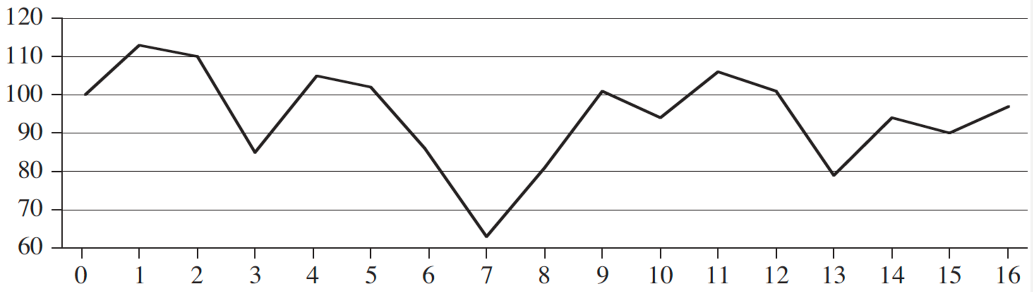
MAXIMUM-SUBARRAY PROBLEM

AGENDA

- Maximum-subarray problem definition
- Divide-and-conquer algorithm
- Time complexity analysis

THE STOCK TRADING STORY

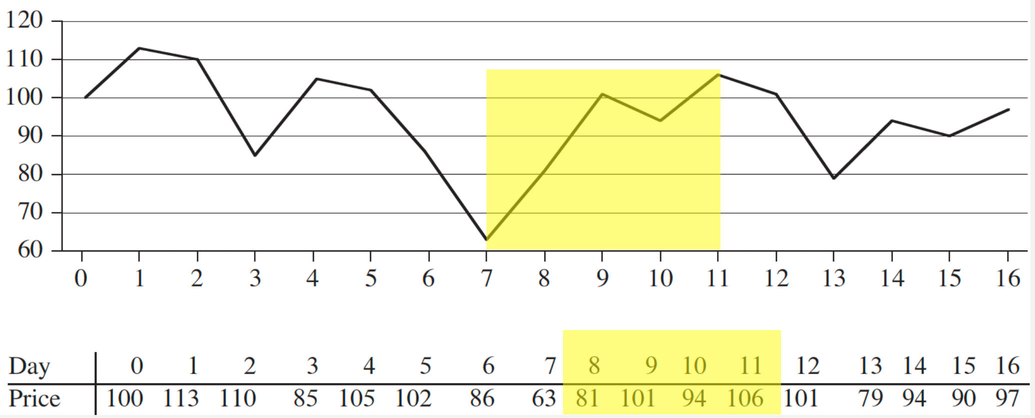
- Suppose that you can learn what the price of the stock will be in the future 17 days.
  - The price of the stock over the17-day period.
  - You can buy or sell after the close of trading for the day.
  - Using this chart, you can make two trading activities: one buy and one sell.
- What is the best time to buy and the best time to sell?
  - What is the profit?



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97

THE STOCK TRADING STORY
MAKING THE MOST MONEY

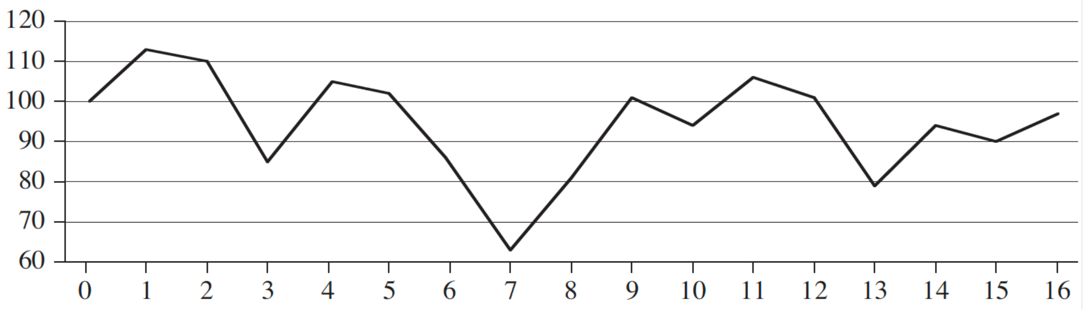
- Suppose that you can learn what the price of the stock will be in the future 17 days.
  - The price of the stock over the17-day period.
  - You can buy or sell after the close of trading for the day.
  - Using this chart, you can make two trading activities: one buy and one sell.
- The maximum profit can be obtained by buying after the market closes on day _______ and sell after the market closes on day _______.
- Share your thoughts?



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97

THE STOCK TRADING STORY
MODEL THE PROBLEM

- Transformation
  - Compute the change of price of two consecutive days.
  - Store the changes in an array.
  - Find the longest subarray whose values have the greatest sum.



	Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97
$A \Rightarrow$	Change		13	−3	−25	20	−3	−16	−23	18	20	−7	12	−5	−22	15	−4	7

PLAY WITH THE MODEL

- Given the following input instance.

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	19	-20	8	4	-4	1	7	-15	8	6	-4	6	-1	2	-6	3

- Find the longest subarray whose values have the greatest sum.

MAXIMUM-SUBARRAY PROBLEM

PROBLEM DEFINITION

- **Input**
  - Array $A[1..n]$ containing both **positive** and **negative** numbers.
- **Output**
  - A contiguous subarray $A[i..j]$ of $A[1..n], 1 \leq i \leq j \leq n$, such that the sum of values in $A[i..j]$ is *the largest* of all *contiguous subarrays* of $A[1..n]$.

MAXIMUM-SUBARRAY PROBLEM
BRUTE-FORCE

- Goal
  - A contiguous subarray $A[i..j]$ of $A[1..n]$, $1 \leq i \leq j \leq n$, such that the sum of values in $A[i..j]$ is *the largest* of all *contiguous subarrays* of $A[1..n]$.
- Brute-force solution
  - **Step 1**: List all contiguous subarrays.
    - There are $\underline{\binom{n}{2} \text{ or } C_n^2}$ different subarrays.

MAXIMUM-SUBARRAY PROBLEM
BRUTE-FORCE

- Goal
  - A contiguous subarray $A[i..j]$ of $A[1..n]$, $1 \le i \le j \le n$, such that the sum of values in $A[i..j]$ is *the largest* of all *contiguous subarrays* of $A[1..n]$.
- Brute-force solution
  - **Step 2**: Calculate the sum of each subarray $A[i..j]$.
    - For each subarray, the cost of computing the sum is the cost of traversing $A[i..j]$.
    - Therefore, the cost of computing the sum of each subarray is $\boldsymbol{j - i}$, where $i \neq j$ and $i, j \in [1, n]$.

MAXIMUM-SUBARRAY PROBLEM
BRUTE-FORCE

- Goal
  - A contiguous subarray $A[i..j]$ of $A[1..n]$, $1 \leq i \leq j \leq n$, such that the sum of values in $A[i..j]$ is *the largest* of all *contiguous subarrays* of $A[1..n]$.
- Brute-force solution
  - **Step 3**: Find the max of all sums.
    - The cost of finding the max of sums is $\underline{\binom{n}{2} - 1}$.

MAXIMUM-SUBARRAY PROBLEM
BRUTE-FORCE

- Goal
  - A contiguous subarray $A[i..j]$ of $A[1..n], 1 \leq i \leq j \leq n$, such that the sum of values in $A[i..j]$ is *the largest* of all *contiguous subarrays* of $A[1..n]$.
- Brute-force solution
  - The time complexity of brute-force solution is $T(n) = \boldsymbol{\Omega(n^2)}$.
- Better solution?

MAXIMUM-SUBARRAY PROBLEM
DIVIDE AND CONQUER

- Apply the **three steps**
  - **Divide** the problem into a number of subproblems that are smaller instances of the same problem.
  - **Conquer** the subproblems by solving them recursively.
    - If the subproblem sizes are small enough, just solve the subproblems in a straightforward manner.
  - **Combine** the solutions to the subproblems into the solution to the original problem.

MAXIMUM-SUBARRAY PROBLEM
DIVIDE

- **Divide** the original array $A[1..n]$ into subarrays $A[1..mid]$ and $A[mid+1..n]$.
  - $mid = \lfloor (1+n)/2 \rfloor$
  - Deal with $A[1..mid]$ and $A[mid+1..n]$ individually by dividing them into smaller subproblems.

$\mathbf{1}$
↓

$\boldsymbol{mid}$
↓

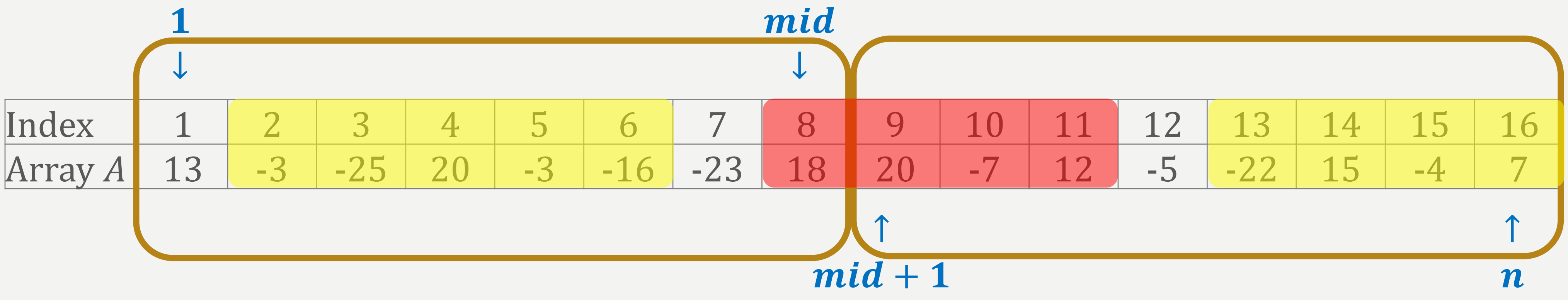
Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Array A	13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

↑
$\boldsymbol{mid+1}$

↑
$\boldsymbol{n}$

MAXIMUM-SUBARRAY PROBLEM
CONQUER

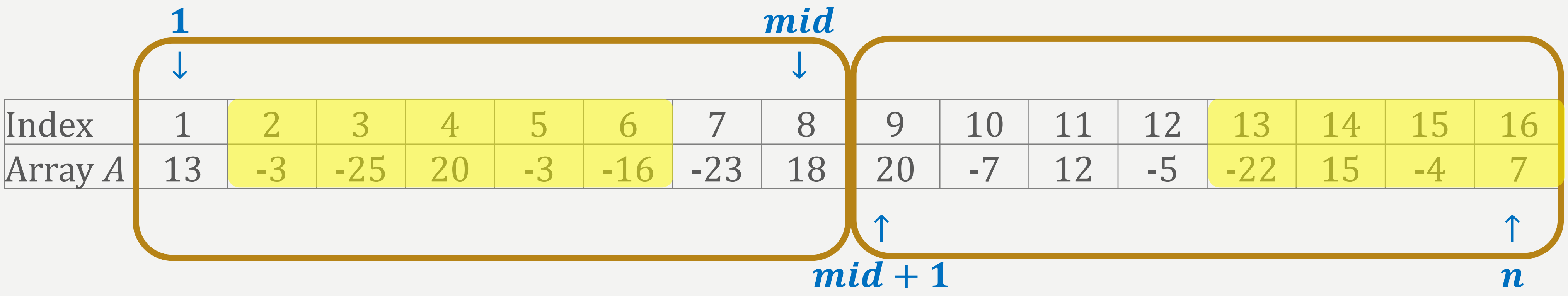
- **Conquer** the subproblems arisen from the **divide** step.
  - Two scenarios emerge.
    - **Case 1**: The maximum subarray lies entirely in the left half $A[1..mid]$ or entirely in the right half $A[mid+1..n]$.
    - **Case 2**:The maximum subarray $A[i..j]$ happens to **cross the midpoint**.



Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Array A	13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

MAXIMUM-SUBARRAY PROBLEM
CONQUER CASE 1

- The subproblems are **the same as the original problem**.
- Generally, the problem can be described as finding the maximum subarray of $A[low\,..\,high]$.
- In this case, the indices of maximum subarray $A[i\,..\,j]$ will always satisfy $\boldsymbol{low \le i < j \le high}$.
- Solving case 1 is trivial.



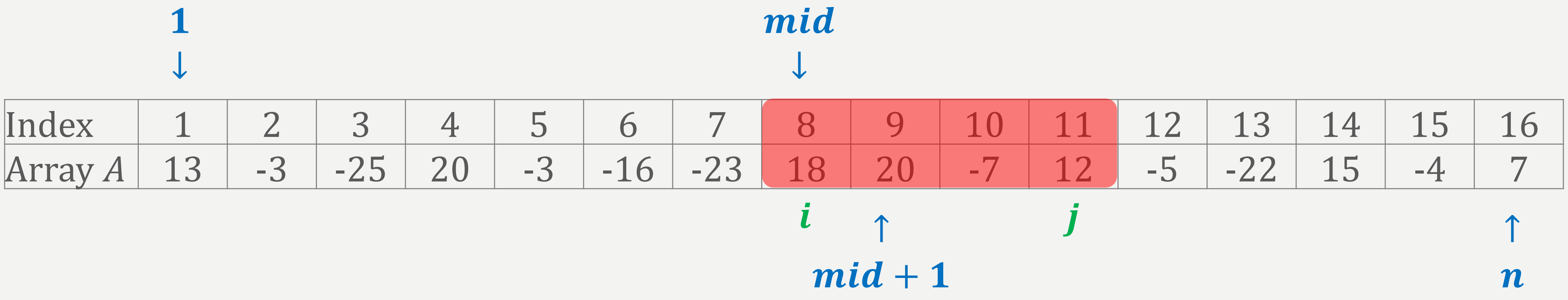
$\mathbf{1}$ ↓ $\quad$ $\boldsymbol{mid}$ ↓

Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Array A	13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

↑ $\boldsymbol{mid + 1}$ $\quad$ ↑ $\boldsymbol{n}$

MAXIMUM-SUBARRAY PROBLEM
CONQUER CASE 2

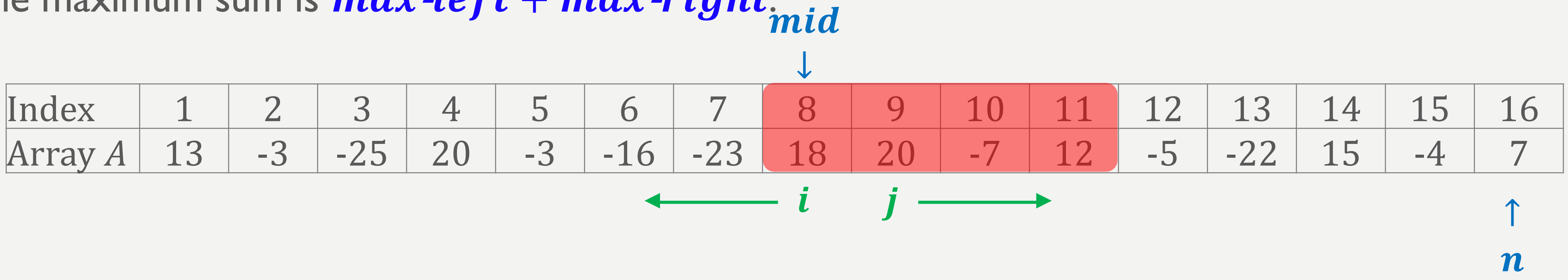
- The maximum subarray $A[\boldsymbol{i}..\boldsymbol{j}]$ happens to **cross the midpoint**.
- In this case, the indices of maximum subarray $A[i..j]$ will be $\boldsymbol{low \le i \le mid < j \le high}$, which is **NOT** exactly the same as the original problem.
- In other words, solution consists of two arrays: $\underline{\boldsymbol{A[i..mid]}}$ and $\underline{\boldsymbol{A[mid+1..j]}}$.



Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Array A	13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

MAXIMUM-SUBARRAY PROBLEM
CONQUER CASE 2 (CONT'D)

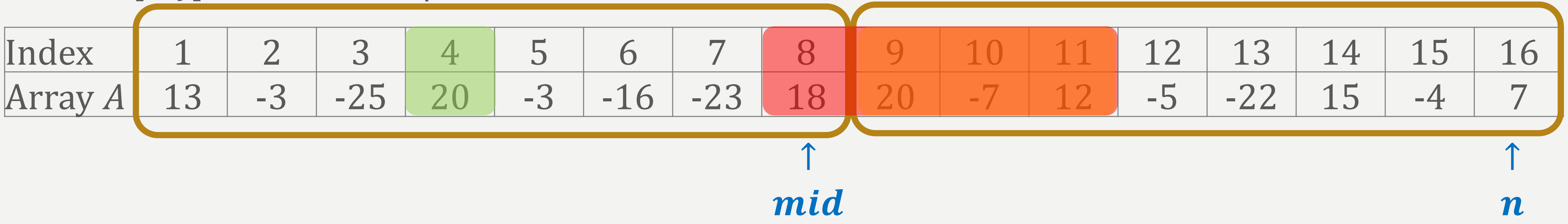
- The solution $A[i..j]$ **crossing the midpoint** means $A[mid]$ is included in $A[i..j]$.
- Start at $A[mid]$.
  - Move i to the left side to find subarray $A[i..mid]$ that has the greatest sum, $max\text{-}left$.
  - Move j to the right side to find subarray $A[mid+1..j]$ that has the greatest sum, $max\text{-}right$.
- The maximum sum is $max\text{-}left + max\text{-}right$.



Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Array A	13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

MAXIMUM-SUBARRAY PROBLEM
COMBINE

- **Combine** the solutions obtained from case 1 and case 2. Pick the **greatest sum**.
  - Case 1
    - $A[i..j]$ exists entirely in the left half. The maximum sum is $\boldsymbol{left\text{-}sum = 20}$.
    - $A[i..j]$ exists entirely in the right half. The maximum sum is $\boldsymbol{right\text{-}sum = 20 \ - 7 + 12 = 25}$.
  - Case 2
    - $A[i..j]$ crosses the midpoint. The maximum sum is $\boldsymbol{cross\text{-}sum = 18 + 20 - 7 + 12 = 43}$.



Index	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Array A	13	-3	-25	20	-3	-16	-23	18	20	-7	12	-5	-22	15	-4	7

MAXIMUM-SUBARRAY PROBLEM
THE ALGORITHM

- The $FIND\text{-}MAXIMUM\text{-}SUBARRAY$ algorithm
    - Use $\Theta(1)$ to denote a constant time cost.
    - Use $f(high - low + 1)$ to denote the cost of line 6
    - Use probability P_i to denote the probability of conditions being true at line 7 and 9.

$FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, high)$		Cost	Time
1	**if** $high == low$	$\Theta(1)$	base
2	**return** $(low, high, A[low])$ // base case	$\Theta(1)$	base
3	**else** $mid = \lfloor (low + high)/2 \rfloor$	$\Theta(1)$	1
4	$(left\text{-}low, left\text{-}high, left\text{-}sum) =$ $FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, mid)$	$T(mid - low + 1)$	1
5	$(right\text{-}low, right\text{-}high, right\text{-}sum) =$ $FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, mid + 1, high)$	$T(high - mid)$	1
6	$(cross\text{-}low, cross\text{-}high, cross\text{-}sum) =$ $FIND\text{-}MAXIMUM\text{-}CROSSING\text{-}SUBARRAY(A, low, mid, high)$	$f(high - low + 1)$	1
7	**if** $left\text{-}sum \geq right\text{-}sum$ **and** $left\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	1
8	**return** $(left\text{-}low, left\text{-}high, left\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_1}$
9	**elseif** $right\text{-}sum \geq left\text{-}sum$ **and** $right\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	$\boldsymbol{P_1}$
10	**return** $(right\text{-}low, right\text{-}high, right\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_2}$
11	**else return** $(cross\text{-}low, cross\text{-}high, cross\text{-}sum)$	$\Theta(1)$	$\boldsymbol{1 - P_1 - P_2}$

MAXIMUM-SUBARRAY PROBLEM
THE RUNNING TIME

- Derive the running time function of $T(high - low + 1)$ $=$

$FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, high)$		Cost	Time
1	**if** $high == low$	$\Theta(1)$	base
2	**return** $(low, high, A[low])$ // base case	$\Theta(1)$	base
3	**else** $mid = \lfloor (low + high)/2 \rfloor$	$\Theta(1)$	1
4	$(left\text{-}low, left\text{-}high, left\text{-}sum) =$ $FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, mid)$	$T(mid - low + 1)$	1
5	$(right\text{-}low, right\text{-}high, right\text{-}sum) =$ $FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, mid + 1, high)$	$T(high - mid)$	1
6	$(cross\text{-}low, cross\text{-}high, cross\text{-}sum) =$ $FIND\text{-}MAXIMUM\text{-}CROSSING\text{-}SUBARRAY(A, low, mid, high)$	$f(high - low + 1)$	1
7	**if** $left\text{-}sum \geq right\text{-}sum$ **and** $left\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	1
8	**return** $(left\text{-}low, left\text{-}high, left\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_1}$
9	**elseif** $right\text{-}sum \geq left\text{-}sum$ **and** $right\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	$\boldsymbol{P_1}$
10	**return** $(right\text{-}low, right\text{-}high, right\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_2}$
11	**else return** $(cross\text{-}low, cross\text{-}high, cross\text{-}sum)$	$\Theta(1)$	$\boldsymbol{1 - P_1 - P_2}$

MAXIMUM-SUBARRAY PROBLEM
THE RUNNING TIME

- Let $low = \mathbf{1}$ and let $high = \boldsymbol{n}$.
  $\boldsymbol{n} =$ ________________.
  $T(high - low + 1)$
  $= \boldsymbol{\Theta(1)} +$
  $T(mid - low + 1) +$
  $T(high - mid) +$
  $f(high - low + 1)$

$FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, high)$		Cost	Time
1	**if** $high == low$	$\Theta(1)$	base
2	**return** $(low, high, A[low])$ // base case	$\Theta(1)$	base
3	**else** $mid = \lfloor (low + high)/2 \rfloor$	$\Theta(1)$	1
4	$(left\text{-}low, left\text{-}high, left\text{-}sum) = FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, mid)$	$T(mid - low + 1)$	1
5	$(right\text{-}low, right\text{-}high, right\text{-}sum) = FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, mid + 1, high)$	$T(high - mid)$	1
6	$(cross\text{-}low, cross\text{-}high, cross\text{-}sum) = FIND\text{-}MAXIMUM\text{-}CROSSING\text{-}SUBARRAY(A, low, mid, high)$	$f(high - low + 1)$	1
7	**if** $left\text{-}sum \geq right\text{-}sum$ **and** $left\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	1
8	**return** $(left\text{-}low, left\text{-}high, left\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_1}$
9	**elseif** $right\text{-}sum \geq left\text{-}sum$ **and** $right\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	$\boldsymbol{P_1}$
10	**return** $(right\text{-}low, right\text{-}high, right\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_2}$
11	**else return** $(cross\text{-}low, cross\text{-}high, cross\text{-}sum)$	$\Theta(1)$	$\boldsymbol{1 - P_1 - P_2}$

MAXIMUM-SUBARRAY PROBLEM
THE RUNNING TIME

- Let $low = \mathbf{1}$ and let $high = \boldsymbol{n}$. $mid =$ ________________.
- mid

  $=$ $(n + 1$ is even$)$

  $(n + 1$ is odd$)$

$FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, high)$		Cost	Time
1	**if** $high == low$	$\Theta(1)$	base
2	**return** $(low, high, A[low])$ // base case	$\Theta(1)$	base
3	**else** $mid = \lfloor (low + high)/2 \rfloor$	$\Theta(1)$	1
4	$(left\text{-}low, left\text{-}high, left\text{-}sum) = FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, mid)$	$T(mid - low + 1)$	1
5	$(right\text{-}low, right\text{-}high, right\text{-}sum) = FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, mid + 1, high)$	$T(high - mid)$	1
6	$(cross\text{-}low, cross\text{-}high, cross\text{-}sum) = FIND\text{-}MAXIMUM\text{-}CROSSING\text{-}SUBARRAY(A, low, mid, high)$	$f(high - low + 1)$	1
7	**if** $left\text{-}sum \geq right\text{-}sum$ **and** $left\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	1
8	**return** $(left\text{-}low, left\text{-}high, left\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_1}$
9	**elseif** $right\text{-}sum \geq left\text{-}sum$ **and** $right\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	$\boldsymbol{P_1}$
10	**return** $(right\text{-}low, right\text{-}high, right\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_2}$
11	**else return** $(cross\text{-}low, cross\text{-}high, cross\text{-}sum)$	$\Theta(1)$	$\boldsymbol{1 - P_1 - P_2}$

MAXIMUM-SUBARRAY PROBLEM
THE RUNNING TIME

- The running time function
  $T(n)$
  $= \mathbf{\Theta(1)} + T\left(\frac{n+1}{2}\right) + T\left(\frac{n-1}{2}\right) + f(\boldsymbol{n})$ ($n+1$ is even)
  $= \mathbf{\Theta(1)} + \mathbf{2}T\left(\frac{\boldsymbol{n}}{\mathbf{2}}\right) + f(\boldsymbol{n})$ ($n+1$ is odd)
- Tolerate sloppiness
  $T(n) = \mathbf{2}T\left(\frac{n}{2}\right) + f(n)$

$FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, high)$		Cost	Time
1	**if** $high == low$	$\Theta(1)$	base
2	**return** $(low, high, A[low])$ // base case	$\Theta(1)$	base
3	**else** $mid = \lfloor (low + high)/2 \rfloor$	$\Theta(1)$	1
4	$(left\text{-}low, left\text{-}high, left\text{-}sum) = FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, low, mid)$	$T(mid - low + 1)$	1
5	$(right\text{-}low, right\text{-}high, right\text{-}sum) = FIND\text{-}MAXIMUM\text{-}SUBARRAY(A, mid + 1, high)$	$T(high - mid)$	1
6	$(cross\text{-}low, cross\text{-}high, cross\text{-}sum) = FIND\text{-}MAXIMUM\text{-}CROSSING\text{-}SUBARRAY(A, low, mid, high)$	$f(high - low + 1)$	1
7	**if** $left\text{-}sum \geq right\text{-}sum$ **and** $left\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	1
8	**return** $(left\text{-}low, left\text{-}high, left\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_1}$
9	**elseif** $right\text{-}sum \geq left\text{-}sum$ **and** $right\text{-}sum \geq cross\text{-}sum$	$\Theta(1)$	$\boldsymbol{P_1}$
10	**return** $(right\text{-}low, right\text{-}high, right\text{-}sum)$	$\Theta(1)$	$\boldsymbol{P_2}$
11	**else return** $(cross\text{-}low, cross\text{-}high, cross\text{-}sum)$	$\Theta(1)$	$\boldsymbol{1 - P_1 - P_2}$

FIND-MAX-CROSSING-SUBARRAY
RUNNING TIME

- Complete the **Cost** and **Time** columns.
  - Consider ONLY the worst-case scenario

$\boldsymbol{FIND\text{-}MAX\text{-}CROSSING\text{-}SUBARRAY}(A, low, mid, high)$		Cost	Time (Worst-case)
1	$left\text{-}sum = -\infty$	$\Theta(1)$	1
2	$sum = 0$	$\Theta(1)$	
3	**for** $i = mid$ **downto** low	$\Theta(1)$	
4	$sum = sum + A[i]$	$\Theta(1)$	
5	**if** $sum > left\text{-}sum$	$\Theta(1)$	
6	$left\text{-}sum = sum$	$\Theta(1)$	
7	$max\text{-}left = i$	$\Theta(1)$	
8	$right\text{-}sum = -\infty$	$\Theta(1)$	
9	$sum = 0$	$\Theta(1)$	
10	**for** $j = mid + 1$ **to** $high$	$\Theta(1)$	
11	$sum = sum + A[j]$	$\Theta(1)$	
12	if $sum > right\text{-}sum$	$\Theta(1)$	
13	$right\text{-}sum = sum$	$\Theta(1)$	
14	$max\text{-}right = j$	$\Theta(1)$	
15	**return** $(max\text{-}left, max\text{-}right, left\text{-}sum + right\text{-}sum)$	$\Theta(1)$	

FIND-MAX-CROSSING-SUBARRAY RUNNING TIME

- Complete the **Cost** and **Time** columns.
  - Consider ONLY the worst-case scenario

$\boldsymbol{FIND\text{-}MAX\text{-}CROSSING\text{-}SUBARRAY}(A, low, mid, high)$		Cost	Time (Worst-case)
1	$left\text{-}sum = -\infty$	$\Theta(1)$	1
2	$sum = 0$	$\Theta(1)$	1
3	**for** $i = mid$ **downto** low	$\Theta(1)$	$(mid - low + \mathbf{2})$
4	$sum = sum + A[i]$	$\Theta(1)$	$(mid - low + 1)$
5	**if** $sum > left\text{-}sum$	$\Theta(1)$	$(mid - low + 1)$
6	$left\text{-}sum = sum$	$\Theta(1)$	$(mid - low + 1)$
7	$max\text{-}left = i$	$\Theta(1)$	$(mid - low + 1)$
8	$right\text{-}sum = -\infty$	$\Theta(1)$	1
9	$sum = 0$	$\Theta(1)$	1
10	**for** $j = mid + 1$ **to** $high$	$\Theta(1)$	$(high - mid + 1)$
11	$sum = sum + A[j]$	$\Theta(1)$	$(high - mid)$
12	**if** $sum > right\text{-}sum$	$\Theta(1)$	$(high - mid)$
13	$right\text{-}sum = sum$	$\Theta(1)$	$(high - mid)$
14	$max\text{-}right = j$	$\Theta(1)$	$(high - mid)$
15	**return** $(max\text{-}left, max\text{-}right, left\text{-}sum + right\text{-}sum)$	$\Theta(1)$	1

FIND-MAX-CROSSING-SUBARRAY
RUNNING TIME

- Running time $f(high - low + 1)$ $=$

$\boldsymbol{FIND\text{-}MAX\text{-}CROSSING\text{-}SUBARRAY}(A, low, mid, high)$		Cost	Time (Worst-case)
1	$left\text{-}sum = -\infty$	$\Theta(1)$	1
2	$sum = 0$	$\Theta(1)$	1
3	**for** $i = mid$ **downto** low	$\Theta(1)$	$(mid - low + \mathbf{2})$
4	$sum = sum + A[i]$	$\Theta(1)$	$(mid - low + 1)$
5	**if** $sum > left\text{-}sum$	$\Theta(1)$	$(mid - low + 1)$
6	$left\text{-}sum = sum$	$\Theta(1)$	$(mid - low + 1)$
7	$max\text{-}left = i$	$\Theta(1)$	$(mid - low + 1)$
8	$right\text{-}sum = -\infty$	$\Theta(1)$	1
9	$sum = 0$	$\Theta(1)$	1
10	**for** $j = mid + 1$ **to** $high$	$\Theta(1)$	$(high - mid + 1)$
11	$sum = sum + A[j]$	$\Theta(1)$	$(high - mid)$
12	**if** $sum > right\text{-}sum$	$\Theta(1)$	$(high - mid)$
13	$right\text{-}sum = sum$	$\Theta(1)$	$(high - mid)$
14	$max\text{-}right = j$	$\Theta(1)$	$(high - mid)$
15	**return** $(max\text{-}left, max\text{-}right, left\text{-}sum + right\text{-}sum)$	$\Theta(1)$	1

FIND-MAX-CROSSING-SUBARRAY
RUNNING TIME

- Let $low = 1, high = n.$
- $f(high - low + 1)$
  $= f(n)$
  $=$

	$\boldsymbol{FIND\text{-}MAX\text{-}CROSSING\text{-}SUBARRAY}(A, low, mid, high)$	Cost	Time (Worst-case)
1	$left\text{-}sum = -\infty$	$\Theta(1)$	1
2	$sum = 0$	$\Theta(1)$	1
3	**for** $i = mid$ **downto** low	$\Theta(1)$	$(mid - low + \mathbf{2})$
4	$sum = sum + A[i]$	$\Theta(1)$	$(mid - low + 1)$
5	**if** $sum > left\text{-}sum$	$\Theta(1)$	$(mid - low + 1)$
6	$left\text{-}sum = sum$	$\Theta(1)$	$(mid - low + 1)$
7	$max\text{-}left = i$	$\Theta(1)$	$(mid - low + 1)$
8	$right\text{-}sum = -\infty$	$\Theta(1)$	1
9	$sum = 0$	$\Theta(1)$	1
10	**for** $j = mid + 1$ **to** $high$	$\Theta(1)$	$(high - mid + 1)$
11	$sum = sum + A[j]$	$\Theta(1)$	$(high - mid)$
12	**if** $sum > right\text{-}sum$	$\Theta(1)$	$(high - mid)$
13	$right\text{-}sum = sum$	$\Theta(1)$	$(high - mid)$
14	$max\text{-}right = j$	$\Theta(1)$	$(high - mid)$
15	**return** $(max\text{-}left, max\text{-}right, left\text{-}sum + right\text{-}sum)$	$\Theta(1)$	1

MAXIMUM-SUBARRAY PROBLEM
TIME COMPLEXITY

- Combine the recursive function $T(n)$ and $f(n)$.
- The recursive running time of the $FIND\text{-}MAXIMUM\text{-}SUBARRAY$ algorithm is

$$T(n) = \begin{cases} \Theta(1) & \text{if } n = 1 \\ 2T(n/2) + \Theta(n) & \text{if } n > 1 \end{cases}$$

NEXT UP
STRASSEN’S ALGORITHM

REFERENCE

- https://www.vectorstock.com/royalty-free-vector/stacks-of-coins-and-money-bag-vector-1099019
- The stock price chart is a screenshot from the textbook.