Divide and Conquer
A Few Examples

S.V.N. (vishy) Vishwanathan

University of California, Santa Cruz
vishy@ucsc.edu

January 25, 2016
Maximum-Subarray Problem

Outline

1. Maximum-Subarray Problem
2. Integer Multiplication
3. Matrix Multiplication
A Stock Market Story

<table>
<thead>
<tr>
<th>Day</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td>100</td>
<td>113</td>
<td>110</td>
<td>85</td>
<td>105</td>
<td>102</td>
<td>86</td>
<td>63</td>
<td>81</td>
<td>101</td>
<td>94</td>
<td>106</td>
<td>101</td>
<td>79</td>
<td>94</td>
<td>90</td>
<td>97</td>
</tr>
<tr>
<td>Change</td>
<td>13</td>
<td>-3</td>
<td>-25</td>
<td>20</td>
<td>-3</td>
<td>-16</td>
<td>-23</td>
<td>18</td>
<td>20</td>
<td>-7</td>
<td>12</td>
<td>-5</td>
<td>-22</td>
<td>15</td>
<td>-4</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>
Rules of the Game

- Buy a unit of stock on any day after day 0
- Sell the stock after you buy it
- Profit = sell price - buy price
First Approach

- Buy at lowest price
- Sell at highest price
- Here high occurs on day 1, low occurs on day 7 :(
Second Approach

- Buy at lowest price
- Sell at highest price in the future after purchase date
- Here best strategy is to buy on day 2 and sell on day 3
Brute Force Approach

- For each \((i, j)\) such that \(j > i\)
- Compute profit
- Find the pair with the maximum profit
- This algorithm runs in \(O(.)\)?
Transform the Problem

Focus on the change in stock prices

<table>
<thead>
<tr>
<th>Day</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
<th>16</th>
</tr>
</thead>
<tbody>
<tr>
<td>Price</td>
<td>100</td>
<td>113</td>
<td>110</td>
<td>85</td>
<td>105</td>
<td>102</td>
<td>86</td>
<td>63</td>
<td>81</td>
<td>101</td>
<td>94</td>
<td>106</td>
<td>101</td>
<td>79</td>
<td>94</td>
<td>90</td>
<td>97</td>
</tr>
<tr>
<td>Change</td>
<td>13</td>
<td>-3</td>
<td>-25</td>
<td>20</td>
<td>-3</td>
<td>-16</td>
<td>-23</td>
<td>18</td>
<td>20</td>
<td>-7</td>
<td>12</td>
<td>-5</td>
<td>-22</td>
<td>15</td>
<td>-4</td>
<td>7</td>
<td></td>
</tr>
</tbody>
</table>

\[
A = \begin{bmatrix}
\end{bmatrix}
\]

maximum subarray
Now the Problem Becomes

- Find the nonempty, contiguous subarray of $A$ whose values have the largest sum
- Also called the maximum subarray problem
- Note: There can be more than one maximum subarrays
The Structure of the Problem

- Divide the array in $A[low \ldots mid]$ and $A[mid + 1 \ldots high]$. The maximum subarray $A[i \ldots j]$

- lies entirely in $A[low \ldots mid]$ (Case 1)

- lies entirely in $A[mid + 1 \ldots high]$ (Case 2)

- crosses the midpoint $low \leq i \leq mid \leq j \leq high$ (Case 3)
The maximum subarray $A[i..j]$

- lies entirely in $A[low..mid]$ (Case 1)
- lies entirely in $A[mid + 1..high]$ (Case 2)

Solve via recursion
Case 3

The maximum subarray $A[i..j]$

- crosses the midpoint $low \leq i \leq mid \leq j \leq high$ (Case 3)

Find maximum subarrays of the form

- $A[i..mid]$ and $A[mid+1..j]$

Time complexity?
Putting Everything Together

- **Divide:** Array $A$ into $A[low .. mid]$ and $A[mid + 1 .. high]$
- **Conquer**
  - Solve Case 1: $A[i .. j]$ lies in $A[low .. mid]$
  - Solve Case 2: $A[i .. j]$ lies in $A[mid + 1 .. high]$
  - Solve Case 3: Find maximum subarrays of the form $A[i .. mid]$ and $A[mid + 1 .. j]$ to find $A[i .. j]$ which crosses $mid$
- **Combine:** maximum of the three cases
Case 3: Pseudo-code

**Find-max-crossing-subarray** \((A, \text{low}, \text{mid}, \text{high})\)

1. \(\text{left-sum} = -\infty\)
2. \(\text{sum} = 0\)
3. for \(i = \text{mid} \) downto \(\text{low}\)
   
   \[
   \text{sum} = \text{sum} + A[i]
   \]
   
   if \(\text{sum} > \text{left-sum}\)
   
   \[
   \text{left-sum} = \text{sum}
   \]
   
   \[
   \text{max-left} = i
   \]
4. \(\text{right-sum} = -\infty\)
5. \(\text{sum} = 0\)
6. for \(j = \text{mid} + 1 \) to \(\text{high}\)
   
   \[
   \text{sum} = \text{sum} + A[j]
   \]
   
   if \(\text{sum} > \text{right-sum}\)
   
   \[
   \text{right-sum} = \text{sum}
   \]
   
   \[
   \text{max-right} = j
   \]
7. return \((\text{max-left}, \text{max-right}, \text{left-sum} + \text{right-sum})\)
Maximum Subarray: Pseudo-code

1. **Find-maximum-subarray**($A$, $low$, $high$)
   
   1. if $high == low$
   2. return $(low, high, A[low])$
   3. else $mid = \lfloor (low + high)/2 \rfloor$
   4. $(left-low, left-high, left-sum) =$  
      **Find-maximum-subarray**($A$, $low$, $mid$)
   5. $(right-low, right-high, right-sum) =$  
      **Find-maximum-subarray**($A$, $mid + 1$, $high$)
   6. $(cross-low, cross-high, cross-sum) =$  
      **Find-max-crossing-subarray**($A$, $low$, $mid$, $high$)
   7. if $left-sum \geq right-sum$ and $left-sum \geq cross-sum$
      return $(left-low, left-high, left-sum)$
   8. elseif $right-sum \geq left-sum$ and $right-sum \geq cross-sum$
      return $(right-low, right-high, right-sum)$
   9. else return $(cross-low, cross-high, cross-sum)$
**Time Complexity**

- **Base case:** \( T(1) = 1 \)
- \( T(n) = 2T(n/2) + \Theta(n) + \Theta(1) = 2T(n/2) + \Theta(n) \)
- Therefore \( T(n) = \Theta(?) \)
Outline

1. Maximum-Subarray Problem
2. Integer Multiplication
3. Matrix Multiplication
Problem

Given two integers \( x \) and \( y \), both \( n \) bits long, compute \( z = x \cdot y \). For simplicity assume \( n = 2^k \) for some \( k \).
First cut at Divide and Conquer

- Divide \( x \) into two parts \( x_L \) and \( x_R \) such that \( x = x_L 2^{n/2} + x_R \)
- Similarly divide \( y \) into two parts \( y_L \) and \( y_R \)
- Now compute

\[
x \cdot y = (x_L 2^{n/2} + x_R) \cdot (y_L 2^{n/2} + y_R)
= x_L \cdot y_L 2^n + (x_L \cdot y_R + x_R \cdot y_L) 2^{n/2} + x_R \cdot y_R.
\]
Time Complexity

\[ T(n) = 4T(n/2) + O(n) \]
Clever observation

- Divide $x$ into two parts $x_L$ and $x_R$ such that $x = x_L 2^{n/2} + x_R$
- Similarly divide $y$ into two parts $y_L$ and $y_R$
- Now compute

$$x \cdot y = (x_L 2^{n/2} + x_R) \cdot (y_L 2^{n/2} + y_R)$$

$$= x_L \cdot y_L 2^n + (x_L \cdot y_R + x_R \cdot y_L) 2^{n/2} + x_R \cdot y_R$$

$$= x_L \cdot y_L 2^n + ((x_L + x_R) \cdot (y_L + y_R) - x_L \cdot y_L - x_R \cdot y_R) 2^{n/2} + x_R \cdot y_R$$
Time Complexity

\[ T(n) = 3T(n/2) + O(n) \]
Outline

1 Maximum-Subarray Problem

2 Integer Multiplication

3 Matrix Multiplication
Matrix Multiplication

\[ \mathbf{C} = \mathbf{A} \cdot \mathbf{B} \]

\( \mathbf{A}, \mathbf{B}, \mathbf{C} \in \mathbb{R}^{n \times n} \). To make life simpler, assume \( n = 2^k \) for some \( k \).
Matrix Multiplication

Pseudocode

**SQUARE-MATRIX-MULTIPLY**(A, B)

1. \( m = A.rows \)
2. let \( C \) be a new \( n \times n \) matrix
3. for \( i = 1 \) to \( n \)
4.     for \( j = 1 \) to \( n \)
5.         \( c_{ij} = 0 \)
6.     for \( k = 1 \) to \( n \)
7.         \( c_{ij} = c_{ij} + a_{ik} \cdot b_{kj} \)
8. return \( C \)
Matrix Multiplication

Time Complexity

- Three nested loops
- Each loop runs for $\Theta(n)$ time
- Total time complexity $\Theta(n^3)$
Matrix Multiplication

Divide and Conquer - I

\[ A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}, \quad B = \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}, \quad C = \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} \]

\[ \begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \cdot \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix} \]
Divide and Conquer - II

\[
C_{11} = A_{11} \cdot B_{11} + A_{12} \cdot B_{21} \\
C_{12} = A_{11} \cdot B_{12} + A_{12} \cdot B_{22} \\
C_{21} = A_{21} \cdot B_{11} + A_{22} \cdot B_{21} \\
C_{22} = A_{21} \cdot B_{12} + A_{22} \cdot B_{22}
\]
Pseudocode

\textbf{Square-matrix-multiply-recursive}(A, B)

1. \( m = A\. \text{rows} \)
2. let \( C \) be a new \( n \times n \) matrix
3. \textbf{if} \( n == 1 \)
4. \hspace{1em} \( c_{11} = a_{11} \cdot b_{11} \)
5. \textbf{else} partition \( A, B, \) and \( C \) as in previous slide
6. \hspace{1em} \( C_{11} = \text{Square-matrix-multiply-recursive}(A_{11}, B_{11}) \)
   + \( \text{Square-matrix-multiply-recursive}(A_{12}, B_{21}) \)
7. \hspace{1em} \( C_{12} = \text{Square-matrix-multiply-recursive}(A_{11}, B_{12}) + \)
   \( \text{Square-matrix-multiply-recursive}(A_{12}, B_{22}) \)
8. \hspace{1em} \( C_{21} = \text{Square-matrix-multiply-recursive}(A_{21}, B_{11}) + \)
   \( \text{Square-matrix-multiply-recursive}(A_{22}, B_{21}) \)
9. \hspace{1em} \( C_{22} = \text{Square-matrix-multiply-recursive}(A_{21}, B_{12}) + \)
   \( \text{Square-matrix-multiply-recursive}(A_{22}, B_{22}) \)
10. \textbf{return} \( C \)
Time Complexity

- $T(1) = \Theta(1)$
- $T(n) = 8T(n/2) + \Theta(n^2)$
- Use master theorem to conclude that $T(n) = \Theta(n^3)$
Strassen’s Method

\[ S_1 = B_{12} - B_{22} \]
\[ S_2 = A_{11} + A_{12} \]
\[ S_3 = A_{21} + A_{22} \]
\[ S_4 = B_{21} - B_{11} \]
\[ S_5 = A_{11} + A_{22} \]
\[ S_6 = B_{11} + B_{22} \]
\[ S_7 = A_{12} - A_{22} \]
\[ S_8 = B_{21} + B_{22} \]
\[ S_9 = A_{11} - A_{21} \]
\[ S_{10} = B_{11} + B_{12} \]
Strassen’s Method

\[ P_1 = A_{11} \cdot S_1 = A_{11} \cdot B_{12} - A_{11} \]
\[ P_2 = S_2 \cdot B_{22} = A_{11} \cdot B_{22} + A_{12} \cdot B_{22} \]
\[ P_3 = S_3 \cdot B_{11} = A_{21} \cdot B_{11} + A_{22} \cdot B_{11} \]
\[ P_4 = A_{22} \cdot S_4 = A_{22} \cdot B_{21} - A_{22} \cdot B_{11} \]
\[ P_5 = S_5 \cdot S_6 = A_{11} \cdot B_{11} + A_{11} \cdot B_{22} + A_{22} \cdot B_{11} + A_{22} \cdot B_{22} \]
\[ P_6 = S_7 \cdot S_8 = A_{12} \cdot B_{21} + A_{12} \cdot B_{22} - A_{22} \cdot B_{21} - A_{22} \cdot B_{22} \]
\[ P_7 = S_9 \cdot S_{10} = A_{11} \cdot B_{11} + A_{11} \cdot B_{12} - A_{21} \cdot B_{11} - A_{21} \cdot B_{12} \]
Computing submatrices of $C$

\[
C_{11} = P_5 + P_4 - P_2 + P_6 \\
C_{12} = P_1 + P_2 \\
C_{21} = P_3 + P_4 \\
C_{22} = P_5 + P_1 - P_3 - P_7
\]
**Time Complexity**

- \( T(1) = \Theta(1) \)
- \( T(n) = 7T(n/2) + \Theta(n^2) \)
- Use master theorem to conclude that \( T(n) = \Theta(n^{\ln 7}) \approx \Theta(n^{2.81}) \)
Questions?