Divide-and-Conquer

Kuan-Yu Chen (陳冠宇)

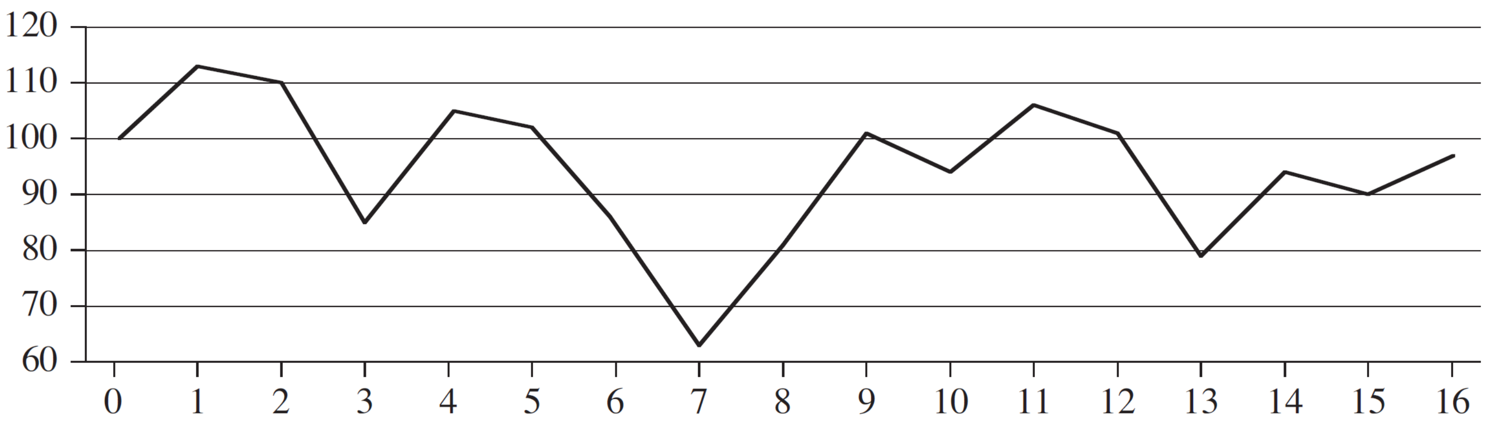
2019/03/19 @ TR-310-1, NTUST

Review

- There are three methods for obtaining asymptotic "Θ" or "O" bounds on the solution
  - Substitution method
    1. Guess the form of the solution
    2. Use mathematical induction to find the constants and show that the solution works
  - Recursion-tree method
    - Each node represents the cost of a single subproblem
    - Sum all of the costs to determine the total cost of the recursion
  - Master method
    - The master method provides a "cookbook" method for solving recurrences of the form $T(n) = aT\left(\frac{n}{b}\right) + f(n)$

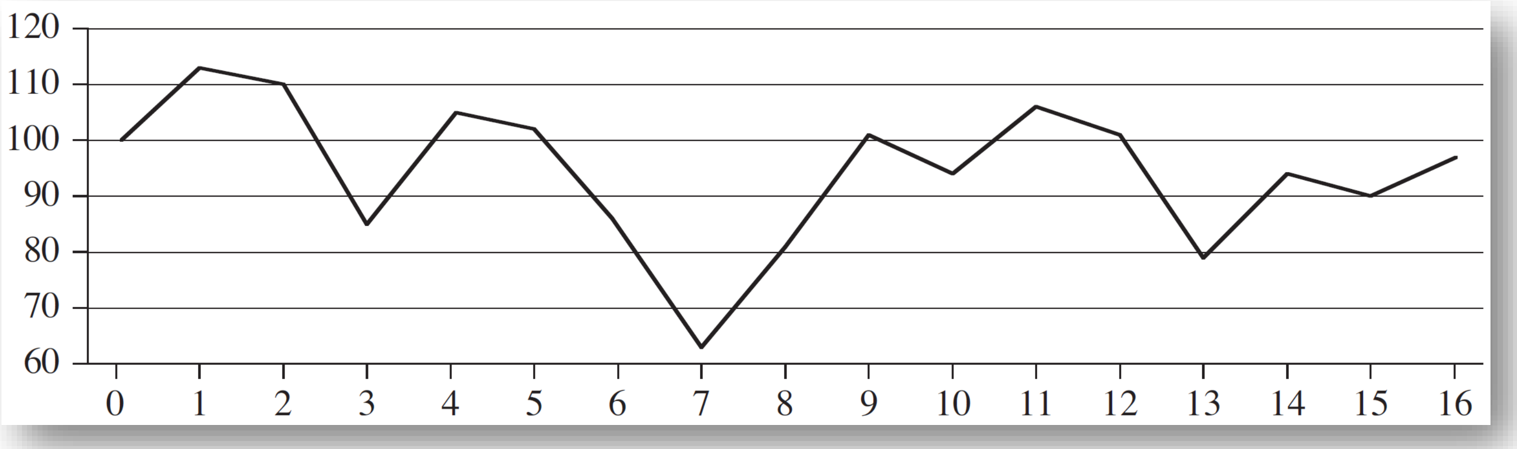
Maximum-subarray Problem.

- The stock price of $STOCK$ is rather volatile
  - You are allowed to buy one unit of stock only one time and then sell it at a later date
  - To compensate for this restriction, you are allowed to learn what the price of the stock will be in the future
    - The price of the stock over a 17-day period



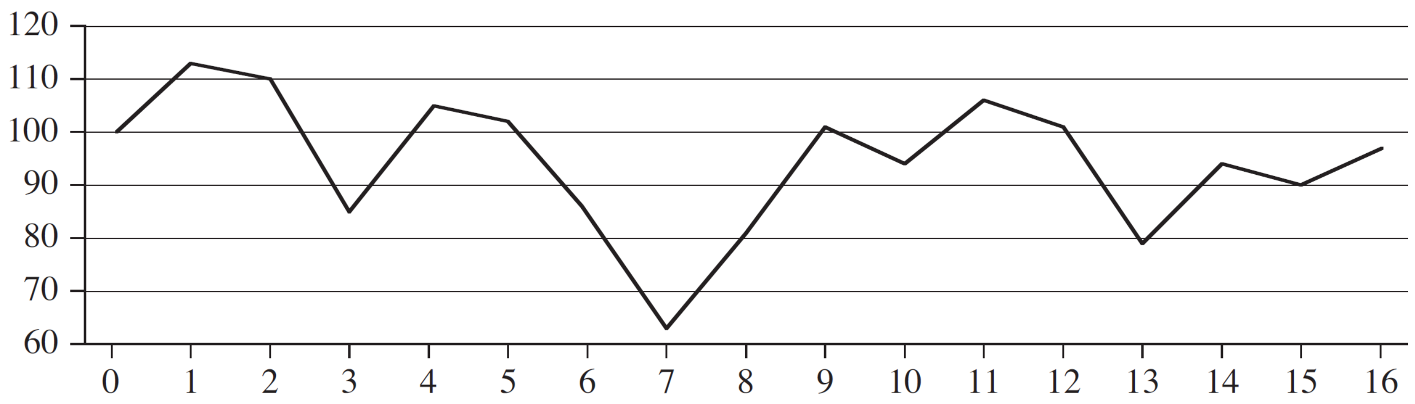
  - Your goal is to maximize your profit
    - Buy at the lowest possible price and later on sell at the highest possible price

Maximum-subarray Problem..



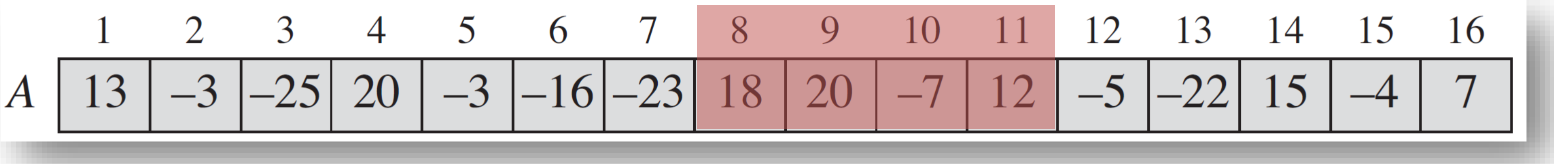
- **A brute-force solution**
  - Just try every possible pair of buy and sell dates in which the buy date precedes the sell date
  - A period of n days has $\binom{n}{2}$ such pairs of dates
  - Since $\binom{n}{2} = \frac{n^2-n}{2}$, the approach would take $\Omega(n^2)$ time

Maximum-subarray Problem…



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
Change		13	−3	−25	20	−3	−16	−23	18	20	−7	12	−5	−22	15	−4	7

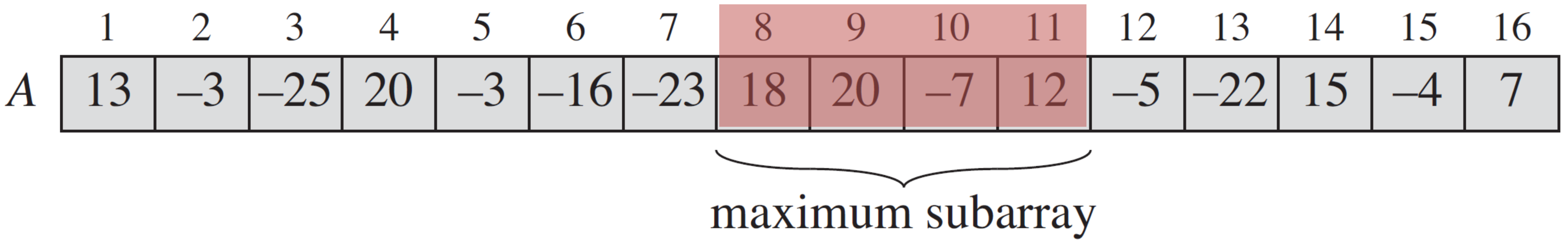
- **A transformation approach**
  - Let us instead consider the daily change in price
  - If we treat this row as an array *A*, we now want to find the nonempty, contiguous subarray of *A* whose values have the largest sum



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

Maximum-subarray Problem….

- We call this contiguous subarray the ***maximum subarray***



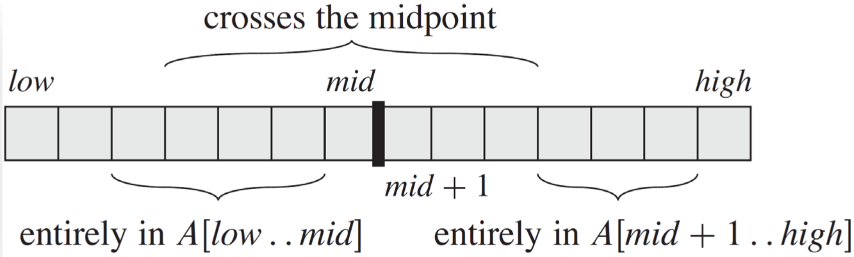
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
A	13	–3	–25	20	–3	–16	–23	18	20	–7	12	–5	–22	15	–4	7

maximum subarray

- The maximum subarray of A is $A[8 \cdots 11]$, with the sum 43
- You would want to buy the stock just before day 8 (that is, after day 7) and sell it after day 11
- This transformation does not help
  - We still need to check $\binom{n-1}{2}$ subarrays for a period of n days
  - It takes $\Theta(n^2)$ time

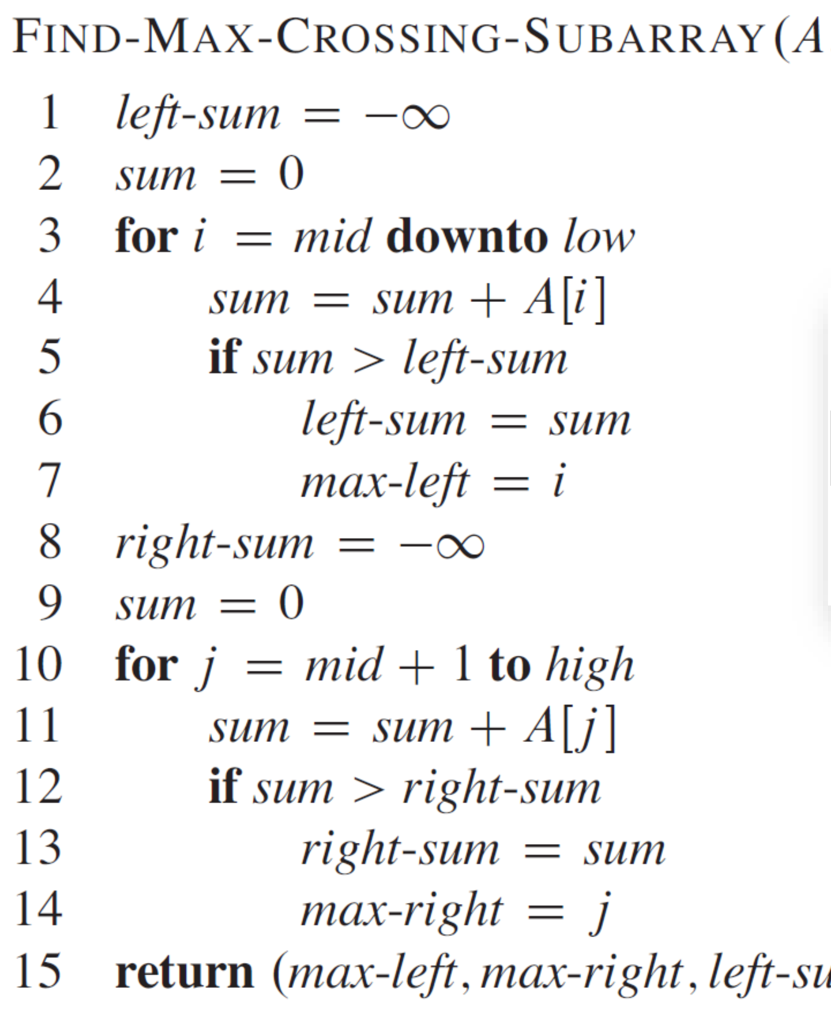
Maximum-subarray Problem…..

- **A solution using divide-and-conquer**
  - Suppose we want to find a maximum subarray of the subarray $A[low \cdots high]$
  - Divide-and-conquer suggests that we divide the subarray into two subarrays $A[low \cdots mid]$ and $A[mid + 1 \cdots high]$
  - Any contiguous subarray $A[i \cdots j]$of $A[low \cdots high]$ must lie in exactly one of the following places
    - Entirely in the subarray $A[low \cdots mid]$, so that $low \le i \le j \le mid$
    - Entirely in the subarray $A[mid + 1 \cdots high]$, so that $mid < i \le j \le high$
    - Crossing the midpoint so that $low \le i \le mid < j \le high$

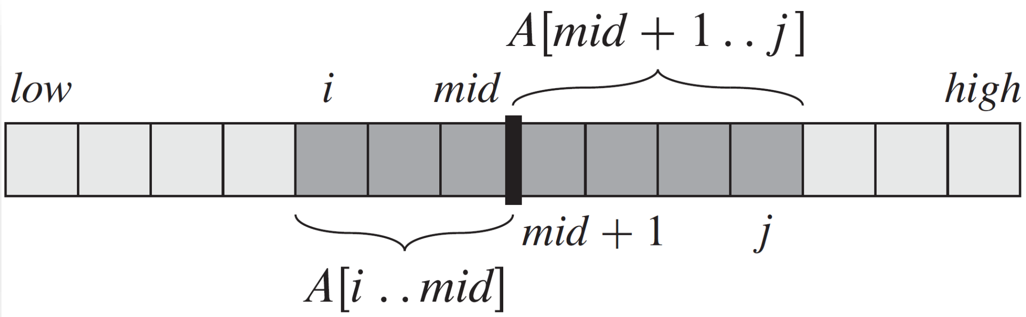


Maximum-subarray Problem……

- We can easily find a maximum subarray crossing the midpoint in time linear in the size of the subarray $A[low \cdots high]$



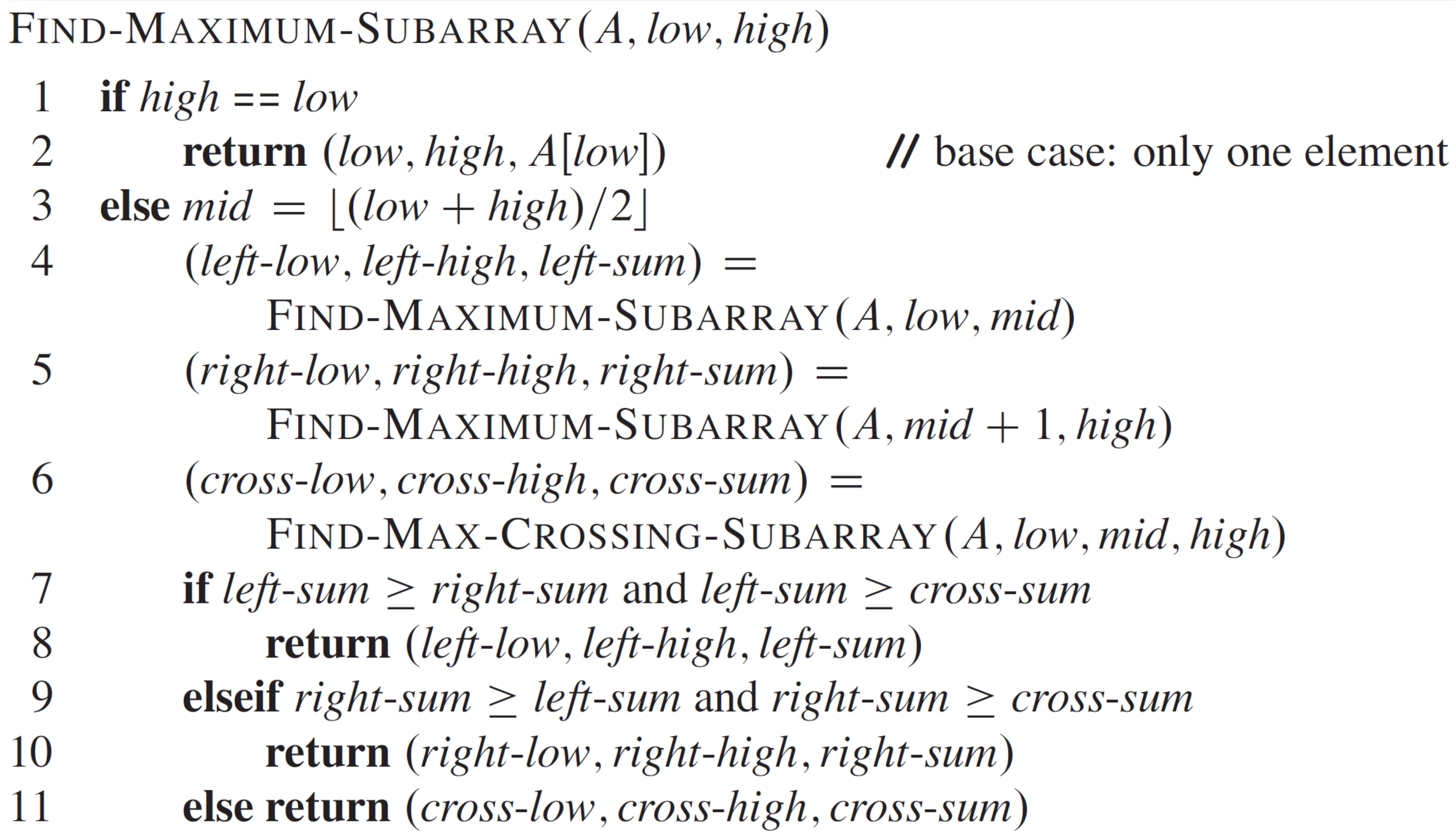
```
FIND-MAX-CROSSING-SUBARRAY$(A, low, mid, high)$
left-sum $= -\infty$
sum $= 0$
for $i$ = mid downto low
    sum = sum + $A[i]$
    if sum > left-sum
        left-sum = sum
        max-left = $i$
right-sum $= -\infty$
sum $= 0$
for $j$ = mid + 1 to high
    sum = sum + $A[j]$
    if sum > right-sum
        right-sum = sum
        max-right = $j$
return (max-left, max-right, left-sum + right-sum)
```



Maximum-subarray Problem…….

- To put everything together!

$$T(n) = \begin{cases} \Theta(1), & if\ n = 1 \\ 2T\left(\frac{n}{2}\right) + \Theta(n), & if\ n > 1 \end{cases}$$



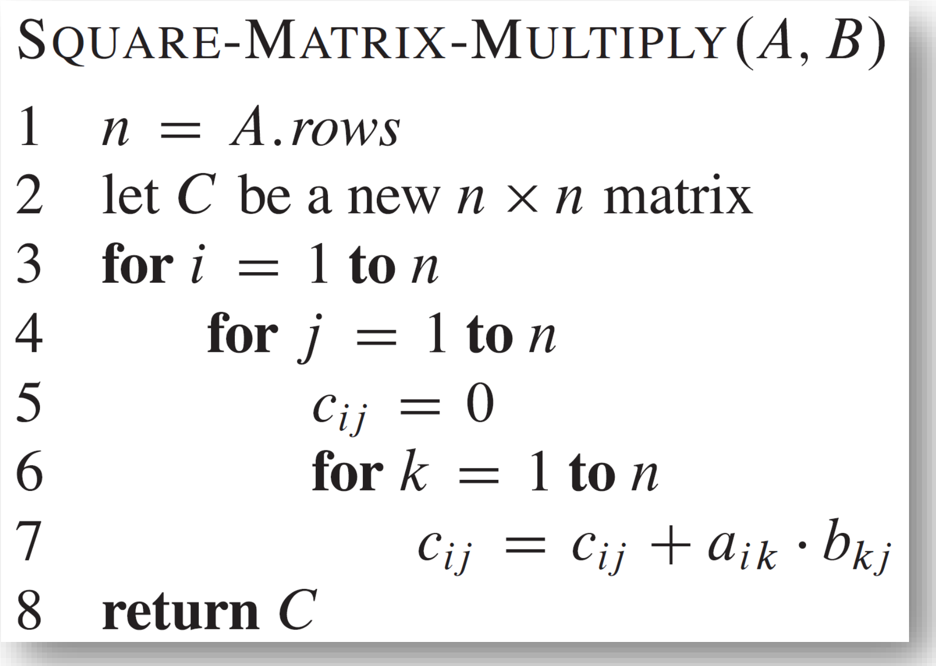
```
FIND-MAXIMUM-SUBARRAY(A, low, high)
if high == low
    return (low, high, A[low])          // base case: only one element
else mid = ⌊(low + high)/2⌋
    (left-low, left-high, left-sum) =
        FIND-MAXIMUM-SUBARRAY(A, low, mid)
    (right-low, right-high, right-sum) =
        FIND-MAXIMUM-SUBARRAY(A, mid + 1, high)
    (cross-low, cross-high, cross-sum) =
        FIND-MAX-CROSSING-SUBARRAY(A, low, mid, high)
    if left-sum ≥ right-sum and left-sum ≥ cross-sum
        return (left-low, left-high, left-sum)
    elseif right-sum ≥ left-sum and right-sum ≥ cross-sum
        return (right-low, right-high, right-sum)
    else return (cross-low, cross-high, cross-sum)
```

Maximum-subarray Problem……..

- The brute-force solution takes $\Omega(n^2)$ time
- A transformation approach takes $\Theta(n^2)$ time
- The divide-and-conquer method takes $\Theta(n \log_2 n)$ time
  - Faster than the brute-force method

Matrix Multiplication.

- If you have seen matrices before, then you probably know how to multiply them
    - $A = a_{ij}$
    - $B = b_{ij}$
    - A and B are $n \times n$ matrices
    - $C = AB$, $c_{ij} = \sum_{k=1}^{n} a_{ik} b_{kj}$
    - The *SQUARE-MATRIX-MULTIPLY* procedure takes $\Theta(n^3)$ time



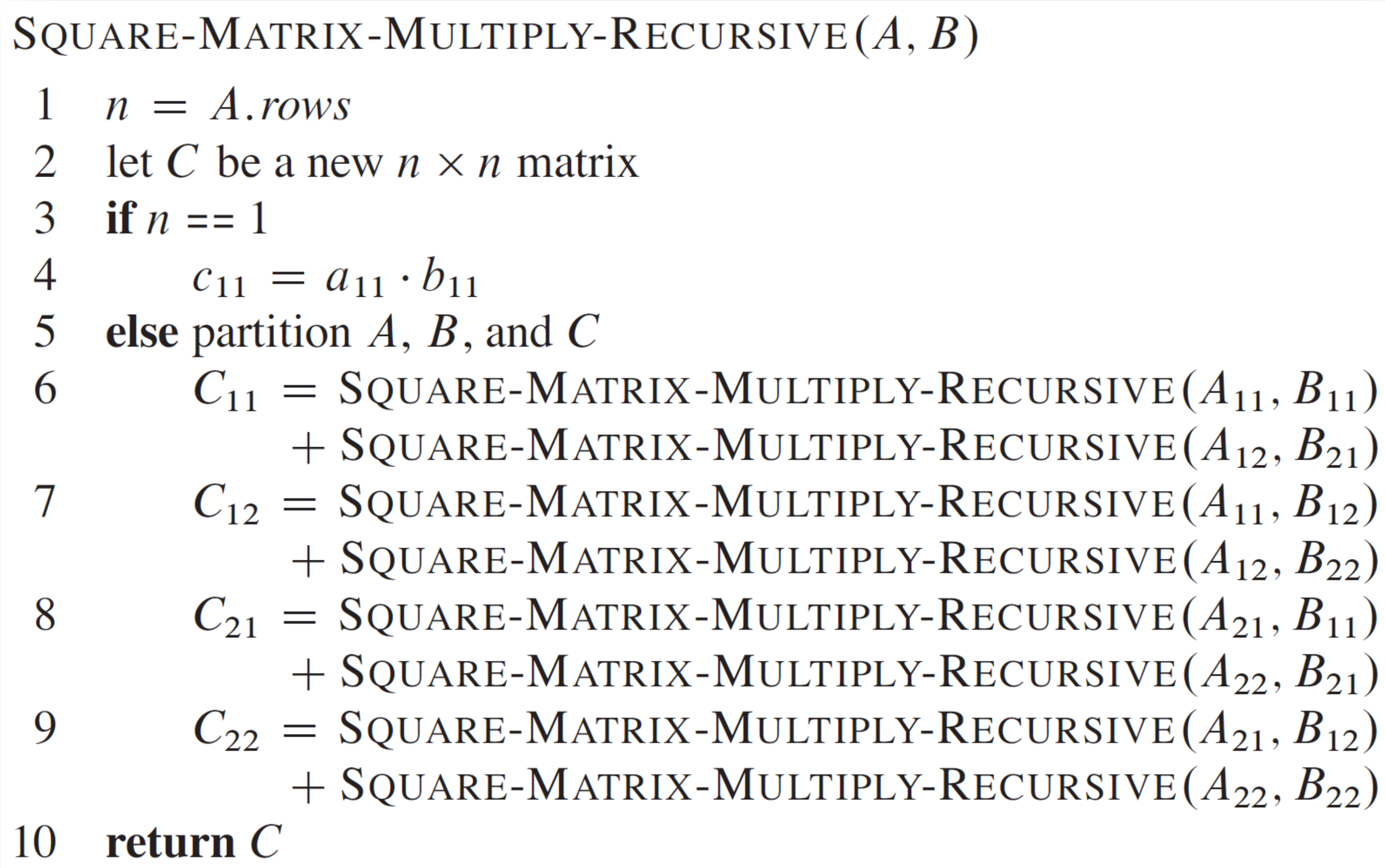
```
SQUARE-MATRIX-MULTIPLY($A$, $B$)
$n = A.rows$
let $C$ be a new $n \times n$ matrix
for $i = 1$ to $n$
    for $j = 1$ to $n$
        $c_{ij} = 0$
        for $k = 1$ to $n$
            $c_{ij} = c_{ij} + a_{ik} \cdot b_{kj}$
return $C$
```

Matrix Multiplication..

- **A simple divide-and-conquer algorithm**
  - Suppose that we partition each of A, B, and C into four $\frac{n}{2} \times \frac{n}{2}$ matrices
    - $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$
    - $\begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$
    - $\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$
    - Each of these four equations specifies two multiplications of $\frac{n}{2} \times \frac{n}{2}$ matrices and the addition of their $\frac{n}{2} \times \frac{n}{2}$ products

Matrix Multiplication…

- We can create a straightforward, recursive, divide-and-conquer algorithm



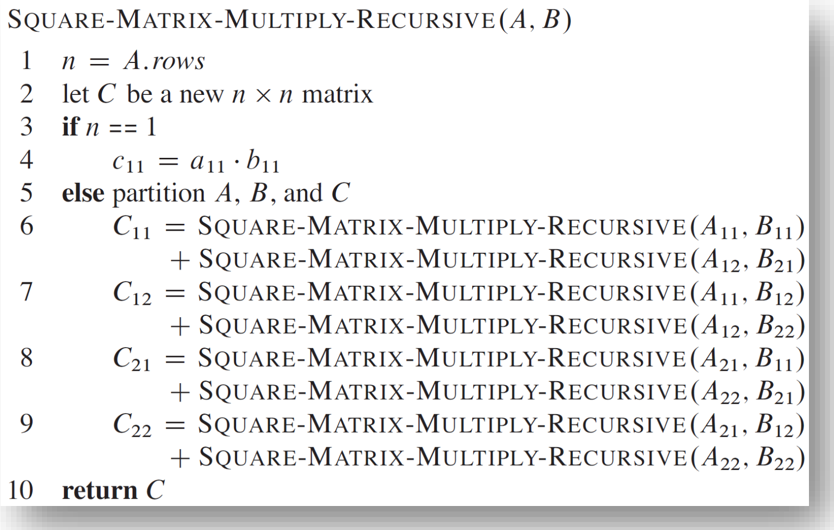
```
SQUARE-MATRIX-MULTIPLY-RECURSIVE$(A, B)$
  $n = A.rows$
  let $C$ be a new $n \times n$ matrix
  if $n == 1$
      $c_{11} = a_{11} \cdot b_{11}$
  else partition $A$, $B$, and $C$
      $C_{11}$ = SQUARE-MATRIX-MULTIPLY-RECURSIVE$(A_{11}, B_{11})$
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE$(A_{12}, B_{21})$
      $C_{12}$ = SQUARE-MATRIX-MULTIPLY-RECURSIVE$(A_{11}, B_{12})$
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE$(A_{12}, B_{22})$
      $C_{21}$ = SQUARE-MATRIX-MULTIPLY-RECURSIVE$(A_{21}, B_{11})$
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE$(A_{22}, B_{21})$
      $C_{22}$ = SQUARE-MATRIX-MULTIPLY-RECURSIVE$(A_{21}, B_{12})$
              + SQUARE-MATRIX-MULTIPLY-RECURSIVE$(A_{22}, B_{22})$
  return $C$
```

Matrix Multiplication….

- To sum up
  - The total time for the recursive case, therefore, is the sum of the partitioning time, the time for all the recursive calls, and the time to add the matrices resulting from the recursive calls

$$T(n) = \Theta(1) + 8T\left(\frac{n}{2}\right) + \Theta(n^2)$$

  - By the master method, it is easy to infer that $T(n) = \Theta(n^3)$
  - This simple divide-and-conquer approach is no faster than the straightforward SQUARE-MATRIX-MULTIPLY procedure



```
SQUARE-MATRIX-MULTIPLY-RECURSIVE(A, B)
n = A.rows
let C be a new n × n matrix
if n == 1
    c_11 = a_11 · b_11
else partition A, B, and C
    C_11 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_11)
           + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_21)
    C_12 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_11, B_12)
           + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_12, B_22)
    C_21 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_11)
           + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_21)
    C_22 = SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_21, B_12)
           + SQUARE-MATRIX-MULTIPLY-RECURSIVE(A_22, B_22)
return C
```

Matrix Multiplication…..

- **Strassen's Method**
  - The Strassen's method has four steps
    1. Divide the input matrices A and B and output matrix C into $\frac{n}{2} \times \frac{n}{2}$ submatrices
    2. Create 10 matrics $S_1, S_2, \cdots, S_{10}$, each of which is $\frac{n}{2} \times \frac{n}{2}$
    3. Compute seven matrix products $P_1, P_2, \cdots, P_7$, each of which is $\frac{n}{2} \times \frac{n}{2}$
    4. Compute the desired submatrices $C_{11}, C_{12}, C_{21}, C_{22}$

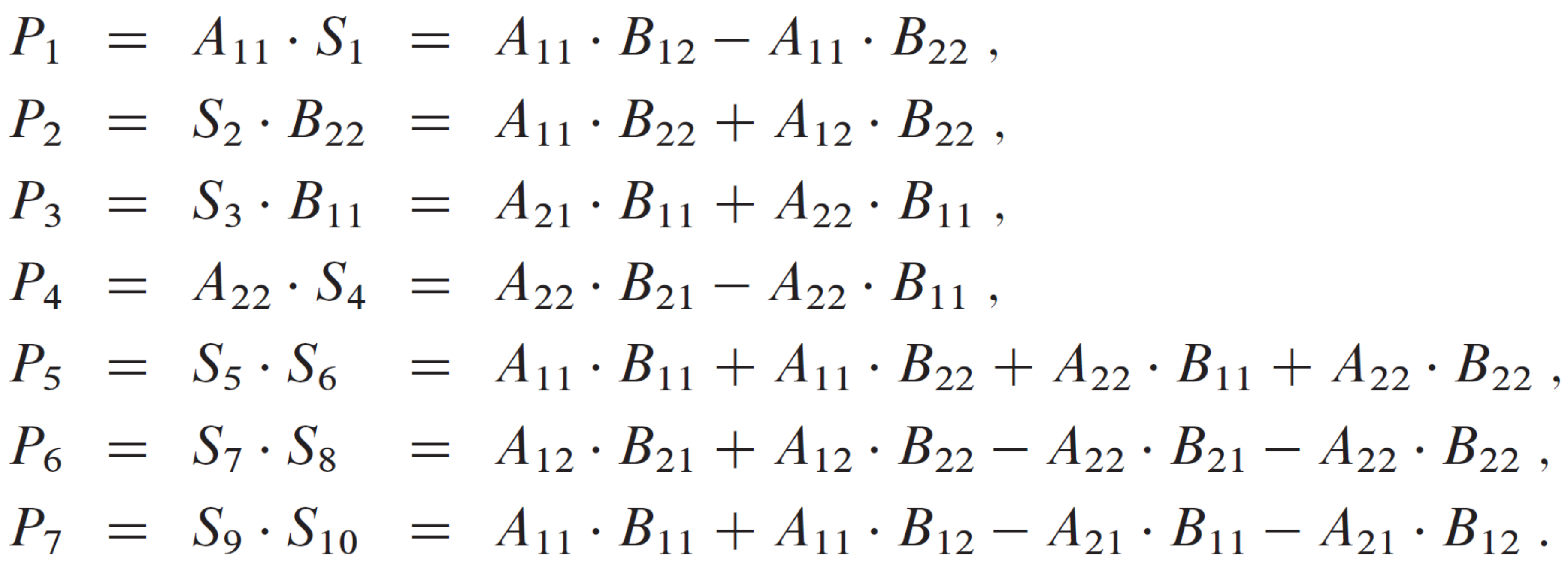
Matrix Multiplication……

- In step 1
    - $A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}, C = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$
    - $\Theta(1)$

- In step 2
    - Since we must add or subtract $\frac{n}{2} \times \frac{n}{2}$ matrices 10 times, this step does indeed take $\Theta(n^2)$ time
    - $S_1 = B_{12} - B_{22}, S_2 = A_{11} + A_{12}$
    - $S_3 = A_{21} + A_{22}, S_4 = B_{21} - B_{11}$
    - $S_5 = A_{11} + A_{22}, S_6 = B_{11} + B_{22}$
    - $S_7 = A_{12} - A_{22}, S_8 = B_{21} + B_{22}$
    - $S_9 = A_{11} - A_{21}, S_{10} = B_{11} + B_{12}$

Matrix Multiplication…….

- In step 3
  - Recursively multiply $\frac{n}{2} \times \frac{n}{2}$ matrices 7 times

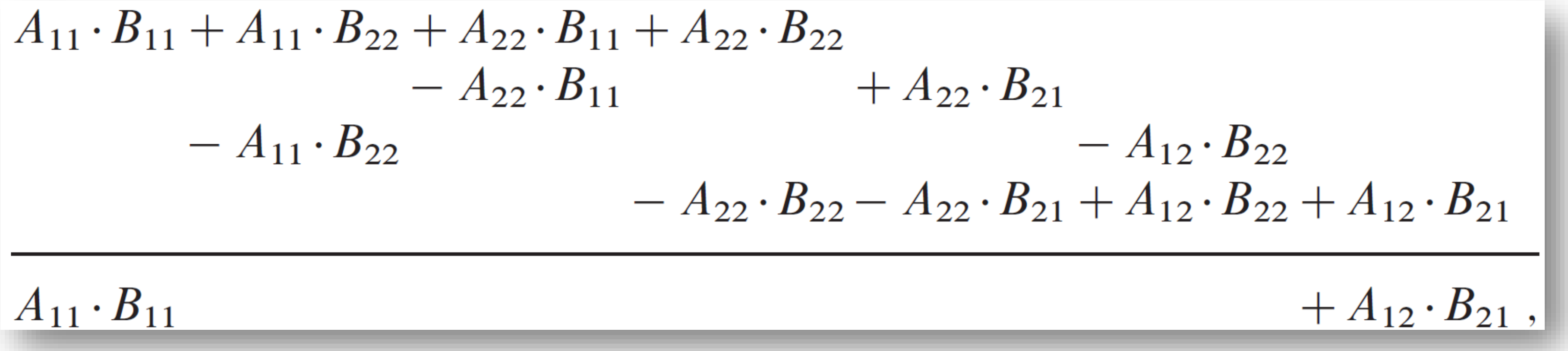


$$\begin{aligned}
P_1 &= A_{11} \cdot S_1 &&= A_{11} \cdot B_{12} - A_{11} \cdot B_{22} \,, \\
P_2 &= S_2 \cdot B_{22} &&= A_{11} \cdot B_{22} + A_{12} \cdot B_{22} \,, \\
P_3 &= S_3 \cdot B_{11} &&= A_{21} \cdot B_{11} + A_{22} \cdot B_{11} \,, \\
P_4 &= A_{22} \cdot S_4 &&= A_{22} \cdot B_{21} - A_{22} \cdot B_{11} \,, \\
P_5 &= S_5 \cdot S_6 &&= A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22} \,, \\
P_6 &= S_7 \cdot S_8 &&= A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22} \,, \\
P_7 &= S_9 \cdot S_{10} &&= A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12} \,.
\end{aligned}$$

Matrix Multiplication……..

$$\begin{aligned} C_{11} &= A_{11}B_{11} + A_{12}B_{21} \\ C_{12} &= A_{11}B_{12} + A_{12}B_{22} \\ C_{21} &= A_{21}B_{11} + A_{22}B_{21} \\ C_{22} &= A_{21}B_{12} + A_{22}B_{22} \end{aligned}$$

- In step 4
    - To construct the four $\frac{n}{2} \times \frac{n}{2}$ submatrices of C
    - $\Theta(n^2)$
    - $C_{11} = P_5 + P_4 - P_2 + P_6$



$$\begin{array}{lllllll}
A_{11} \cdot B_{11} & + A_{11} \cdot B_{22} & + A_{22} \cdot B_{11} & + A_{22} \cdot B_{22} & & & \\
 & & - A_{22} \cdot B_{11} & & + A_{22} \cdot B_{21} & & \\
 & - A_{11} \cdot B_{22} & & & & - A_{12} \cdot B_{22} & \\
 & & & - A_{22} \cdot B_{22} & - A_{22} \cdot B_{21} & + A_{12} \cdot B_{22} & + A_{12} \cdot B_{21} \\
\hline
A_{11} \cdot B_{11} & & & & & & + A_{12} \cdot B_{21} ,
\end{array}$$

$$\begin{aligned}
C_{11} &= A_{11}B_{11} + A_{12}B_{21}\\
C_{12} &= A_{11}B_{12} + A_{12}B_{22}\\
C_{21} &= A_{21}B_{11} + A_{22}B_{21}\\
C_{22} &= A_{21}B_{12} + A_{22}B_{22}
\end{aligned}$$

Matrix Multiplication………

- $C_{12} = P_1 + P_2$

$$\begin{array}{llll}
A_{11}\cdot B_{12} & - A_{11}\cdot B_{22} & & \\
 & + A_{11}\cdot B_{22} & + A_{12}\cdot B_{22} & \\
\hline
A_{11}\cdot B_{12} & & + A_{12}\cdot B_{22}\ , &
\end{array}$$

- $C_{21} = P_3 + P_4$

$$\begin{array}{llll}
A_{21}\cdot B_{11} & + A_{22}\cdot B_{11} & & \\
 & - A_{22}\cdot B_{11} & + A_{22}\cdot B_{21} & \\
\hline
A_{21}\cdot B_{11} & & + A_{22}\cdot B_{21}\ , &
\end{array}$$

- $C_{22} = P_5 + P_1 - P_3 - P_7$

$$\begin{array}{lllllll}
A_{11}\cdot B_{11} & + A_{11}\cdot B_{22} & + A_{22}\cdot B_{11} & + A_{22}\cdot B_{22} & & & \\
 & - A_{11}\cdot B_{22} & & & + A_{11}\cdot B_{12} & & \\
 & & - A_{22}\cdot B_{11} & & & - A_{21}\cdot B_{11} & \\
- A_{11}\cdot B_{11} & & & & - A_{11}\cdot B_{12} & + A_{21}\cdot B_{11} & + A_{21}\cdot B_{12} \\
\hline
 & & & A_{22}\cdot B_{22} & & & + A_{21}\cdot B_{12}\ ,
\end{array}$$

Matrix Multiplication………

– Consequently, the Strassen's method takes

$$T(n) = \Theta(1) + \Theta(n^2) + 7T\left(\frac{n}{2}\right) + \Theta(n^2)$$
$$= 7T\left(\frac{n}{2}\right) + \Theta(n^2)$$

– By the master method, you can derive that $T(n) = \Theta(n^{\log_2 7})$
  - Strassen's method is asymptotically faster than the straightforward SQUAREMATRIX-MULTIPLY procedure

Questions?

kychen@mail.ntust.edu.tw