CS 4102: Algorithms

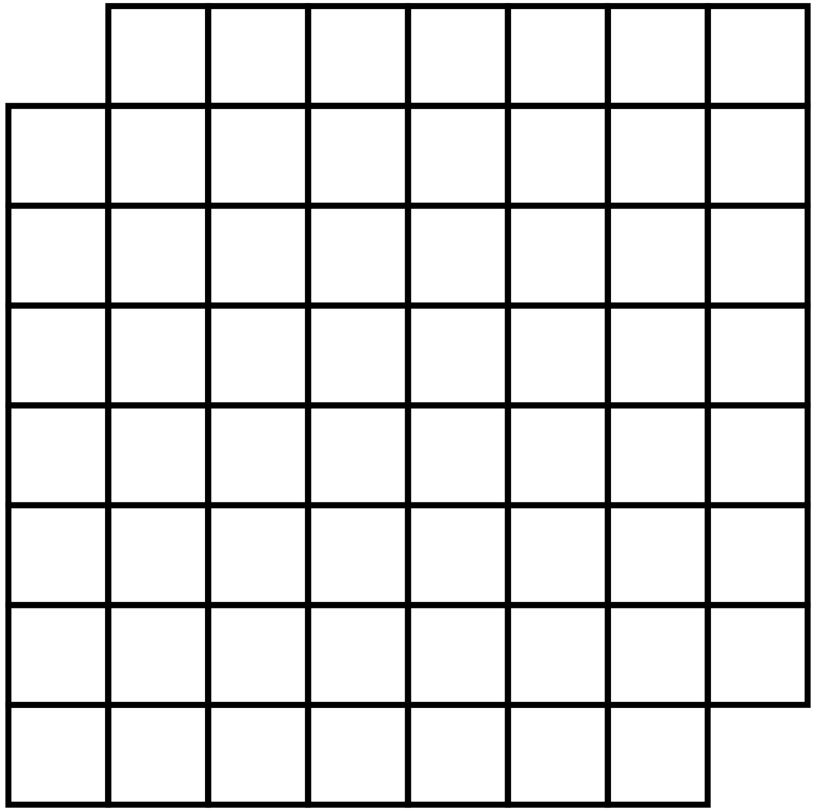
Lecture 12: Dynamic Programming

David Wu

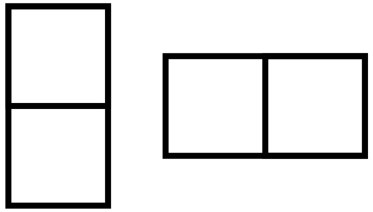
Fall 2019

Warm-Up

Problem: Can you fill a 8×8 board with two corners missing using 2×1 dominoes?

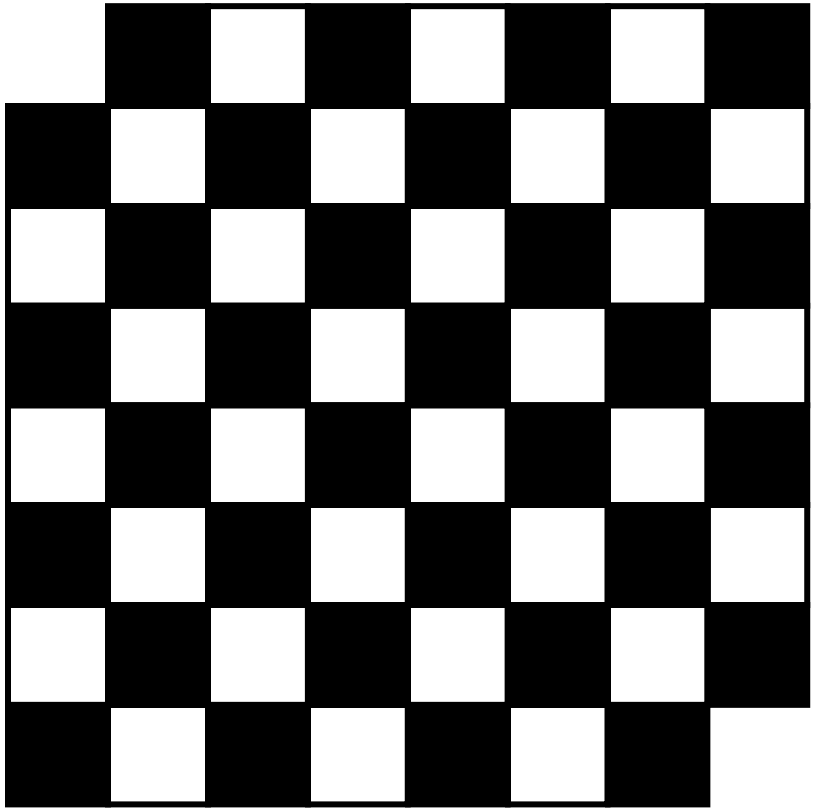


Dominoes:

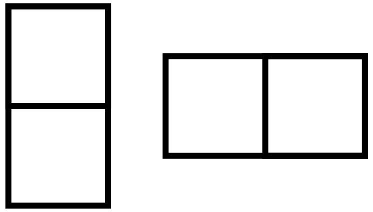


Warm-Up

Problem: Can you fill a 8×8 board with two corners missing using 2×1 dominoes?



Dominoes:



32 black squares
30 white squares

Today’s Keywords

Dynamic Programming

Matrix Chaining (Review)

Longest Common Subsequence

Seam Carving

CLRS Readings: Chapter 14

Homework

- **HW4** due **Saturday, October 12, 11pm**
  - Divide and conquer, sorting, and dynamic programming
  - Written (use LaTeX!) – Submit both **zip** and **pdf** (two separate attachments)!

- **HW5** released next week (after exam)
  - Seam Carving
  - Dynamic Programming (implementation)
  - Java or Python

Midterm

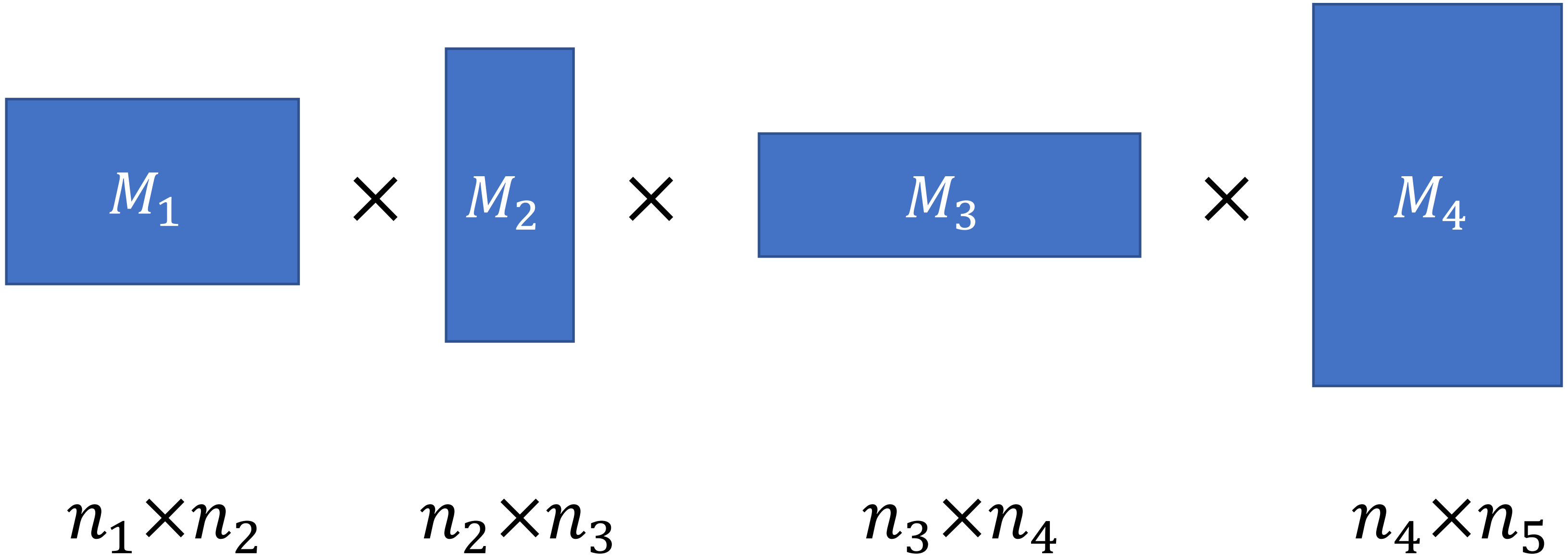
- **Tuesday, October 15 (in class)**
  - SDAC: Please schedule with SDAC for Tuesday
  - Mostly in-class with a take-home portion

- **Practice Midterm** (Last Semester’s Midterm) available on Collab today

- **Optional Review Session:** Sunday, October 13 at 3pm, Olsson 120

Review: Matrix Chaining

Problem: Given a sequence of matrices $M_1, \dots, M_n$, what is the most efficient way to multiply them?



Remember: matrix multiplication is associative

Identify Recursive Structure

More generally:

$\text{Best}(i, j)$ = cheapest way to multiply together M_i through M_j

Possible ways to compute $M_i \times M_{i+1} \times \cdots \times M_j$

$$M_i \times M_{i+1,j} = M_i \times (M_{i+1} \times \cdots \times M_j)$$

$$M_{i,i+1} \times M_{i+2,j} = (M_i \times M_{i+1}) \times (M_{i+2} \times \cdots \times M_j)$$

$$M_{i,i+2} \times M_{i+3,j} = (M_i \times M_{i+1} \times M_{i+2}) \times (M_{i+3} \times \cdots \times M_j)$$

$$\vdots \qquad\qquad \vdots \qquad\qquad \vdots$$

$$M_{i,j-1} \times M_j = (M_i \times \cdots \times M_{j-1}) \times M_j$$

Position of the "split" changes

Identify Recursive Structure

More generally:

$\text{Best}(i, j) =$ cheapest way to multiply together M_i through M_j

Possible ways to compute $M_i \times M_{i+1} \times \cdots \times M_j$

$$\text{Best}(i, j) = \min \begin{cases} \text{Best}(i, i) + \text{Best}(i+1, j) + n_i n_{i+1} n_{j+1} \\ \text{Best}(i, i+1) + \text{Best}(i+2, j) + n_i n_{i+2} n_{j+1} \\ \text{Best}(i, i+2) + \text{Best}(i+3, j) + n_i n_{i+3} n_{j+1} \\ \\ \text{Best}(i, j-1) + \text{Best}(j, j) + n_i n_j n_{j+1} \end{cases}$$

$$\text{Best}(i, i) = 0$$

$$\text{Best}(i, j) = \min_{k=0,\dots,j-i-1} \text{Best}(i, i+k) + \text{Best}(i+k+1, j) + n_i n_{i+k+1} n_{j+1}$$

Select a Good Order for Solving Subproblems

$\text{Best}(i, i) = 0$

$$\text{Best}(i, j) = \min_{k=0,\ldots,j-i-1} \text{Best}(i, i+k) + \text{Best}(i+k+1, j) + n_i n_{i+k+1} n_{j+1}$$

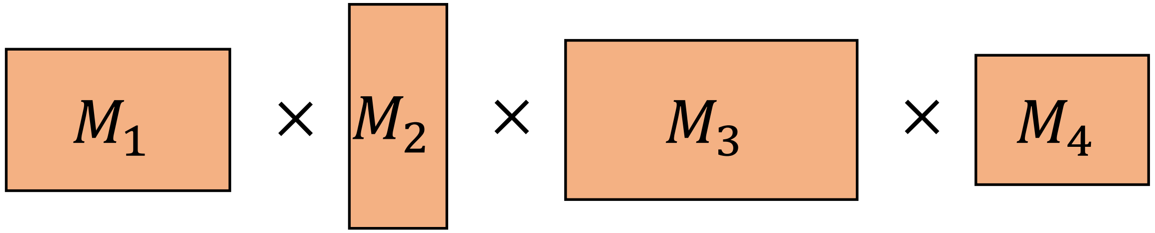
$i = 1, j = 4$

j

i	1	2	3	4
1				▨
2				
3				
4				

$n_1 = 5 \qquad n_2 = 10$

$n_3 = 20 \qquad n_4 = 8 \qquad n_5 = 6$



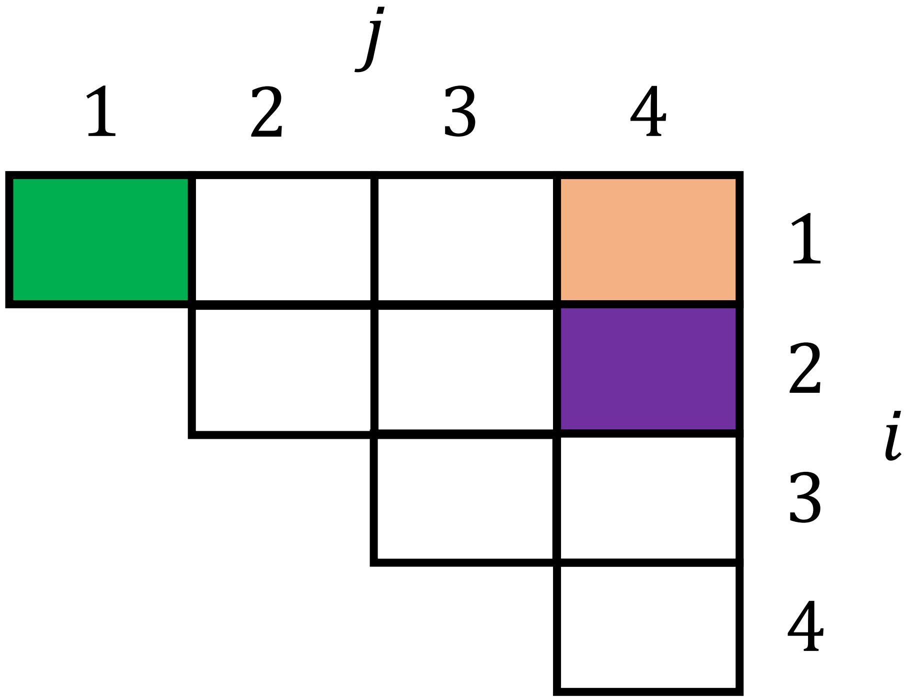
Select a Good Order for Solving Subproblems

$\text{Best}(i, i) = 0$

$$\text{Best}(i, j) = \min_{k=0,\ldots,j-i-1} \text{Best}(i, i+k) + \text{Best}(i+k+1, j) + n_i n_{i+k+1} n_{j+1}$$

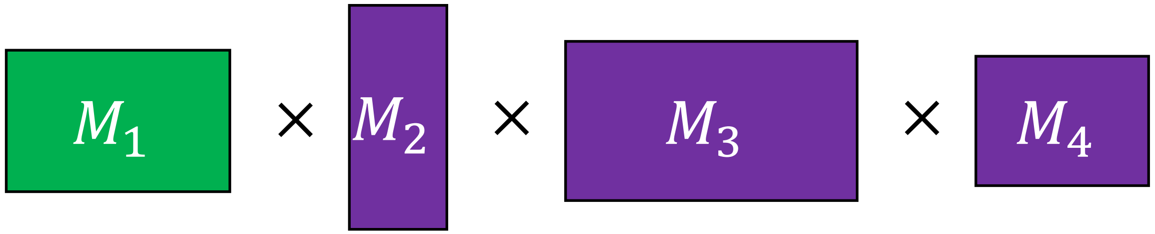
$i = 1, j = 4$

$k = 0$



$n_1 = 5$ $n_2 = 10$

$n_3 = 20$ $n_4 = 8$ $n_5 = 6$



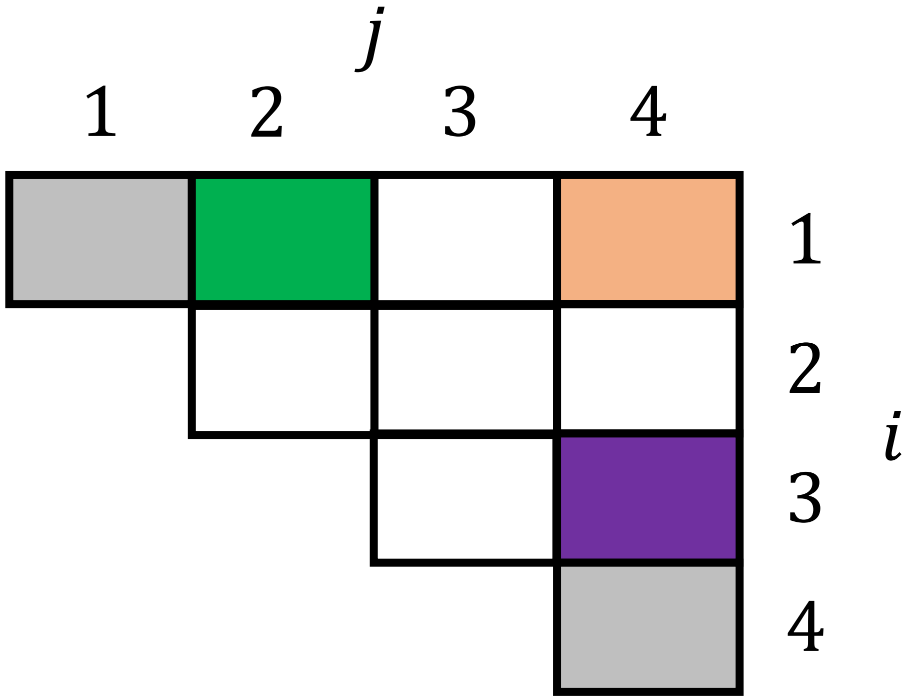
Select a Good Order for Solving Subproblems

$\text{Best}(i,i) = 0$

$$\text{Best}(i,j) = \min_{k=0,\dots,j-i-1} \text{Best}(i,i+k) + \text{Best}(i+k+1,j) + n_i n_{i+k+1} n_{j+1}$$

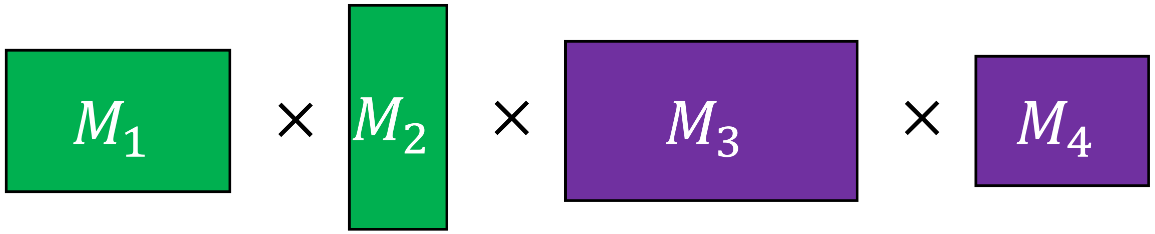
$i = 1, j = 4$

$k = 1$



$n_1 = 5$ $n_2 = 10$

$n_3 = 20$ $n_4 = 8$ $n_5 = 6$



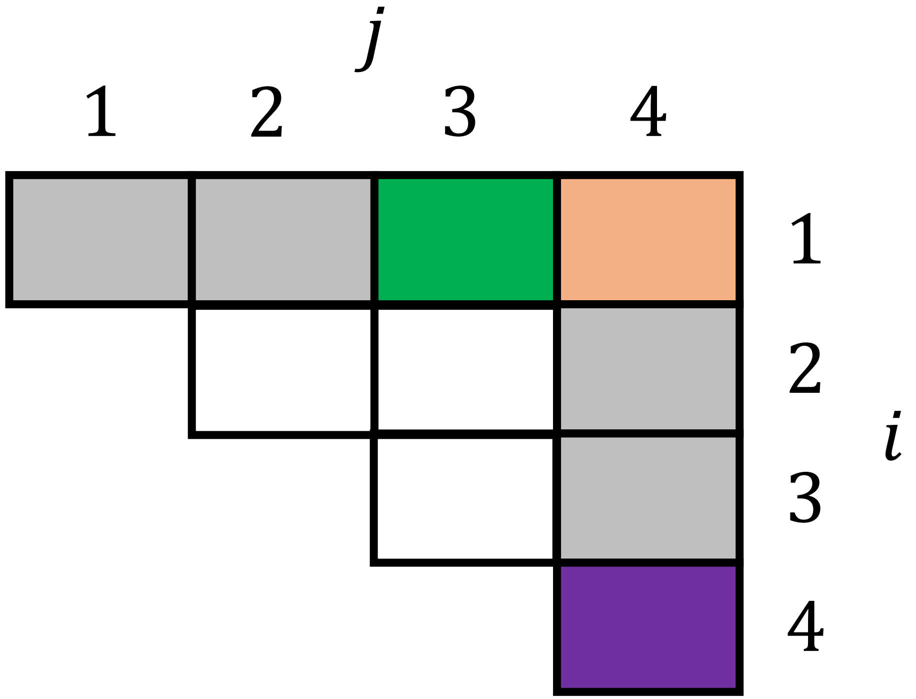
Select a Good Order for Solving Subproblems

$\mathrm{Best}(i,i) = 0$

$$\mathrm{Best}(i,j) = \min_{k=0,\dots,j-i-1} \mathrm{Best}(i, i+k) + \mathrm{Best}(i+k+1, j) + n_i n_{i+k+1} n_{j+1}$$

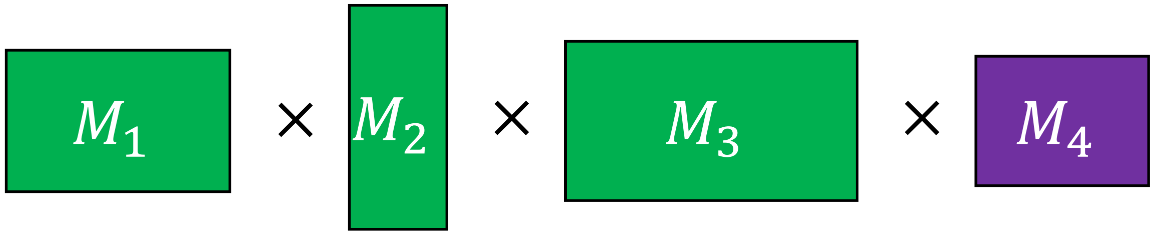
$i = 1, j = 4$

$k = 2$



$n_1 = 5$ $n_2 = 10$

$n_3 = 20$ $n_4 = 8$ $n_5 = 6$

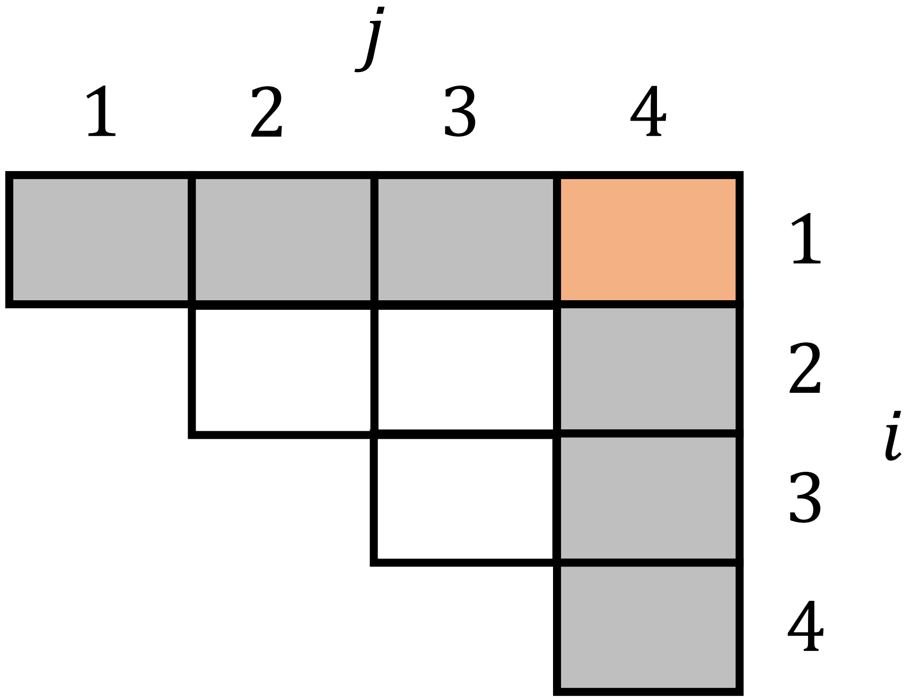


Select a Good Order for Solving Subproblems

$\text{Best}(i, i) = 0$

$$\text{Best}(i, j) = \min_{k=0,\ldots,j-i-1} \text{Best}(i, i+k) + \text{Best}(i+k+1, j) + n_i n_{i+k+1} n_{j+1}$$

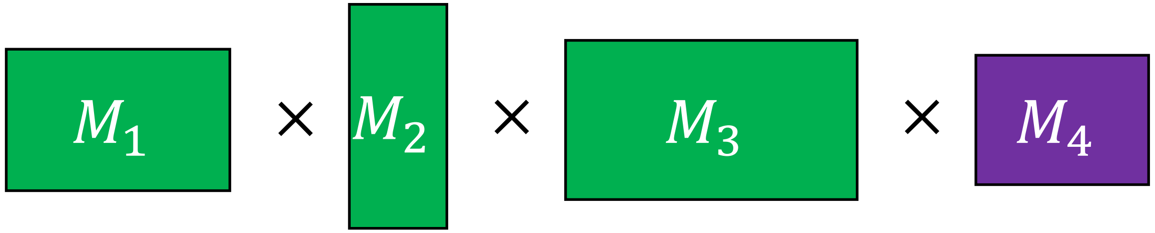
$i = 1, j = 4$



Observation: Value depends on values to its left and below

$n_1 = 5$ $\quad n_2 = 10$

$n_3 = 20$ $\quad n_4 = 8$ $\quad n_5 = 6$



Select a Good Order for Solving Subproblems

$\text{Best}(i,i) = 0$

$$\text{Best}(i,j) = \min_{k=0,\ldots,j-i-1} \text{Best}(i,i+k) + \text{Best}(i+k+1,j) + n_i n_{i+k+1} n_{j+1}$$

$i = 1, j = 4$

j

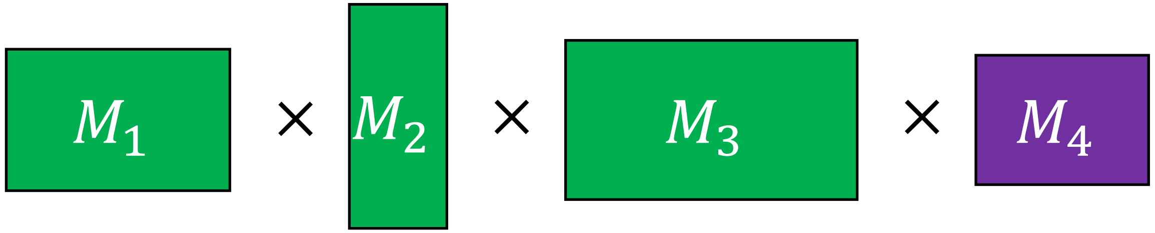
1	2	3	4	i
1	5	8	10	1
	2	6	9	2
		3	7	3
			4	4

Observation: Value depends on values to its left and below

Order: Fill values along diagonal

$n_1 = 5$ $\quad n_2 = 10$

$n_3 = 20$ $\quad n_4 = 8$ $\quad n_5 = 6$



Select a Good Order for Solving Subproblems

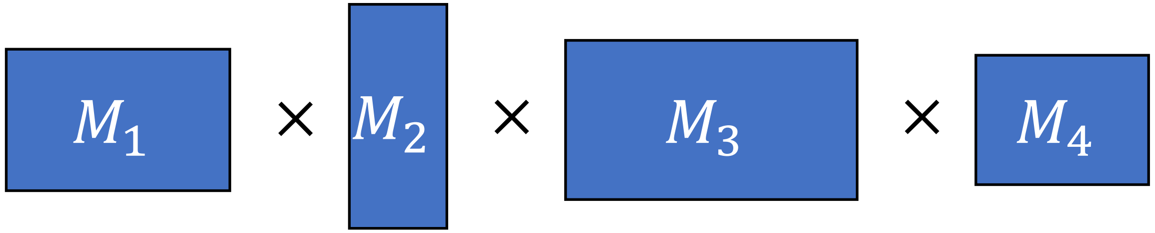
$\text{Best}(i,i) = 0 \quad \text{Best}(i,j) = \min_{k=0,\ldots,j-i-1} \text{Best}(i,i+k) + \text{Best}(i+k+1,j) + n_i n_{i+k+1} n_{j+1}$

j

1	2	3	4	i
0	1000	1800	2040	1
	0	1600	2080	2
		0	960	3
			0	4

$n_1 = 5 \qquad n_2 = 10$

$n_3 = 20 \qquad n_4 = 8 \qquad n_5 = 6$



$M_1 \times M_2 \times M_3 \times M_4$

Run Time

1. Initialize $\text{Best}[i, i]$ to be all 0s
2. Starting at the main diagonal, working to the upper-right, fill in each cell using:
   - $\text{Best}(i, i) = 0$
   - $\text{Best}(i, j) = \min_{k=0,\dots,j-i-1} \text{Best}(i, i+k) + \text{Best}(i+k+1, j) + n_i n_{i+k+1} n_{j+1}$

$\Theta(n^2)$ cells in the array

$\Theta(n)$ options per cell

$\Theta(n^3)$ overall run time

Backtrack to Find the Best Order

$\text{Best}(i,i) = 0 \quad \text{Best}(i,j) = \min_{k=0,\ldots,j-i-1} \text{Best}(i,i+k) + \text{Best}(i+k+1,j) + n_i n_{i+k+1} n_{j+1}$

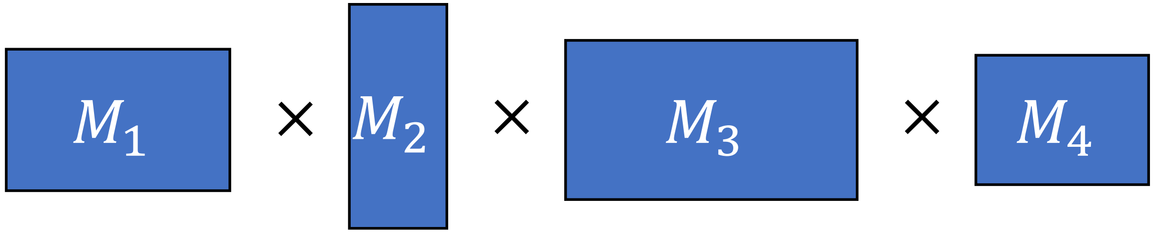
j

1	2	3	4	i
0	1000 ⓪	1800 ①	2040 ②	1
	0	1600 ⓪	2080 ①	2
		0	960 ⓪	3
			0	4

"remember" which choice of k was the minimum at each cell

$n_1 = 5 \quad n_2 = 10$

$n_3 = 20 \quad n_4 = 8 \quad n_5 = 6$



Longest Common Subsequence

Given two sequences X and Y, find the length of their longest common <u>subsequence</u>

Example:

$X =$ ATCTGAT

$Y =$ TGCATA

Longest Common Subsequence

Given two sequences X and Y, find the length of their longest common subsequence

Example:

$X =$ ATCTGAT

$Y =$ TGCATA

LCS $=$ TCTA

Brute force: Compare every subsequence of X with Y

Running Time: $\Omega(2^n)$

Dynamic Programming

Requires optimal substructure

- Solution to larger problem contains the solutions to smaller ones

General Blueprint:

1. Identify recursive structure of the problem
   - What is the “last thing” done?
2. Select a good order for solving subproblems
   - “Top Down:” Solve each problem recursively
   - “Bottom Up:” Iteratively solve each problem from smallest to largest
3. Save solution to each subproblem in memory

Identify Recursive Structure

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
T	1	**0**							
G	2	**0**							
C	3	**0**							
A	4	**0**							
T	5	**0**							
A	6	**0**							

$i = 2$ and $j = 2$

Identify Recursive Structure

Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
T	1	**0**							
G	2	**0**					■		
C	3	**0**							
A	4	**0**							
T	5	**0**							
A	6	**0**							

$i = 2$ and $j = 5$

Identify Recursive Structure

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

Suppose $X[i] = Y[j]$ $\quad$ $i = 2$ and $j = 1$

$X =$ ATCTGAT

$Y =$ TGCATA

Observation: We can always include the last character (T) in the LCS

Why is this the case? (Argument for optimality)

- If last character in LCS is not T, then can extend it to include T
- If the last character in LCS is T, then it does not matter whether we use an earlier T or the last T

Identify Recursive Structure

Let $\mathrm{LCS}(i,j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

Suppose $X[i] = Y[j]$ $\quad i = 2$ and $j = 1$

$X =$ `ATCTGAT`

$Y =$ `TGCATA`

Observation: We can always include the last character (`T`) in the LCS

Optimal choice: always take the last character (add it to the LCS), and recursively solve LCS on remainder

$$\mathrm{LCS}(i,j) = \mathrm{LCS}(i-1, j-1) + 1$$

Identify Recursive Structure

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

Suppose $X[i] \neq Y[j]$ $\quad$ $i = 2$ and $j = 2$

$$X = \text{ATCTGAT}$$
$$Y = \text{TGCATA}$$

Observation: At least one of the characters will not be in the LCS

Why is this the case? (Argument for optimality)

- Cannot take both, since otherwise, the last character of the two subsequences are different

Identify Recursive Structure

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

Suppose $X[i] \neq Y[j]$ $i = 2$ and $j = 2$

$X =$ ATCTGAT

$Y =$ TGCATA

Observation: At least one of the characters will not be in the LCS

$$\mathrm{LCS}(i, j) = \mathrm{LCS}(i - 1, j)$$

Drop T

or

$$\mathrm{LCS}(i, j) = \mathrm{LCS}(i, j - 1)$$

Drop G

Identify Recursive Structure

Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

Suppose $X[i] \neq Y[j]$ $\quad$ $i = 2$ and $j = 2$

$$X = \texttt{ATCTGAT}$$
$$Y = \texttt{TGCATA}$$

Observation: At least one of the characters will not be in the LCS

$$\text{LCS}(i, j) = \max\big(\text{LCS}(i - 1, j), \text{LCS}(i, j - 1)\big)$$

Identify Recursive Structure

Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

Suppose $X[i] = Y[j]$

$$\text{LCS}(i, j) = \text{LCS}(i - 1, j - 1) + 1$$

Suppose $X[i] \neq Y[j]$

$$\text{LCS}(i, j) = \max\big(\text{LCS}(i - 1, j), \text{LCS}(i, j - 1)\big)$$

Base case:

$$\text{LCS}(i, 0) = 0 = \text{LCS}(0, j)$$

Identify Recursive Structure

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

$$\mathrm{LCS}(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \mathrm{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\mathrm{LCS}(i-1, j), \mathrm{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Dynamic Programming

Requires optimal substructure

- Solution to larger problem contains the solutions to smaller ones

General Blueprint:

1. Identify recursive structure of the problem
    - What is the “last thing” done?
2. Select a good order for solving subproblems
    - “Top Down:” Solve each problem recursively
    - “Bottom Up:” Iteratively solve each problem from smallest to largest
3. Save solution to each subproblem in memory

Select a Good Order for Solving Subproblems

Let $\mathrm{LCS}(i,j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

$Y =$	$X =$		A	T	C	T	G	A	T
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0							
G	2	0							
C	3	0							
A	4	0							
T	5	0							
A	6	0							

$$\mathrm{LCS}(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \mathrm{LCS}(i-1,j-1)+1 & X[i] = Y[j] \\ \max\big(\mathrm{LCS}(i-1,j), \mathrm{LCS}(i,j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Select a Good Order for Solving Subproblems

Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	🟧						
G	2	0		🟦					
C	3	0							
A	4	0							
T	5	0							
A	6	0							

$$\text{LCS}(i, j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \text{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\text{LCS}(i-1, j), \text{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Select a Good Order for Solving Subproblems

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	🟧						
G	2	0	🟩	🟦					
C	3	0							
A	4	0							
T	5	0							
A	6	0							

$$\mathrm{LCS}(i, j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \mathrm{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\mathrm{LCS}(i-1, j), \mathrm{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Select a Good Order for Solving Subproblems

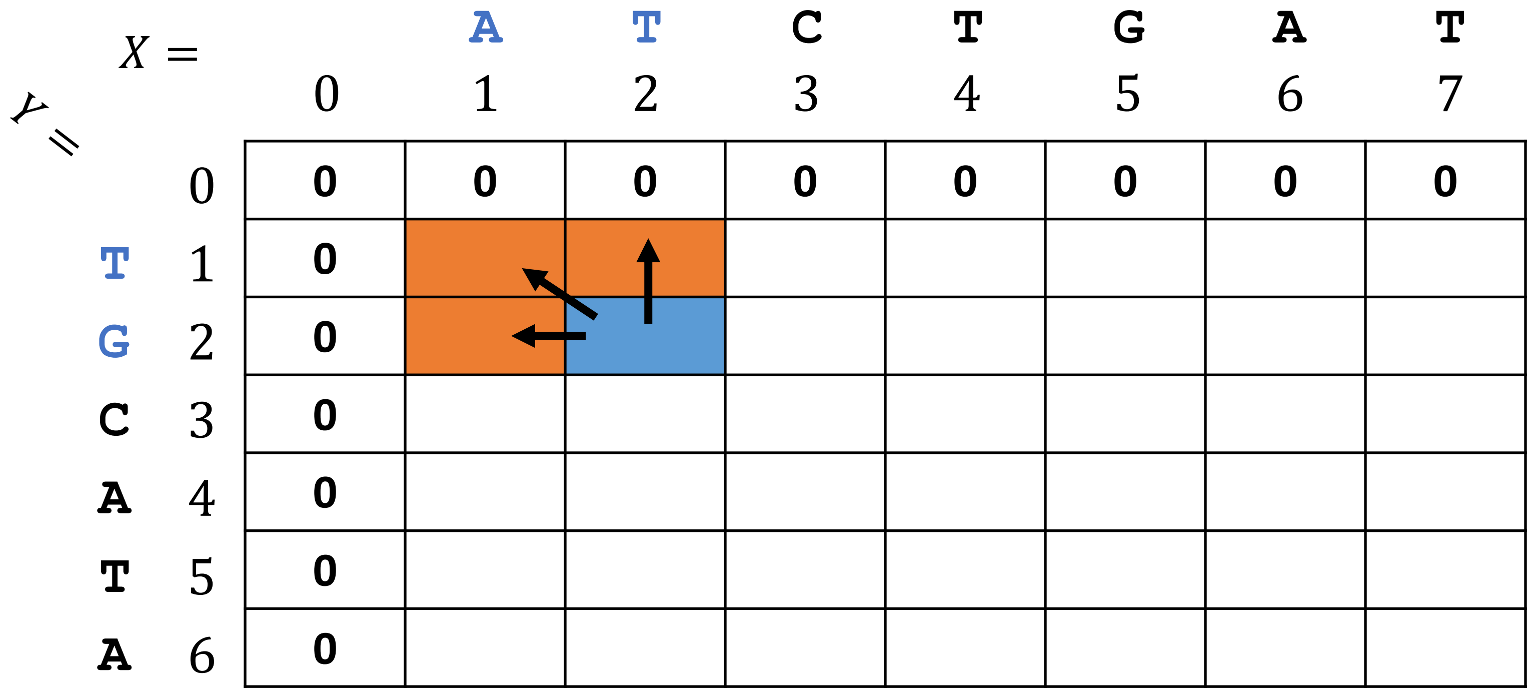
Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
T	1	**0**	🟧	🟨					
G	2	**0**	🟩	🟦					
C	3	**0**							
A	4	**0**							
T	5	**0**							
A	6	**0**							

$$\text{LCS}(i, j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \text{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\text{LCS}(i-1, j), \text{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Select a Good Order for Solving Subproblems

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y



$Y =$	$X =$		A	T	C	T	G	A	T
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0							
G	2	0							
C	3	0							
A	4	0							
T	5	0							
A	6	0							

$$\mathrm{LCS}(i, j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \mathrm{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\mathrm{LCS}(i-1, j), \mathrm{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Value depends on values to the left and above

Fill rows top to bottom, left to right

Select a Good Order for Solving Subproblems

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

$Y =$	$X =$		A	T	C	T	G	A	T
		0	1	2	3	4	5	6	7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
T	1	**0**	**0**						
G	2	**0**							
C	3	**0**							
A	4	**0**							
T	5	**0**							
A	6	**0**							

$$\mathrm{LCS}(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \mathrm{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\mathrm{LCS}(i-1, j), \mathrm{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Select a Good Order for Solving Subproblems

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
T	1	**0**	**0**	**1**					
G	2	**0**							
C	3	**0**							
A	4	**0**							
T	5	**0**							
A	6	**0**							

$$\mathrm{LCS}(i, j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \mathrm{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\mathrm{LCS}(i-1, j), \mathrm{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Select a Good Order for Solving Subproblems

Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

$Y =$	$X =$		A	T	C	T	G	A	T
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1				
G	2	0							
C	3	0							
A	4	0							
T	5	0							
A	6	0							

$$\text{LCS}(i, j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \text{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\text{LCS}(i-1, j), \text{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Select a Good Order for Solving Subproblems

Let $\text{LCS}(i,j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		**A**	**T**	**C**	**T**	**G**	**A**	**T**
$Y =$		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

$$\text{LCS}(i,j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \text{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\text{LCS}(i-1, j), \text{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Run Time: $\Theta(n \cdot m)$
(for $|X| = n$, $|Y| = m$)

Backtrack to Find the LCS

Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

$X =$			A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

$$\text{LCS}(i, j) = \begin{cases} 0 & i = 0 \text{ or } j = 0 \\ \text{LCS}(i-1, j-1) + 1 & X[i] = Y[j] \\ \max\big(\text{LCS}(i-1, j), \text{LCS}(i, j-1)\big) & X[i] \neq Y[j] \end{cases}$$

Length of LCS is 4

How did we get here?

Backtrack to Find the LCS

Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

$Y =$	$X =$		A	T	C	T	G	A	T
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Length of LCS is 4

How did we get here?

Take the character and move diagonally up if characters match

Otherwise, move to the larger of the value above or to the left

Backtrack to Find the LCS

Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

$Y =$	$X =$		A	T	C	T	G	A	T
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Take the character and move diagonally up if characters match

Otherwise, move to the larger of the value above or to the left

Backtrack to Find the LCS

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

$Y =$	$X =$	0	A 1	T 2	C 3	T 4	G 5	A 6	T 7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Take the character and move diagonally up if characters match

Otherwise, move to the larger of the value above or to the left

Backtrack to Find the LCS

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

$Y =$	$X =$		A	T	C	T	G	A	T
		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Take the character and move diagonally up if characters match

Otherwise, move to the larger of the value above or to the left

Backtrack to Find the LCS

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Take the character and move diagonally up if characters match

Otherwise, move to the larger of the value above or to the left

Backtrack to Find the LCS

Let $\mathrm{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Take the character and move diagonally up if characters match

Otherwise, move to the larger of the value above or to the left

Backtrack to Find the LCS

Let $\text{LCS}(i, j)$ denote the length of the longest common subsequence between the first i characters of X and first j character of Y

	$X =$		A	T	C	T	G	A	T
$Y =$		0	1	2	3	4	5	6	7
	0	0	0	0	0	0	0	0	0
T	1	0	0	1	1	1	1	1	1
G	2	0	0	1	1	1	2	2	2
C	3	0	0	1	2	2	2	2	2
A	4	0	1	1	2	2	2	3	3
T	5	0	1	2	2	3	3	3	4
A	6	0	1	2	2	3	3	4	4

Take the character and move diagonally up if characters match

Otherwise, move to the larger of the value above or to the left

Not necessarily unique!

Seam Carving

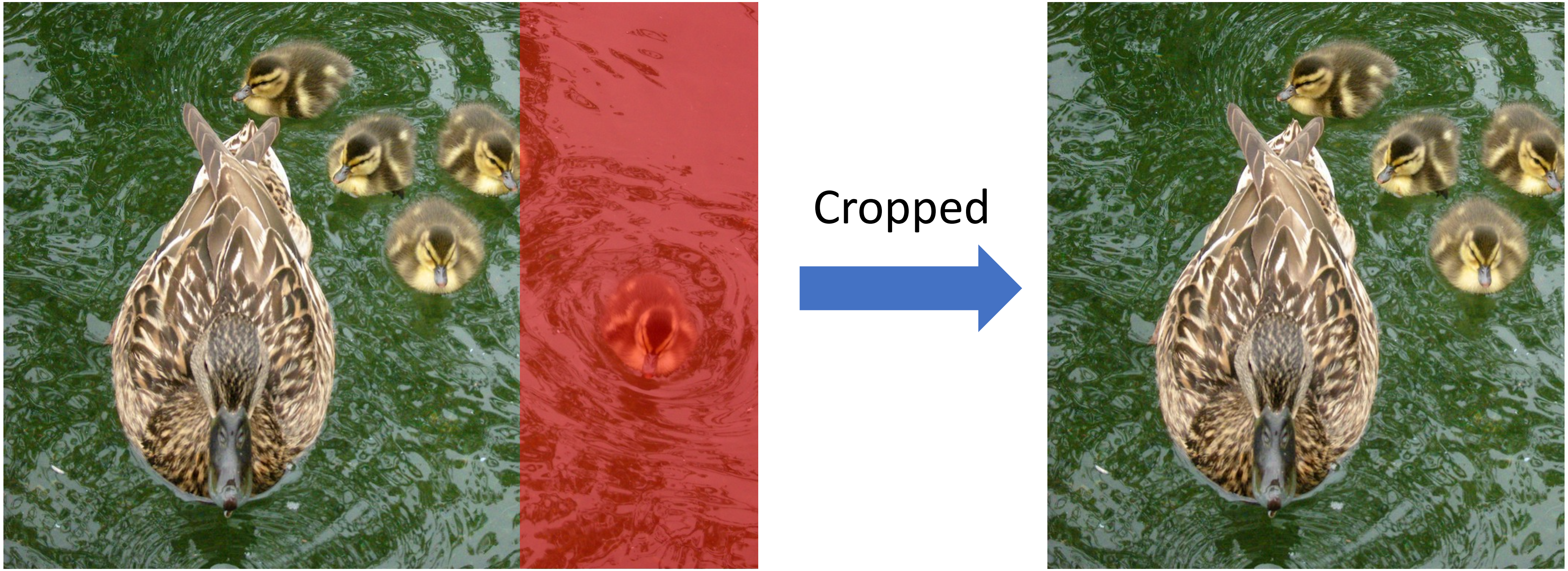
Method for image resizing that does not scale/crop the image

Seam Carving

Method for image resizing that does not scale/crop the image

Cropping

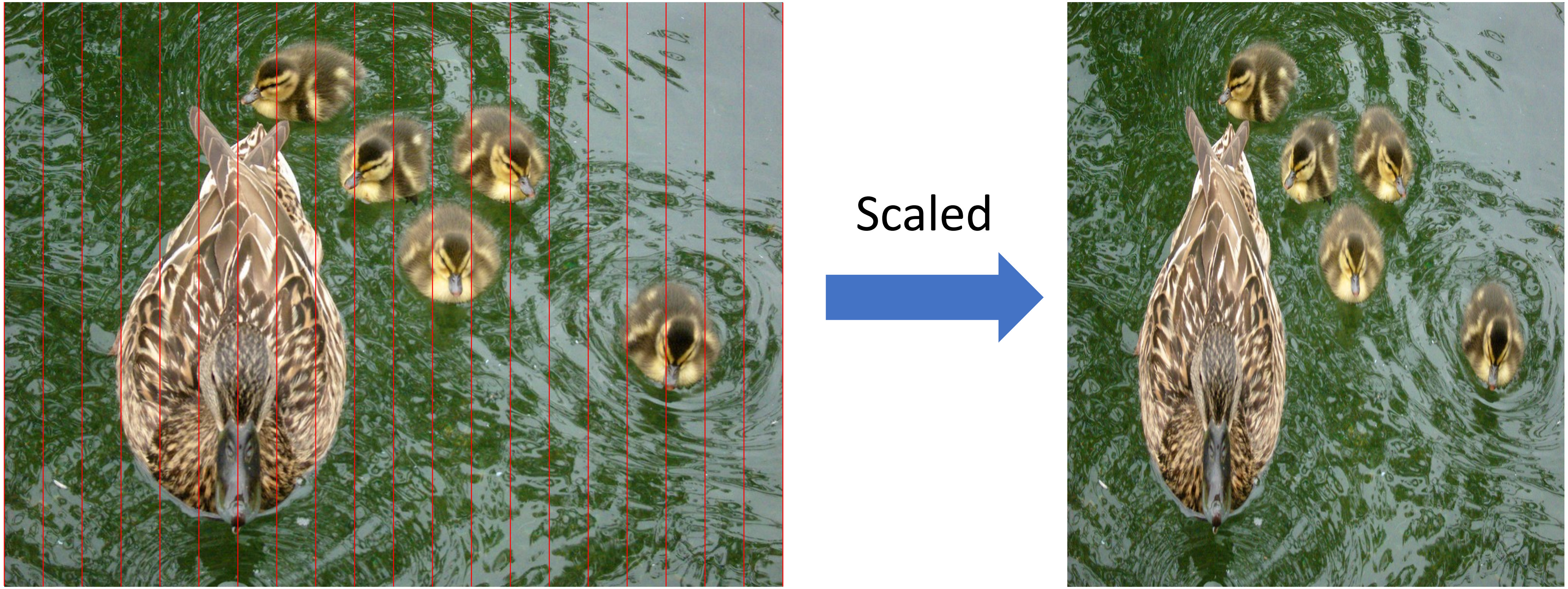
Removes a “block” of pixels



Cropped

Scaling

Removes “stripes” of pixels

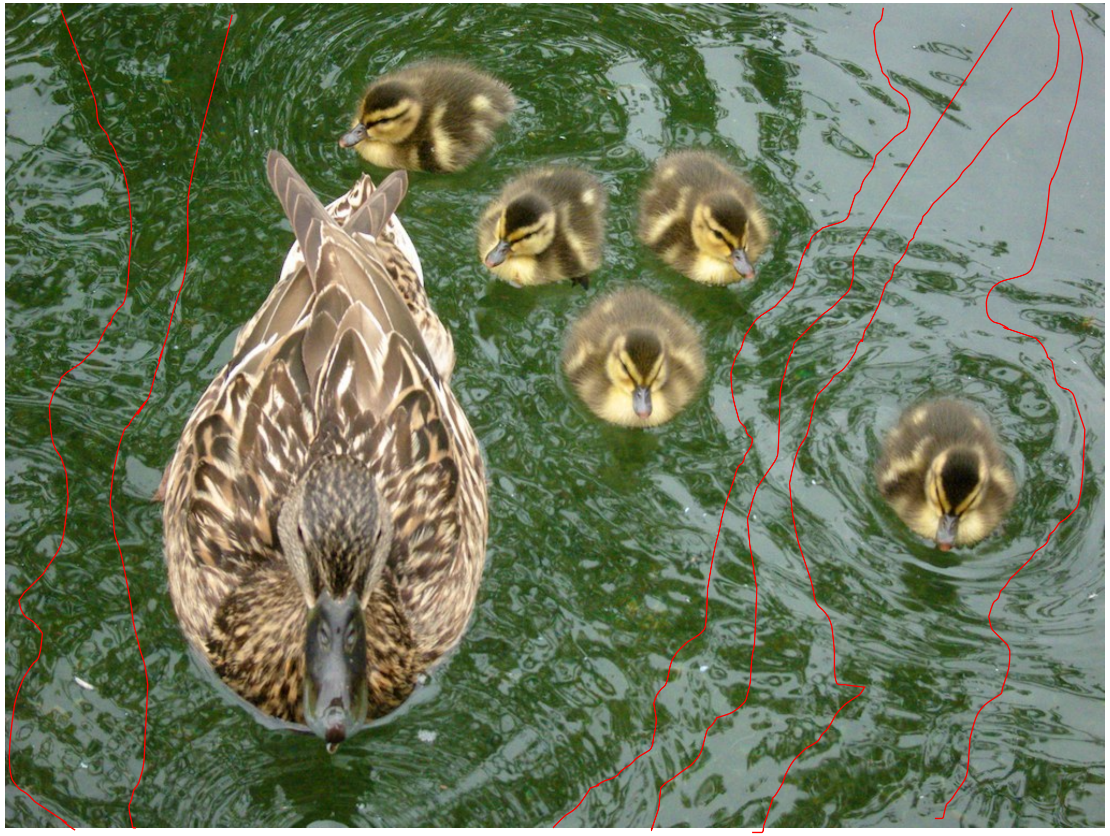


Scaled

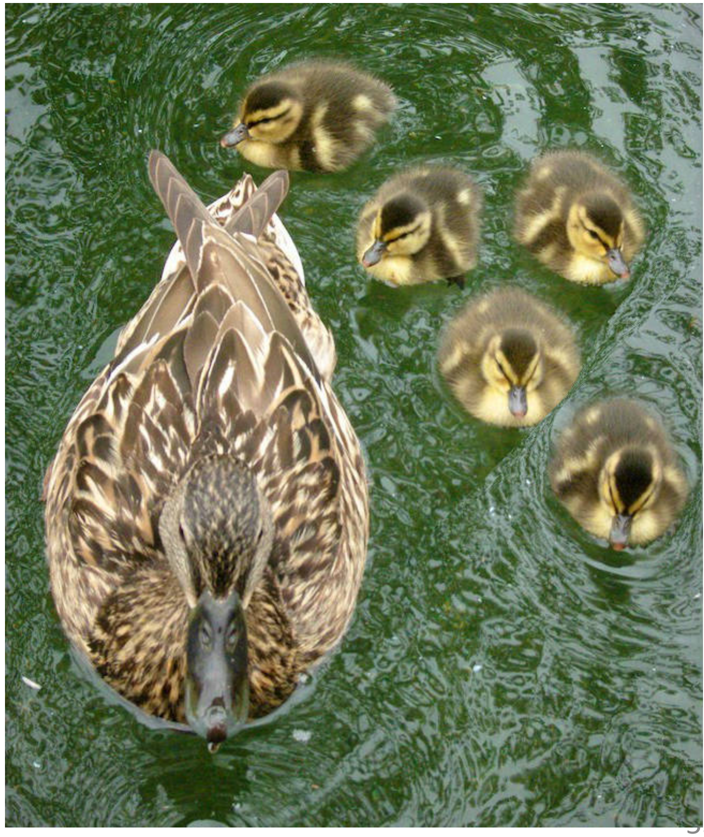
Seam Carving

Removes “least energy seam” of pixels

Demo: http://nparashuram.com/seamcarving/



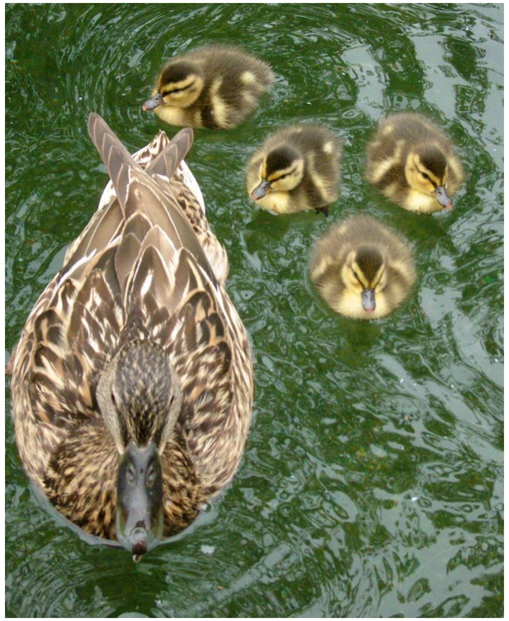
Carved



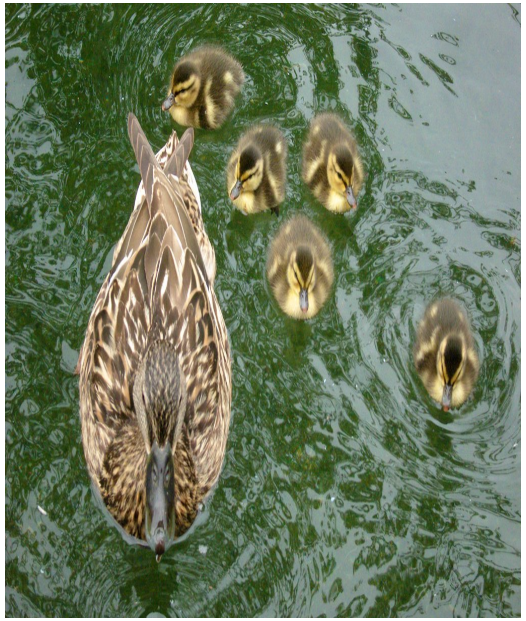
Seam Carving

Method for image resizing that does not scale/crop the image

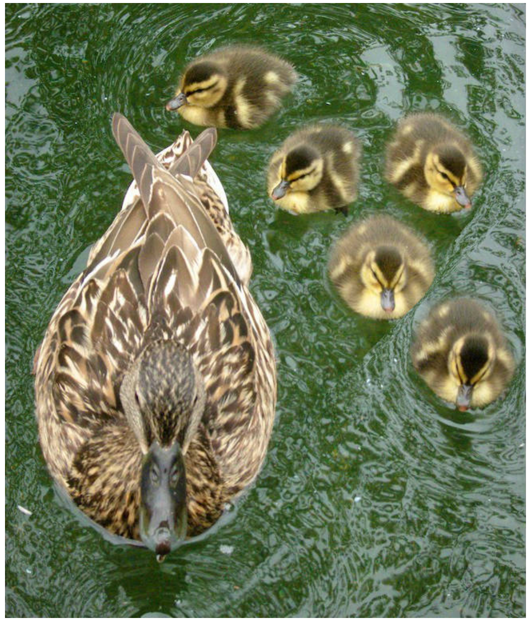
Cropped



Scaled



Carved



Seattle Skyline

Demo: http://nparashuram.com/seamcarving/

Energy of a Seam

Sum of the energies of each pixel

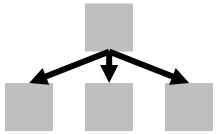
- $e(p) =$ energy of pixel p

Many choices

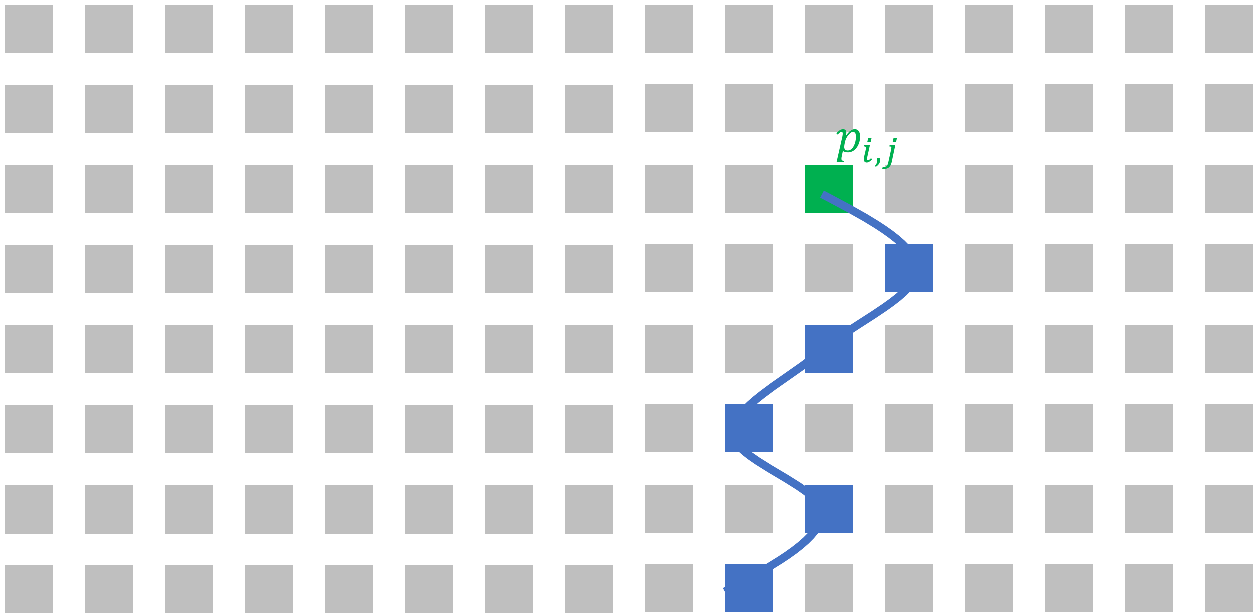
- **Example:** Gradient (how much the color of this pixel differs from its neighbors)
- Particular choice doesn’t matter, we use it as a “black box”

Seam Carving

$S(i,j)$ = seam with minimal energy from the bottom of the image to pixel $p_{i,j}$



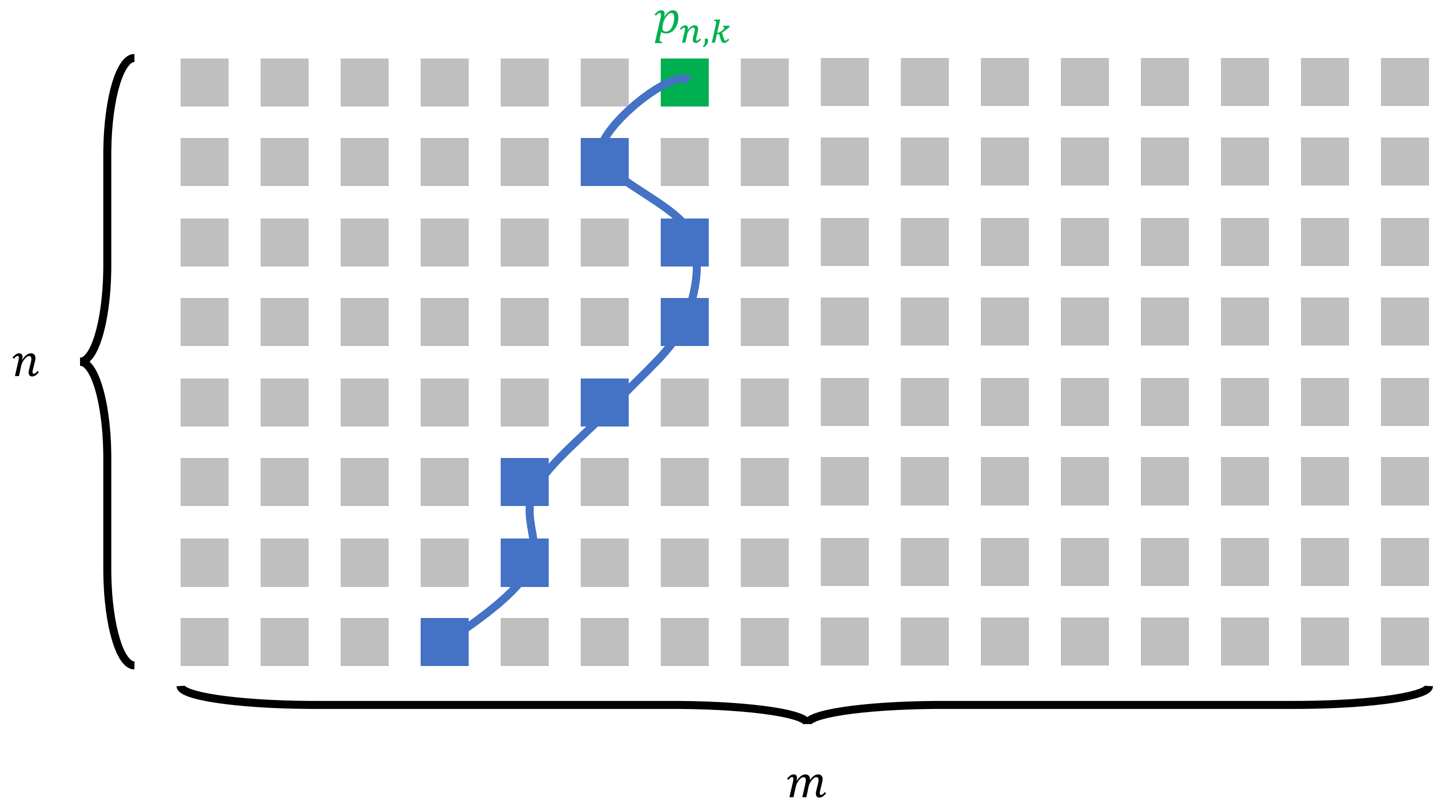
Seam extends from one pixel to (diagonally) adjacent pixel on next row



Seam Carving

Goal: find the least energy seam going from bottom to top, so delete:

$$\min_{k=1,\ldots,m} \left(S(n,k)\right)$$



Dynamic Programming

Requires optimal substructure

- Solution to larger problem contains the solutions to smaller ones

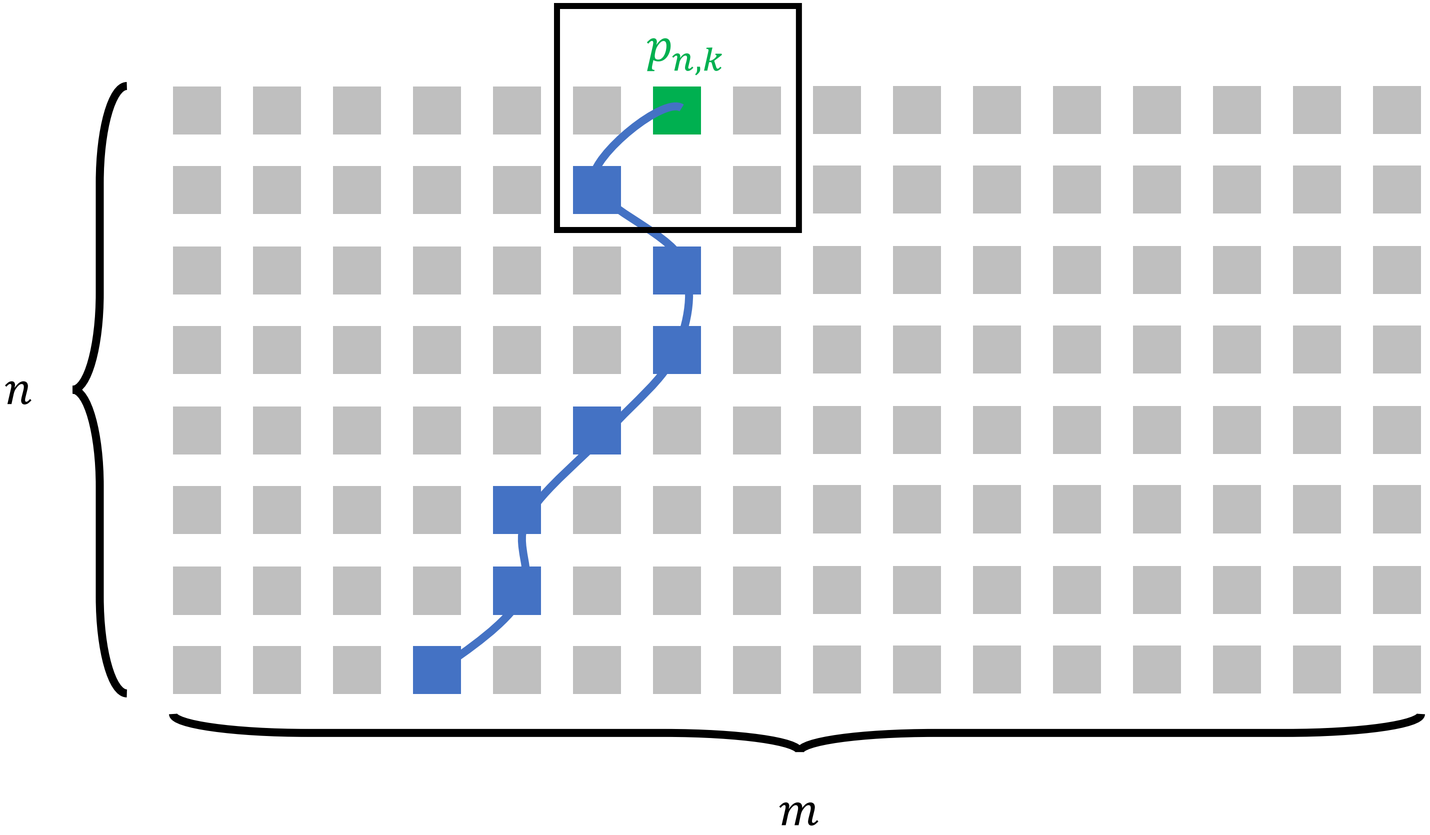
General Blueprint:

1. Identify recursive structure of the problem
    - What is the "last thing" done?
2. Select a good order for solving subproblems
    - "Top Down:" Solve each problem recursively
    - "Bottom Up:" Iteratively solve each problem from smallest to largest
3. Save solution to each subproblem in memory

Computing $S(n,k)$

Suppose we know the least energy seams for all rows up to $n-1$

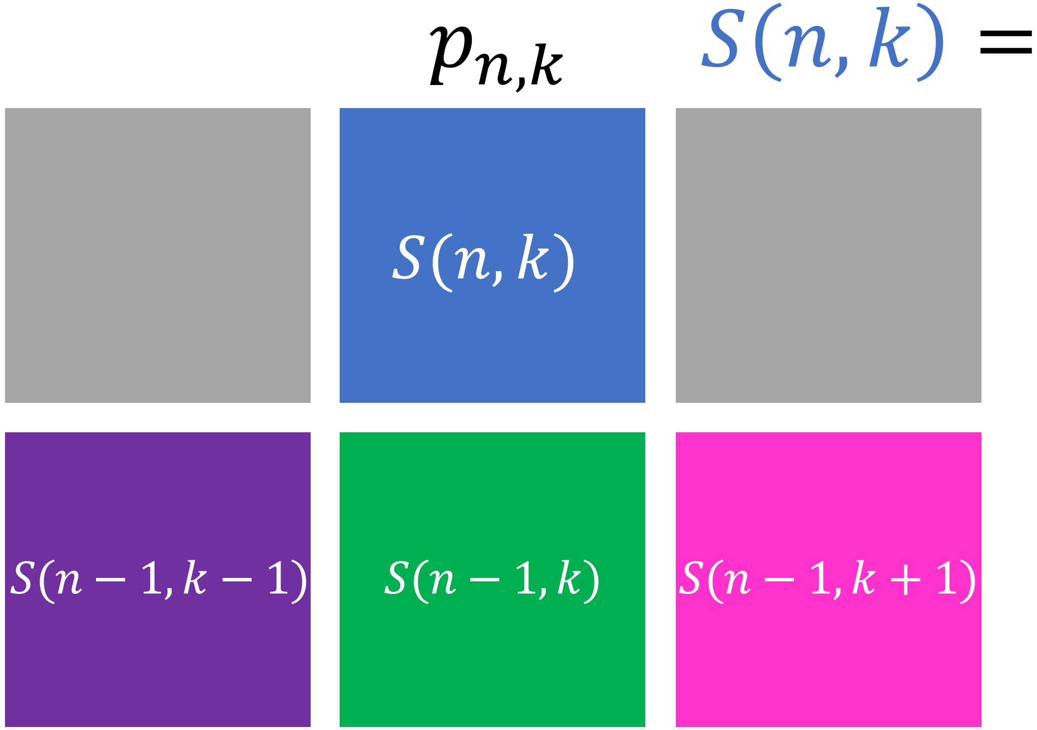
(i.e., we know $S(n-1,\ell)$ for all ℓ)



Computing $S(n, k)$

Suppose we know the least energy seams for all rows up to $n - 1$

(i.e., we know $S(n-1, \ell)$ for all ℓ)



$p_{n,k}$

$S(n, k)$

$S(n-1, k-1)$

$S(n-1, k)$

$S(n-1, k+1)$

$$S(n, k) = \min \begin{cases} S(n-1, k-1) + e(p_{n,k}) \\ S(n-1, k) + e(p_{n,k}) \\ S(n-1, k+1) + e(p_{n,k}) \end{cases}$$

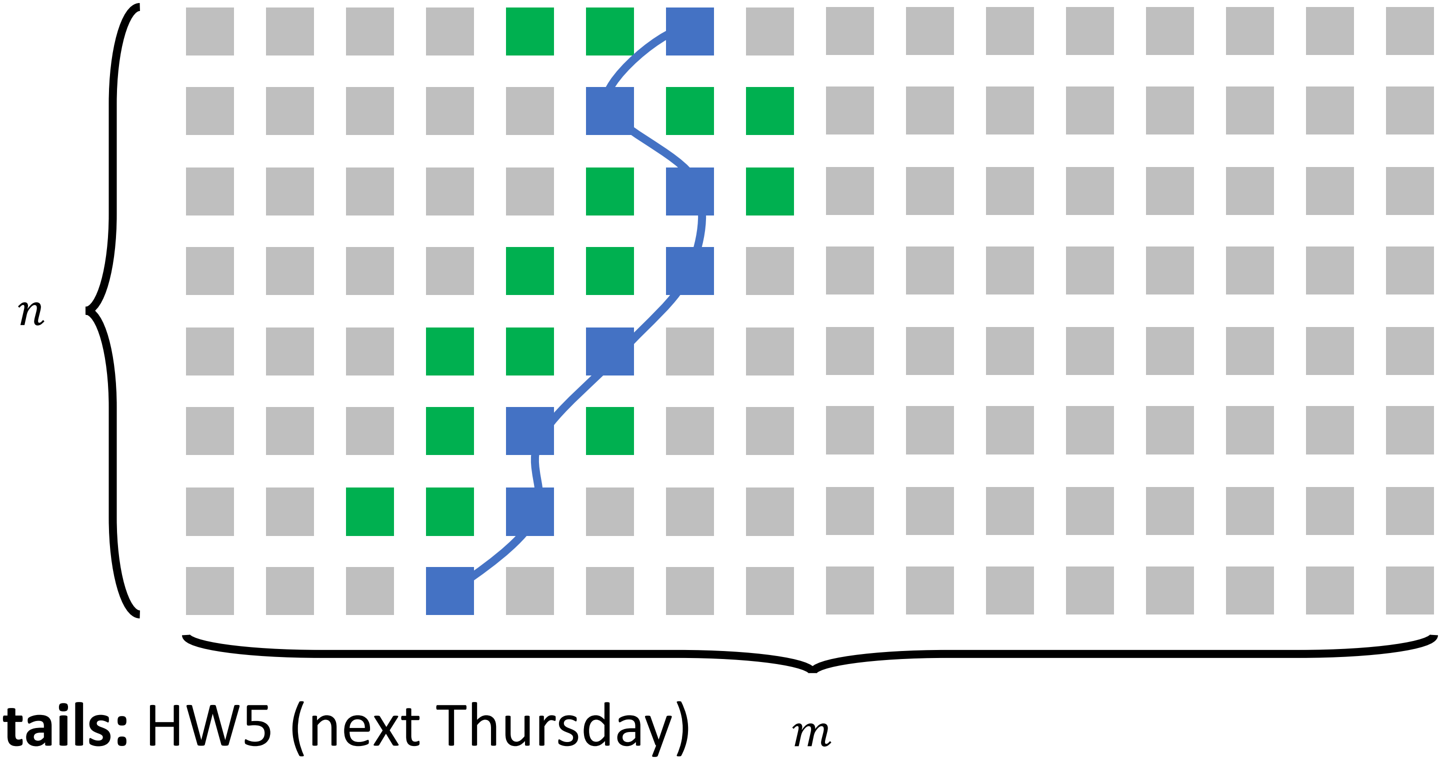
Repeated Seam Removal

Only need to update pixels that depend on the removed seam

At most $2n$ pixels change

$\Theta(n)$ time to update pixels

$\Theta(n+m)$ time to find minimum + backtrack



Full details: HW5 (next Thursday)