CSL 356: Analysis and Design of Algorithms

Ragesh Jaiswal

CSE, IIT Delhi

Dynamic Programming: Examples

Longest Common Subsequence

Dynamic Programming: Examples

- Problem(longest common subsequence): Let S and T be strings of characters. S is of length n and T is of length m. Find the *longest common subsequence*. This is the longest sequence of characters that appear in both S and T. The characters are not necessarily contiguous.
- Example: S = XYXZPQ, T = YXQYXP.
  - The longest common subsequence is XYXP
  - S=**XYX**Z**P**Q, T=Y**X**Q**YXP**

- Let $L(i, j)$ denote the length of the longest common subsequence in the strings ($S[1] \ldots S[i]$) and ($T[1] \ldots T[j]$)
- What is $L(1, j)$ for $1 < j \leq m$?

Dynamic Programming: Examples

- Problem(longest common subsequence): Let S and T be strings of characters. S is of length n and T is of length m. Find the *longest common subsequence*. This is the longest sequence of characters that appear in both S and T. The characters are not necessarily contiguous.
- Example: S = XYXZPQ, T =YXQYXP.
  - The longest common subsequence is XYXP
  - S=**XYX**Z**P**Q, T=Y**X**Q**YXP**

- Let $L(i, j)$ denote the length of the longest common subsequence in the strings $(S[1] \ldots S[i])$ and $(T[1] \ldots T[j])$
- What is $L(1, j)$ for $1 < j \leq m$?
  - 1 if $S[1]$ is present in the string $(T[1] \ldots T[j])$.

Dynamic Programming: Examples

- Problem(longest common subsequence): Let S and T be strings of characters. S is of length n and T is of length m. Find the *longest common subsequence*. This is the longest sequence of characters that appear in both S and T. The characters are not necessarily contiguous.
- Example: S = XYXZPQ, T =YXQYXP.
  - The longest common subsequence is XYXP
  - S=**XYX**Z**P**Q, T=Y**X**Q**YXP**

- Let $L(i, j)$ denote the length of the longest common subsequence in the strings ($S[1] \dots S[i]$) and ($T[1] \dots T[j]$)
- What is $L(1, j)$ for $1 < j \leq m$?
  - 1 if $S[1]$ is present in the string ($T[1] \dots T[j]$).
  - 1 if $S[1] = T[j]$ else $L(1, j) = L(1, j-1)$.

Dynamic Programming: Examples

- Let $L(i,j)$ denote the length of the longest common subsequence in the strings $(S[1] \dots S[i])$ and $(T[1] \dots T[j])$
- What is $L(1,j)$ for $1 < j \leq m$?
  - 1 if $S[1]$ is present in the string $(T[1] \dots T[j])$.
  - 1 if $S[1] = T[j]$ else $L(1,j) = L(1,j-1)$.
- Similarly, we can define $L(i,1)$.
- Can you say something similar for $L(i,j)$ for $i,j \neq 1$?

Dynamic Programming: Examples

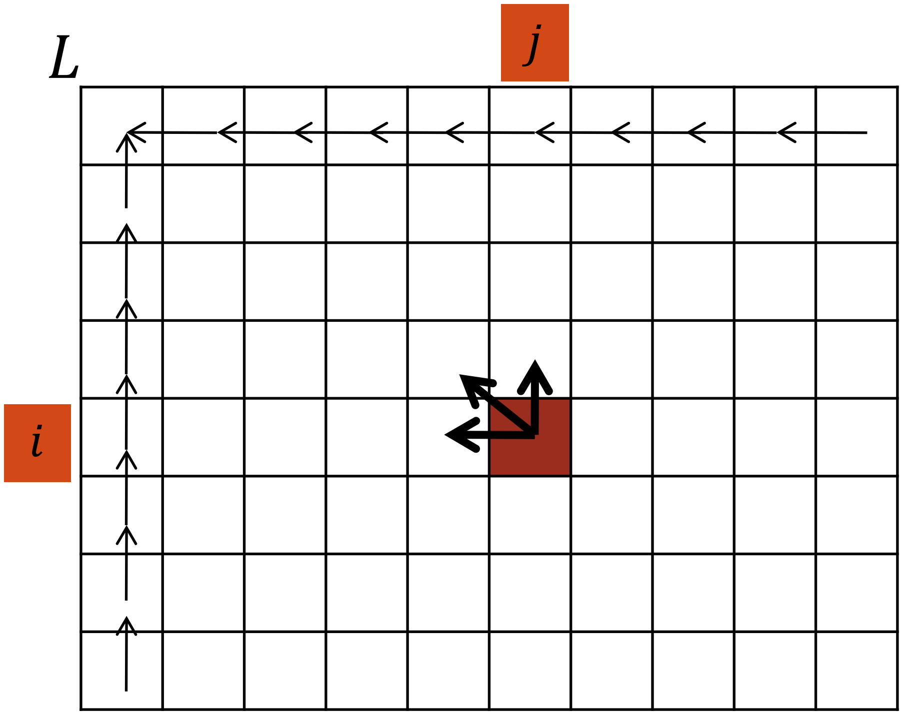
- Let $L(i, j)$ denote the length of the longest common subsequence in the strings $(S[1] \ldots S[i])$ and $(T[1] \ldots T[j])$
- What is $L(1, j)$ for $1 < j \leq m$?
  - 1 if $S[1]$ is present in the string $(T[1] \ldots T[j])$.
  - 1 if $S[1] = T[j]$ else $L(1, j) = L(1, j - 1)$.
- Similarly, we can define $L(i, 1)$.
- Can you say something similar for $L(i, j)$ for $i, j \neq 1$?
  - <u>Claim 1</u>: If $S[i] = T[j]$, then $L(i, j) = 1 + L(i - 1, j - 1)$.

Dynamic Programming: Examples

- Let $L(i, j)$ denote the length of the longest common subsequence in the strings $(S[1] \dots S[i])$ and $(T[1] \dots T[j])$
- What is $L(1, j)$ for $1 < j \leq m$?
  - 1 if $S[1]$ is present in the string $(T[1] \dots T[j])$.
  - 1 if $S[1] = T[j]$ else $L(1, j) = L(1, j - 1)$.
- Similarly, we can define $L(i, 1)$.
- Can you say something similar for $L(i, j)$ for $i, j \neq 1$?
  - <u>Claim 1</u>: If $S[i] = T[j]$, then $L(i, j) = 1 + L(i - 1, j - 1)$.
  - <u>Claim 2</u>: If $S[i] \neq T[j]$, then $L(i, j) = \max(L(i - 1, j), L(i, j - 1))$.

Dynamic Programming: Examples

- What is $L(1, j)$ for $1 < j \leq m$?
  - 1 if $S[1]$ is present in the string $(T[1] \dots T[j])$.
  - 1 if $S[1] = T[j]$ else $L(1, j) = L(1, j-1)$.
- Can you say something similar for $L(i, j)$ for $i, j \neq 1$?
  - Claim 1: If $S[i] = T[j]$, then $L(i, j) = 1 + L(i-1, j-1)$.
  - Claim 2: If $S[i] \neq T[j]$, then $L(i, j) = \max(L(i-1, j), L(i, j-1))$.



- The black arrows show dependencies between sub-problems.

Dynamic Programming: Examples

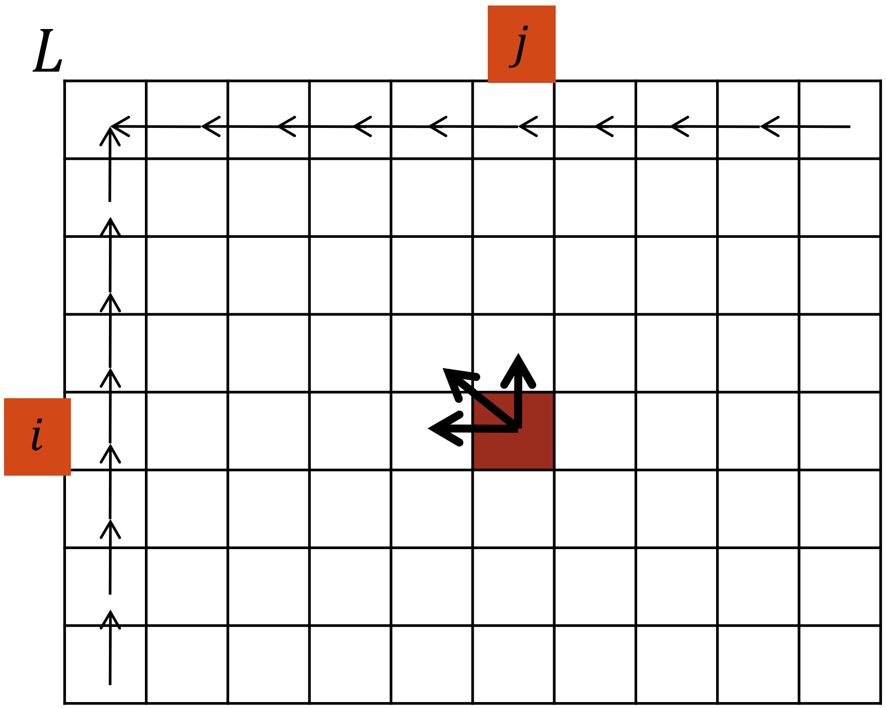
Length-LCS(S, T)

- if ($S[1] = T[1]$)then $L[1,1] = 1$ else $L[1,1] = 0$
- for $j = 2$ to m
    - If ($S[1] = T[j]$) then $L[1,j] = 1$ else $L[1,j] = L[1,j-1]$
- for $i = 2$ to n
    - If ($S[i] = T[1]$) then $L[i,1] = 1$ else $L[i,1] = L[i-1,1]$
- for $i = 2$ to n
    - for $j = 2$ to m
        - if ($S[i] = T[j]$) then $L[i,j] = 1 + L[i-1, j-1]$

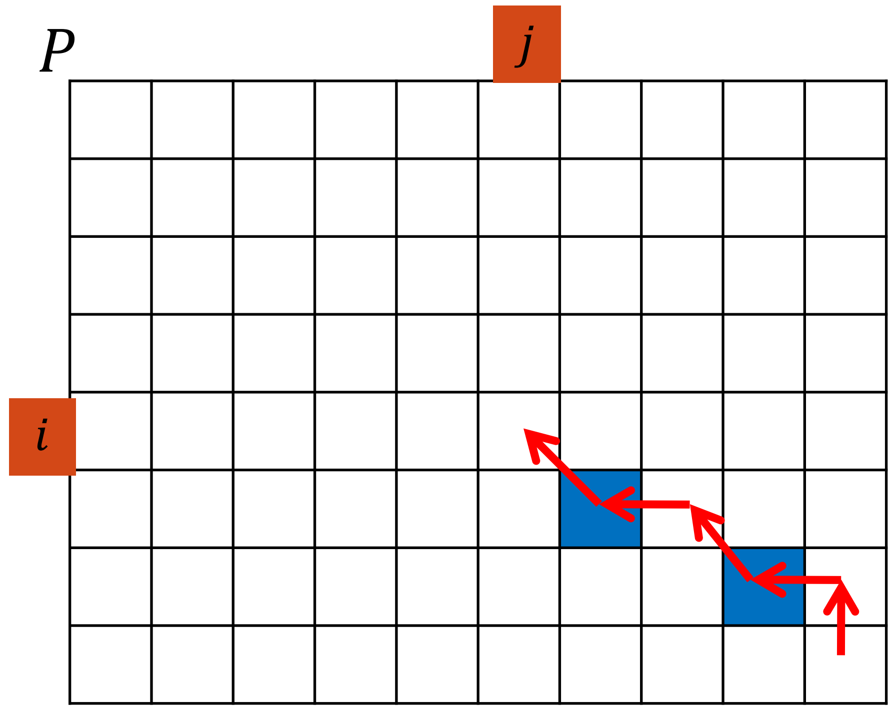
          else $L[i,j] = \max(L[i-1,j], L[i,j-1])$

Dynamic Programming: Examples

- How do we find the longest common subsequence?



- The black arrows show dependencies between sub-problems.



- Array P is used to maintain the pointers to the appropriate sub-problem.
- The blue squares give the position of the characters in a longest common subsequence.

Dynamic Programming: Examples

- Example: S= XYXZPQ, T=YXQYXP

L

0	1	1	1	1	1
1					
1					
1					
1					
1					

P

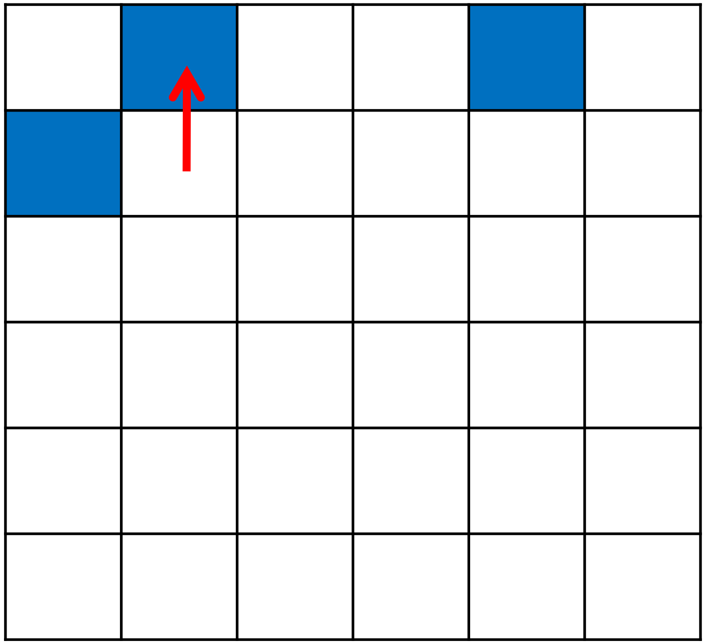
Dynamic Programming: Examples

- Example: S= XYXZPQ, T=YXQYXP

L

0	1	1	1	1	1
1	1				
1					
1					
1					
1					

P



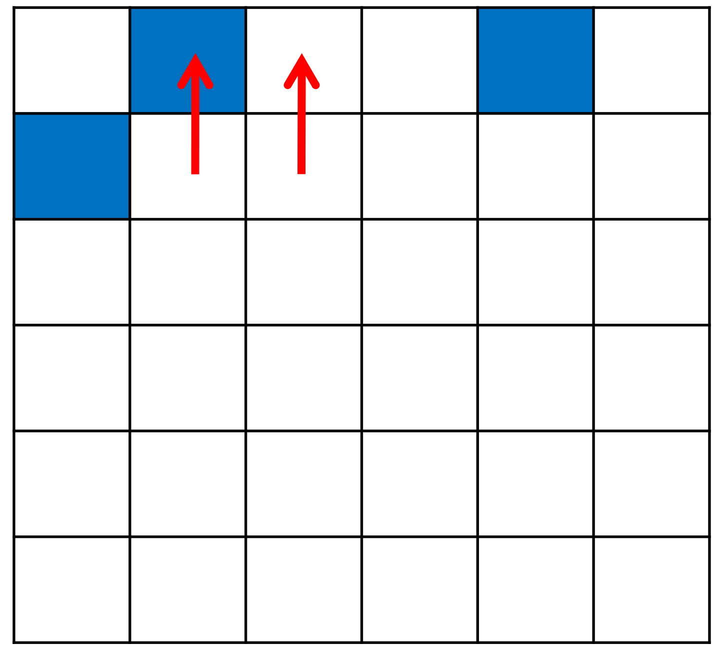
Dynamic Programming: Examples

- Example: S= XYXZPQ, T=YXQYXP

L

0	1	1	1	1	1
1	1	1			
1					
1					
1					
1					

P



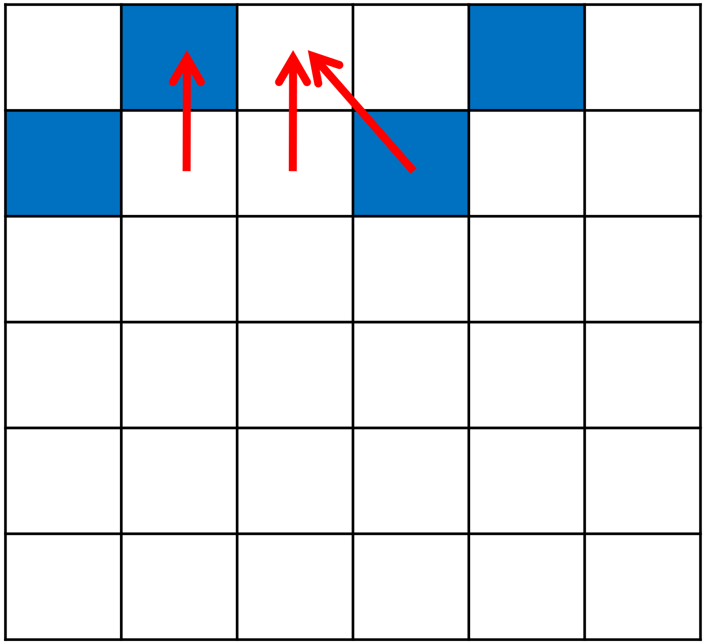
Dynamic Programming: Examples

- Example: S= XYXZPQ, T=YXQYXP

L

0	1	1	1	1	1
1	1	1	2		
1					
1					
1					
1					

P



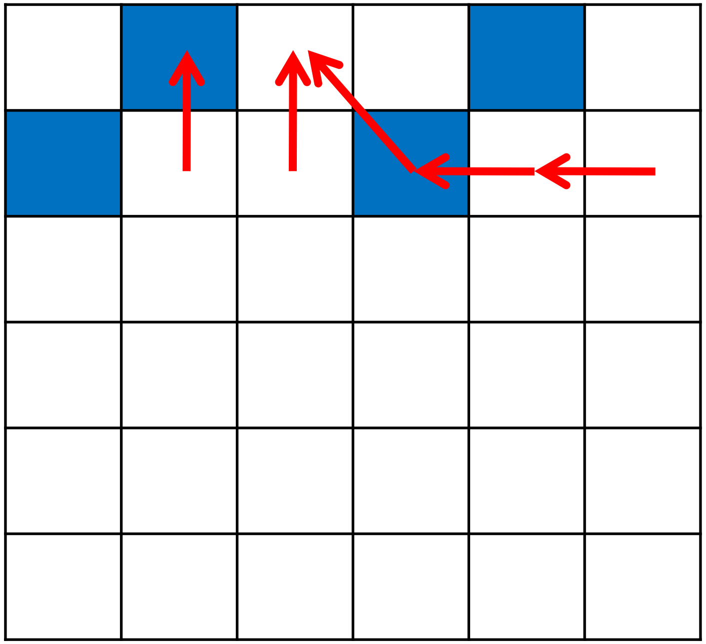
Dynamic Programming: Examples

- Example: S= XYXZPQ, T=YXQYXP

L

0	1	1	1	1	1
1	1	1	2	2	2
1					
1					
1					
1					

P



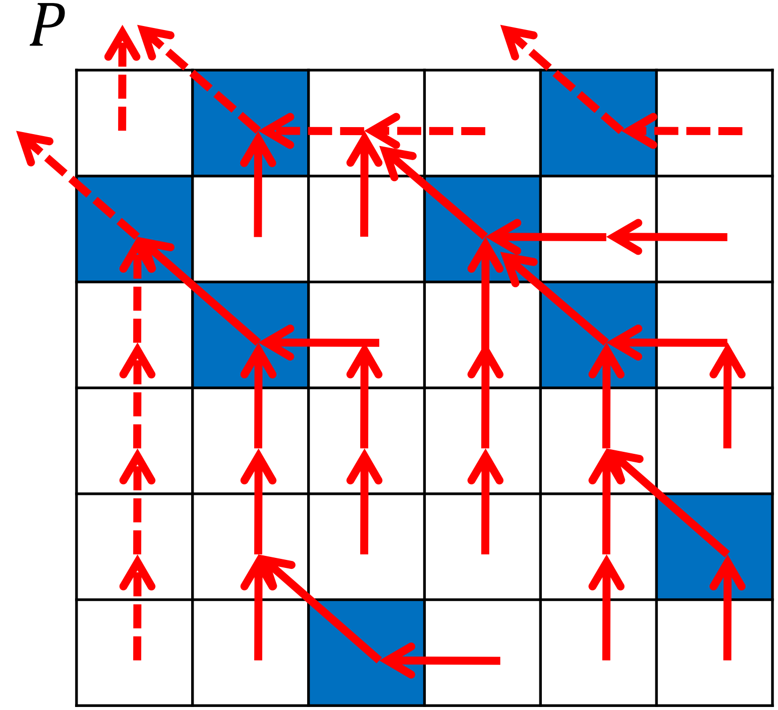
Dynamic Programming: Examples

- Example: S= XYXZPQ, T=YXQYXP

L

0	1	1	1	1	1
1	1	1	2	2	2
1	2	2	2	3	3
1	2	2	2	3	3
1	2	2	2	3	4
1	2	3	3	3	4

P



Dynamic Programming

Memoization

Dynamic Programming: Examples

- Problem(longest common subsequence): Let S and T be strings of characters. S is of length n and T is of length m. Find the *longest common subsequence*. This is the longest sequence of characters that appear in both S and T. The characters are not necessarily contiguous.
- Example: S = XYXZPQ, T = YXQYXP.
  - The longest common subsequence is XYXP
  - S=**XYX**Z**P**Q, T=Y**X**Q**YXP**

- Claim 1: If $i = 0$ or $j = 0$, then $L(i, j) = 0$.
- Claim 2: If $S[i] = T[j]$, then $L(i, j) = 1 + L(i - 1, j - 1)$.
- Claim 3: If $S[i] \neq T[j]$, then $L(i, j) = \max(L(i - 1, j), L(i, j - 1))$.

Dynamic Programming: Examples

- Recursive program to find the length of the longest common subsequence.

Length-LCS(S, n, T, m)

- if ($n = 0$ OR $m = 0$) then return(0)
- if ($S[n] = T[m]$) return(1+Length-LCS($S, n-1, T, m-1$))
- if ($S[n] \neq T[m]$)
  return(max(Length-LCS($S, n, T, m-1$)),Length-LCS($S, n-1, T, m$)))

- What is the running time of the above algorithm?
  - Could be exponentially large in the worst case!

Dynamic Programming: Examples

- Memoized version of the algorithm:

Length-LCS(S, n, T, m)

- if ($n = 0$ OR $m = 0$) then return(0)
- if ($L[n, m]$ is known) then return($L[n, m]$)
- if ($S[n] = T[m]$) then

  $length = 1 +$Length-LCS($S, n - 1, T, m - 1$)
- if ($S[n] \neq T[m]$) then

  $length =$ max(Length-LCS($S, n, T, m - 1$)),Length-LCS($S, n - 1, T, m$))
- $L[n, m] = length$
- return($length$)

- What is the running time of the above algorithm?

Dynamic Programming: Examples

- Memoized version of the algorithm:

Length-LCS(S, n, T, m)

- if ($n = 0$ OR $m = 0$) then return(0)
- if ($L[n, m]$ is known) then return($L[n, m]$)
- if ($S[n] = T[m]$) then

  $length = 1 +$Length-LCS($S, n-1, T, m-1$)
- if ($S[n] \neq T[m]$) then

  $length =$ max(Length-LCS($S, n, T, m-1$)),Length-LCS($S, n-1, T, m$))
- $L[n, m] = length$
- return($length$)

- What is the running time of the above algorithm?
  - $O(nm)$

Dynamic Programming: Examples

Matrix Chain Multiplication

Dynamic Programming: Examples

- <u>Problem(matrix chain multiplication)</u>: You are given a sequence of n matrices $M_1, \ldots, M_n$ of size $(m_1 \times m_2), (m_2 \times m_3), \ldots, (m_n \times m_{n+1})$. Determine in what order these matrices should be multiplied (using naïve method) so as to reduce the total running time.
- <u>Example</u>: Consider four matrices of size
  - M_1: 50×20
  - M_2: 20×1
  - M_3: 1×10
  - M_4: 10×100

- $M_1 \times M_2 \times M_3 \times M_4 \ = \ M_1 \times ((M_2 \times M_3) \times M_4)$
  - Time:
- $M_1 \times M_2 \times M_3 \times M_4 \ = \ (M_1 \times (M_2 \times M_3)) \times M_4$
  - Time:
- $M_1 \times M_2 \times M_3 \times M_4 \ = \ (M_1 \times M_2) \times (M_3 \times M_4)$
  - Time:

Dynamic Programming: Examples

- Problem(matrix chain multiplication): You are given a sequence of n matrices $M_1, \dots, M_n$ of size $(m_1 \times m_2), (m_2 \times m_3), \dots, (m_n \times m_{n+1})$. Determine in what order these matrices should be multiplied (using naïve method) so as to reduce the total running time.
- Example: Consider four matrices of size
  - M_1: 50×20
  - M_2: 20×1
  - M_3: 1×10
  - M_4: 10×100

- $M_1 \times M_2 \times M_3 \times M_4 = M_1 \times ((M_2 \times M_3) \times M_4)$
  - Time: $20 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100$
- $M_1 \times M_2 \times M_3 \times M_4 = (M_1 \times (M_2 \times M_3)) \times M_4$
  - Time: $20 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100$
- $M_1 \times M_2 \times M_3 \times M_4 = (M_1 \times M_2) \times (M_3 \times M_4)$
  - Time: $50 \cdot 20 + 10 \cdot 100 + 50 \cdot 100$

Dynamic Programming: Examples

- $C(i,j)$: Minimum cost of multiplying matrices $M_i, \ldots, M_j$.
- $C(i,i) = 0$
- $C(i,j)$?

Dynamic Programming: Examples

- $C(i,j)$: Minimum cost of multiplying matrices $M_i, \ldots, M_j$.
- $C(i,i) = 0$
- $C(i,j) = \min_{i \le k < j} \left(C(i,k) + C(k+1,j) + m_i \cdot m_{k+1} \cdot m_{j+1} \right)$

Matrix-Cost($M_1, \ldots, M_n$)

- for $i = 1$ to n
    - $C[i,i] = 0$
- for $s = 1$ to $n - 1$
    - for $i = 1$ to $n - s$
        - $j = i + s$
        - $C[i,j] = \min_{i \le k < j} \left(C[i,k] + C[k+1,j] + m_i \cdot m_{k+1} \cdot m_{j+1} \right)$
- return($C[1,n]$)

- Running time:

Dynamic Programming: Examples

- $C(i,j)$: Minimum cost of multiplying matrices $M_i, \ldots, M_j$.
- $C(i,i) = 0$
- $C(i,j) = \min_{i \le k < j} \left(C(i,k) + C(k+1,j) + m_i \cdot m_{k+1} \cdot m_{j+1} \right)$

Matrix-Cost($M_1, \ldots, M_n$)
- for $i = 1$ to n
  - $C[i,i] = 0$
- for $s = 1$ to $n - 1$
  - for $i = 1$ to $n - s$
    - $j = i + s$
    - $C[i,j] = \min_{i \le k < j} \left(C[i,k] + C[k+1,j] + m_i \cdot m_{k+1} \cdot m_{j+1} \right)$
- return($C[1,n]$)

- Running time: $O(n^3)$.

End