Intro to CS16

CS16: Introduction to Algorithms & Data Structures
Summer 2021

Welcome to CS16!

Join Prismia at https://bit.ly/3tEJxTB

Feel free to turn your camera on (but you don't have to)

Eid Mubarak!

Doug Woos

he/him/his

Call me Doug, Professor Woos, etc.

Meet the TAs!

- They are great

Data structures

Algorithms

A problem

- We have a collection of **tasks** we want to accomplish
  - Each task has a **priority** (1, 2, etc.)
  - Multiple tasks can have the same priority
  - Tasks with lower priority numbers need to be done first
- In what **order** should I do these tasks?
- Example: email inbox
  - Question from a colleague about a paper (priority 2)
  - Urgent message from TAs about class (priority 1)
  - Good deal on a used banjo (priority 3)

Data structures

how should our data be organized?

Algorithms

how should we use our organized data to solve the problem?

A problem

- We have a collection of **tasks** we want to accomplish
  - Some tasks depend on other tasks
  - Some are independent
- In what **order** should I do these tasks?
- Example: I make really good burritos
  - Need to chop an onion before sautéing it
  - But, can sauté onion and cook rice simultaneously
  - BAD: sauté onions, chop onions, cook rice
  - GOOD: chop onions, cook rice, sauté onions

Data structures

how should our data be organized?

Algorithms

how should we use our organized data to solve the problem?

Another example: PageRank

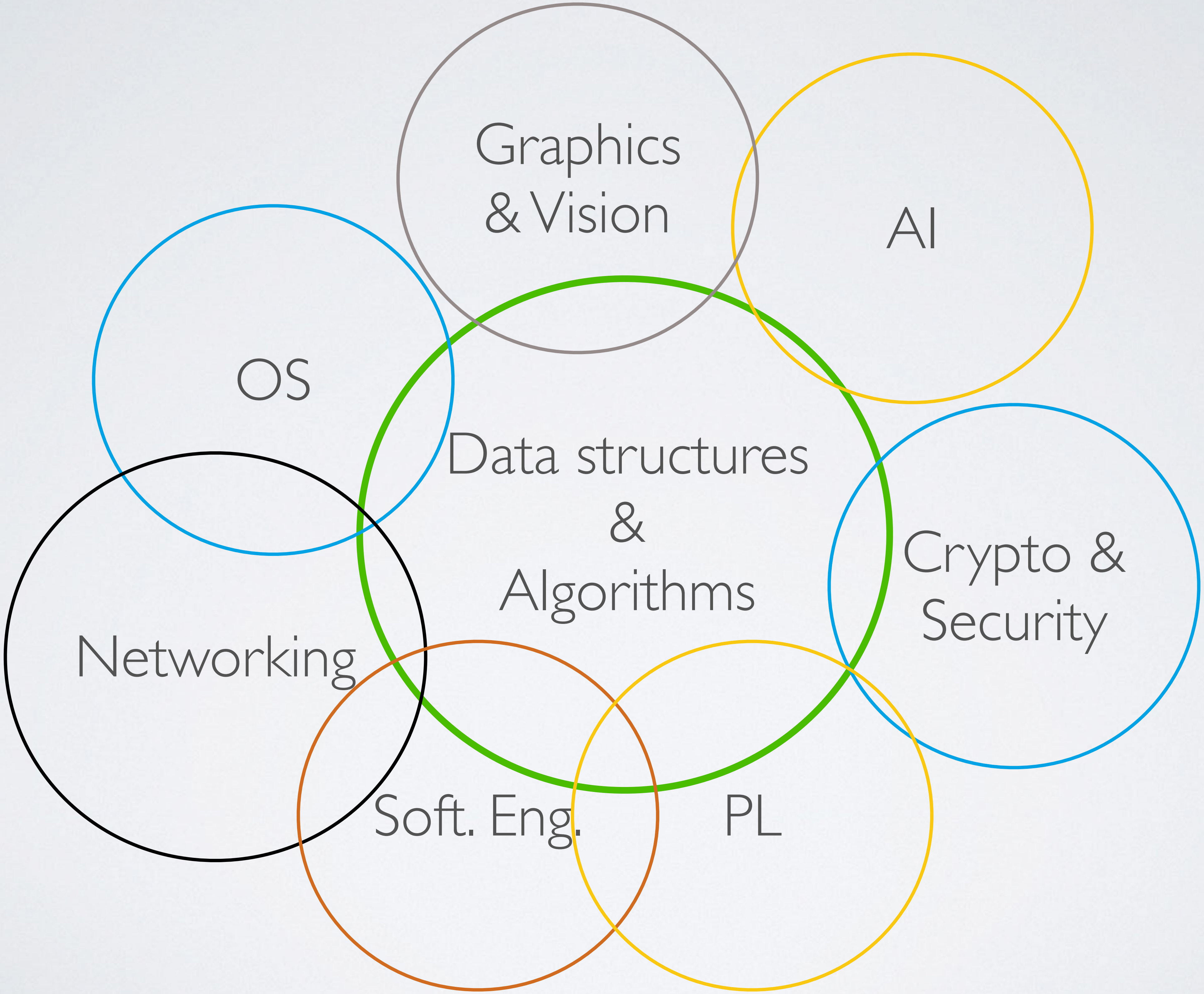
- Before 1999
  - search engines ranked pages using keyword frequency
  - well-known and worked OK
- Larry Page & Sergey Brin (PhD students @ Stanford)
  - noticed that links were important too!
  - links convey information about importance
  - But what exactly? and how can you make use of it?
  - This lead them to design PageRank

CS16 topics

- Implementing data structures and algorithms
- Analyzing data structures and algorithms
- Designing data structures and algorithms

Analysis: what makes an algorithm "good?"

CS is diverse



How CS16 works
(briefly)

Course Page

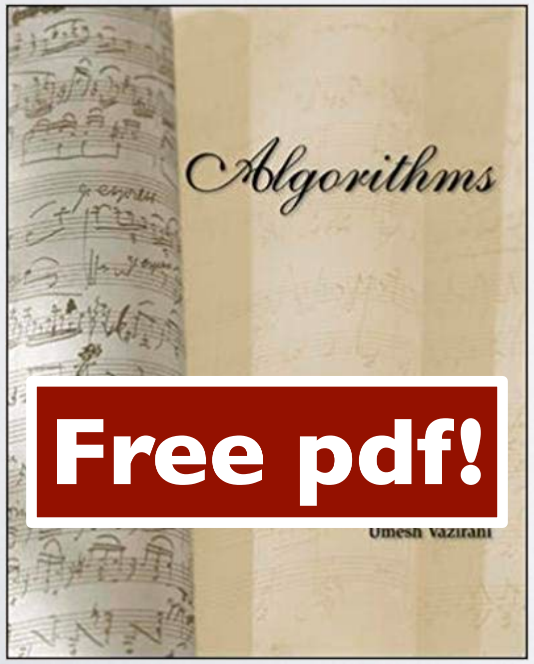
- Missive & Policies
- Slides
- Lecture capture
- Announcements
- Helpful Documents
  - Java, Latex & Python tips
  - Guides for testing, readmes, working from home, …

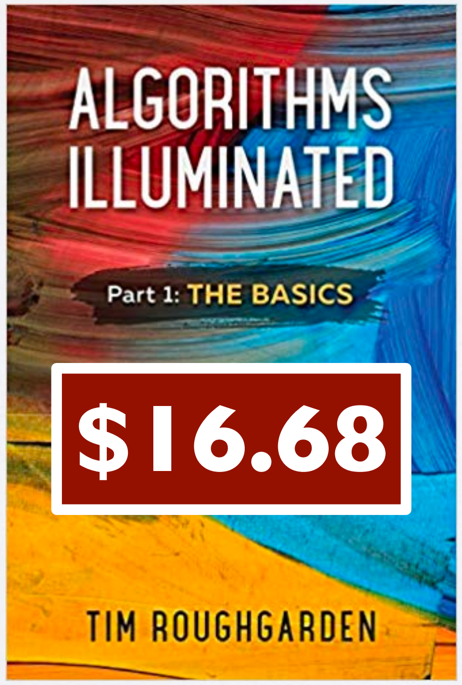
Lectures

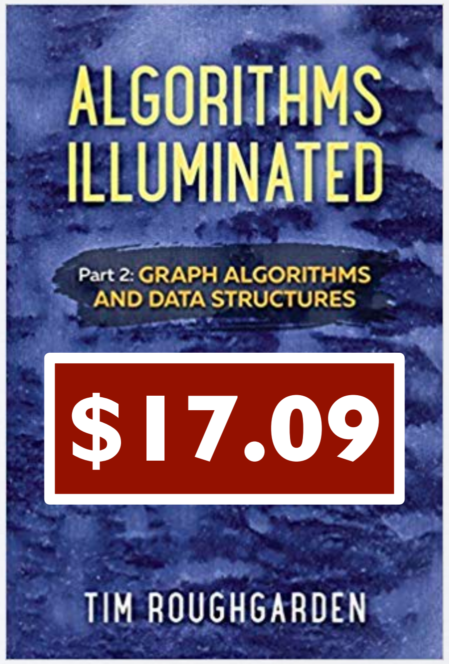
- Cover various algorithms & data structures
  - How they work
  - Why they work
  - Analysis
- Activities & discussions
- You are responsible for content in lecture (whether on slides or not)

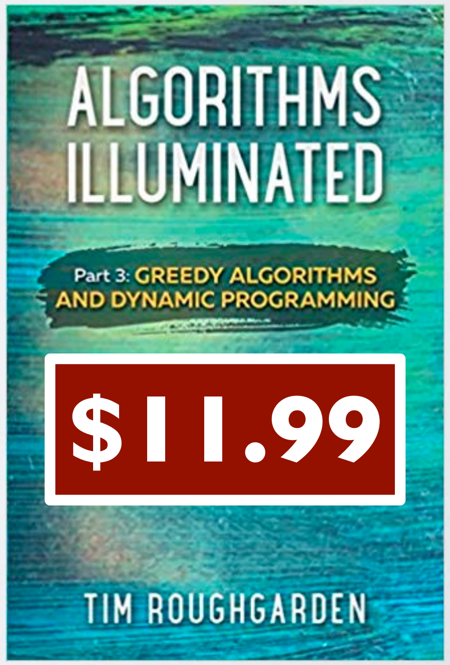
Textbook

- No required textbook
- Helpful
  - Algorithms by Dasgupta, Papadimtriou and Vazirani
  - Algorithms Illuminated 1, 2 & 3 by Roughgarden









Ed

- Announcements
- Questions and answers
- Links to helpful material (blogs, Youtube videos)

Sections

- `1` hour/week with TAs
- `6-10` students
- Required!
- Mini assignments
- Mentor

Office Hours

- TA hours are very helpful
  - Try to get unstuck on your own first
  - TAs will ask you what you tried…
    - … and send you back if you didn't try anything
- Doug's hours: Tuesdays 2:30-4:30 on Zoom
  - Open Zoom call
  - Come with conceptual questions, career questions, study/debugging skills questions, etc.
  - Also available by appointment
- Questions about HW or projects:
  - Post on Ed
  - Ask in Section
  - Hours

Assignments

- Homeworks
  - Due every(ish) week
  - Python code, proofs, analysis, …
- Projects
  - 4 over the whole semester
  - Larger-scale Java programming
- Online midterm and final

Email Policy

- Unless matter is private always email HTAs!
  - Your email can get lost in Doug's inbox
  - It may take me a while to get to your email
  - HTAs may get to it faster & will remind me

Seam carving

our first algorithm!

Why seam carving?

- A cool algorithm with interesting applications
- Leads us into analysis of algorithms in general
  - We'll develop the tools we need over the next several lectures
  - For now, just try to understand what it's doing and why it works!


MAHOPAC INDIANS

The New York Times

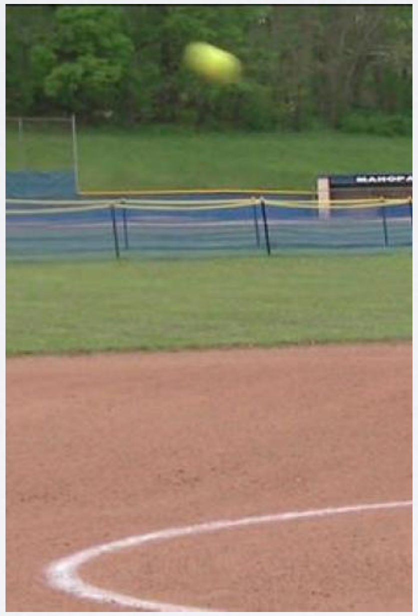
The New York Times

Image Resizing

- Preserve important elements
- Remove/reduce repetitive areas

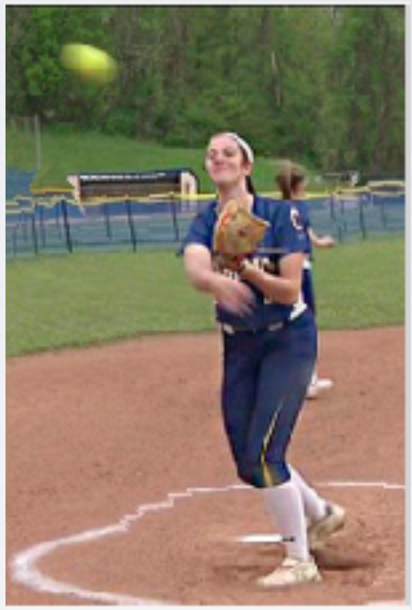
Image Resizing

Fail



Fail

Fail



Success

Image Resizing

- To shrink image
  - remove unimportant pixels
- Quantify pixel *importance*
  - How much it varies from neighbors
  - Sum of differences with horizontal & vertical neighbors

Image Resizing

- Grayscale 3x3 image with the following pixel intensities
- Importance of the center pixel?

4	**6**	**5**
2	**5**	**7**
3	**2**	**6**

Image Resizing

- Quantify importance of every pixel
- Determine most and least important pixels

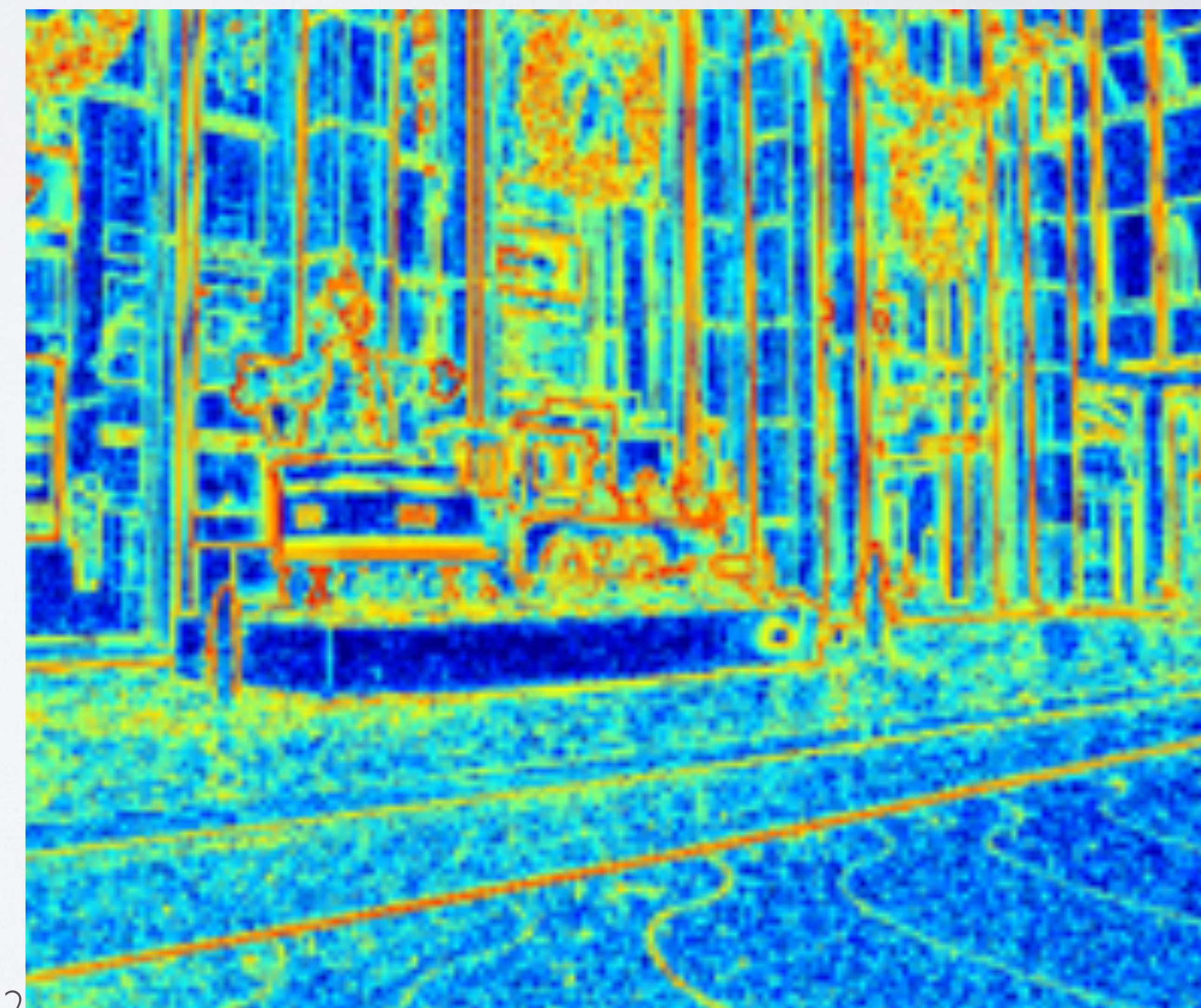


Image Resizing: Approach 1

- Remove all pixels with importance below some threshold
- Problem?
  - removing different # of pixels from each row
  - causes jagged right side

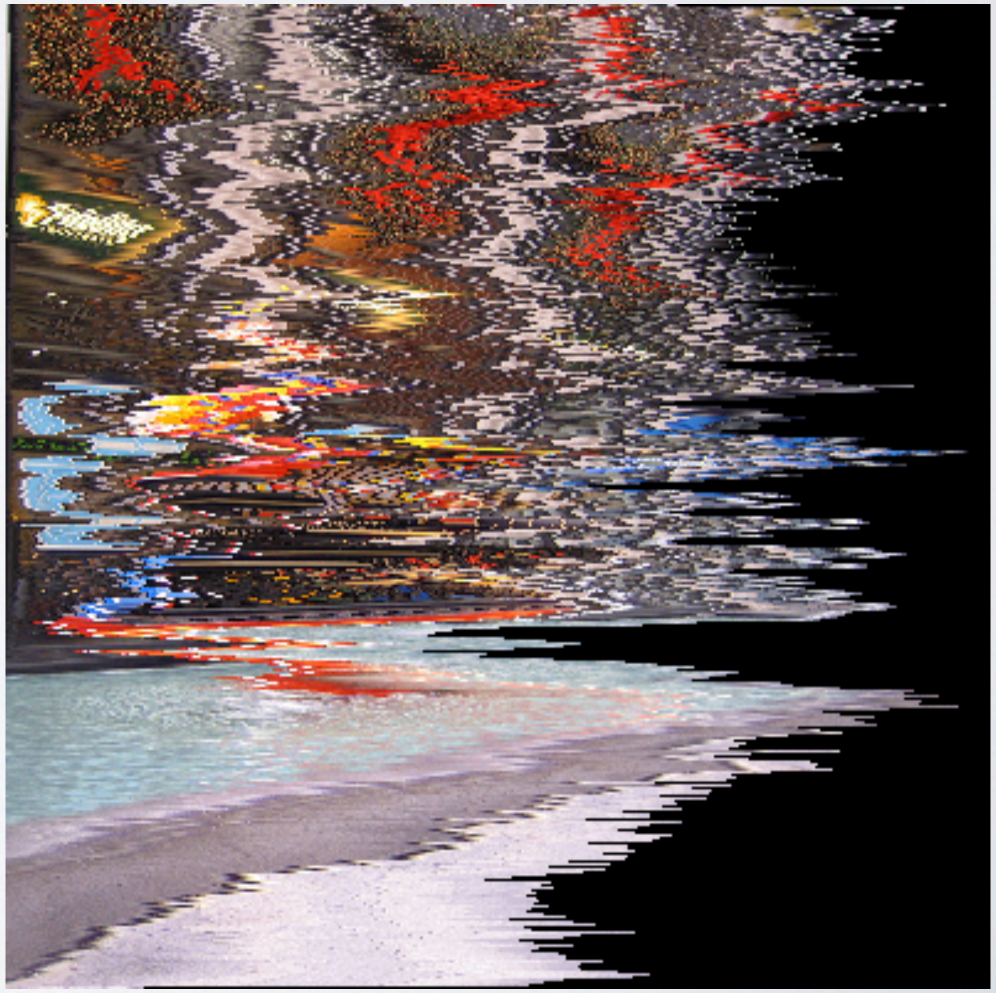


Image Resizing: Approach 2

- Remove **n** least important pixels in each row
- Still not great, too much shifting between adjacent rows

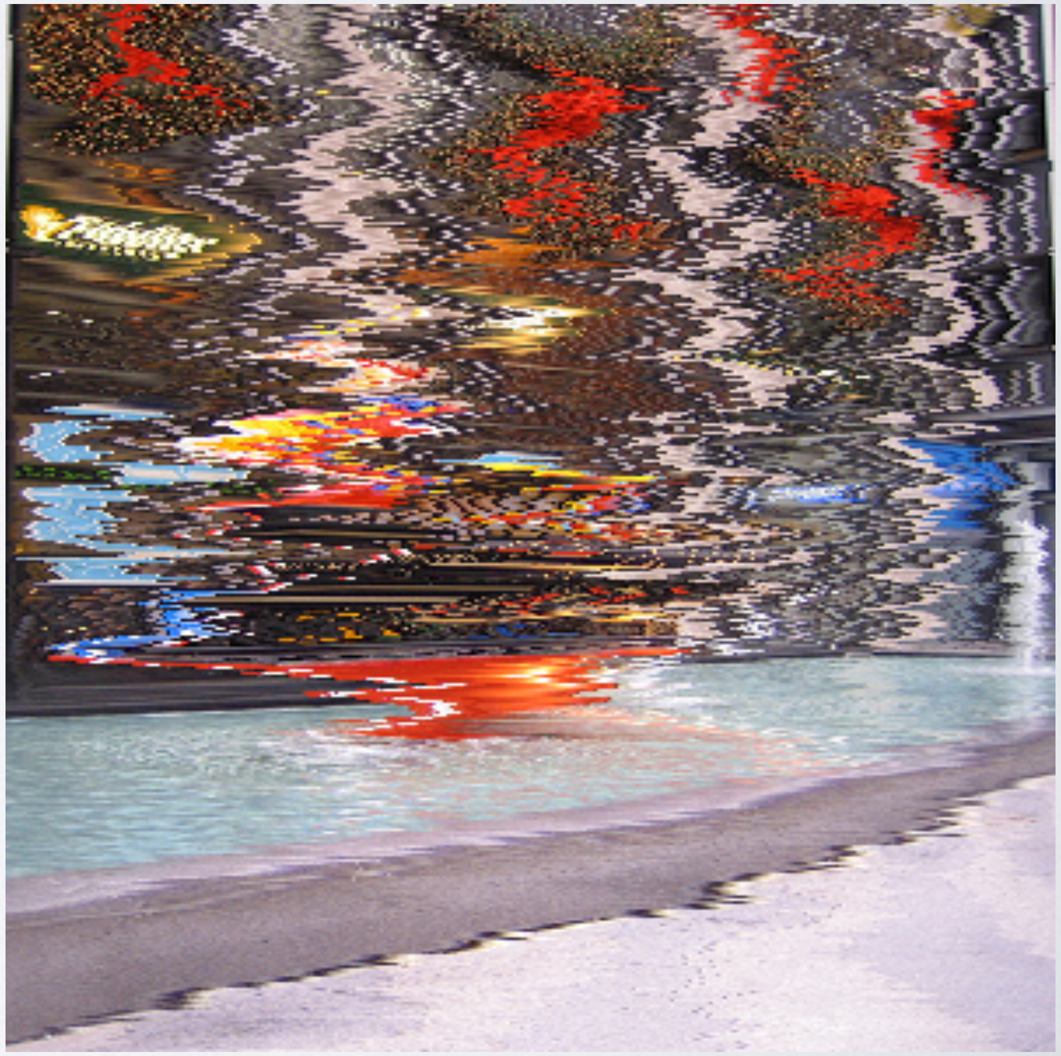


Image Resizing: Approach 3

- Remove column whose total importance is smallest, and repeat
- Much better! But not perfect…

Image Resizing

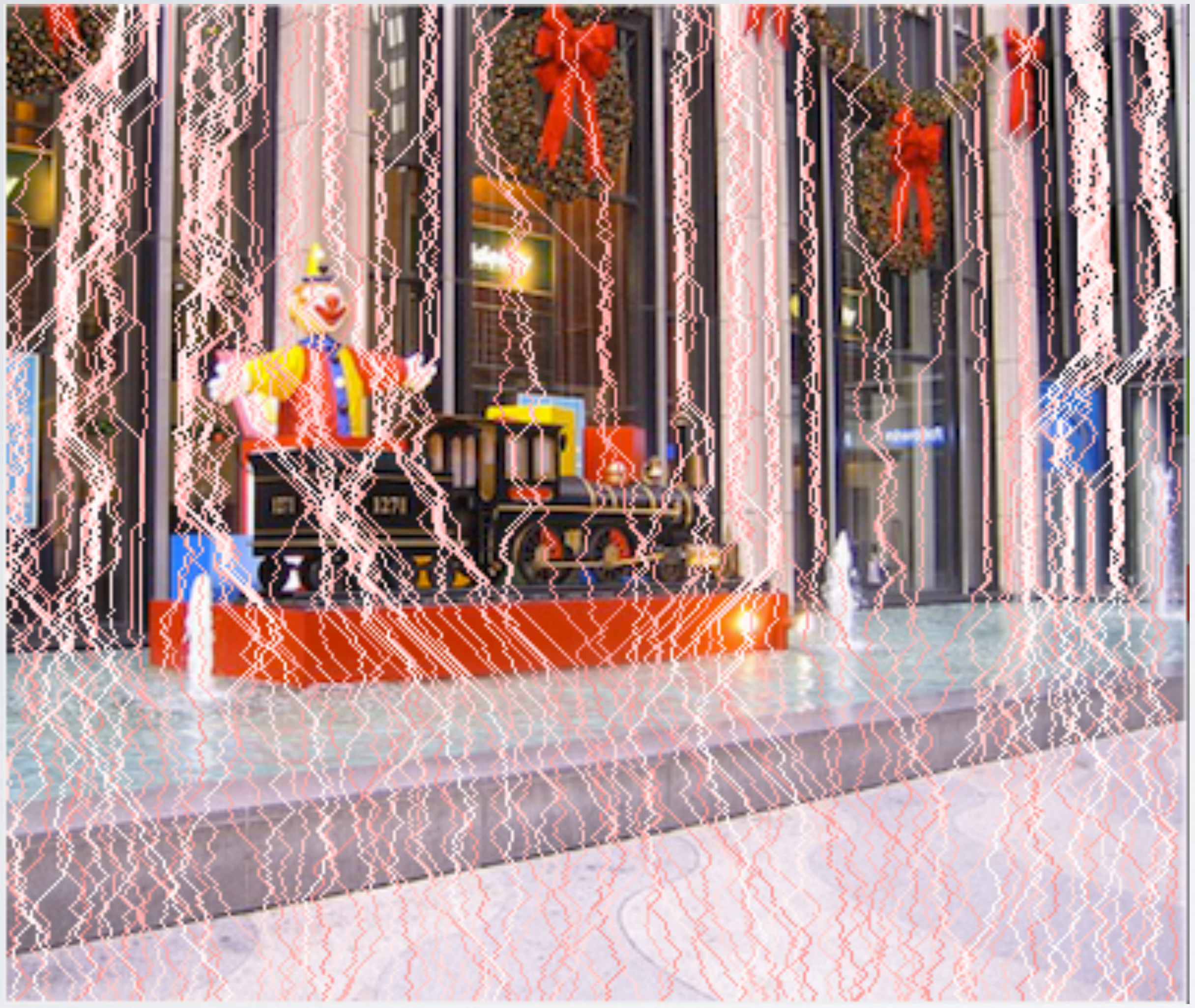
- Problem
  - removing entire column or entire row can distort image
- What pixels should we remove to resize this image?


YOU ARE
WELCOME
HERE
oSTEM
BROWN

Seam carving

- **Idea:** remove **seams** not columns
  - (vertical) seam is a path from top to bottom
  - that moves left or right by at most one pixel per row

Seam carving



Near Perfection!

Object Removal via seam carving

- Mark object to remove as "unimportant"
  - artificially deflate the importance of its pixels
- Pixels will be removed by algorithm

Seam carving

- Input
  - 2D array of importance values
- Output
  - Vertical seam with lowest importance

7x3 Importance Array

9	3	8	15	1	11	7
6	13	9	5	10	4	14
9	6	7	9	14	7	11

7x3 Importance Array

9	3	8	15	1	11	7
6	13	9	5	10	4	14
9	6	7	9	14	7	11

10x10 Importance Array

1	2	6	9	12	6	5	12	5	6
2	3	11	14	10	6	15	9	9	1
2	9	13	4	1	7	10	4	12	11
6	5	15	12	11	4	7	15	8	5
14	15	11	12	4	14	3	10	1	10
6	12	13	8	15	6	13	3	13	11
2	1	14	6	14	4	13	14	7	4
14	8	4	11	14	6	12	10	2	7
6	8	12	13	2	11	6	6	8	7
11	2	15	9	8	12	10	8	6	9

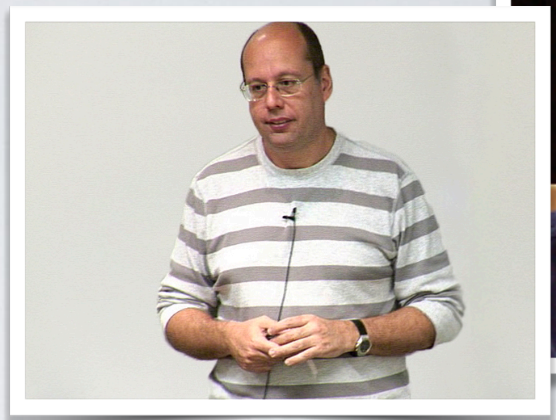
10x10 Importance Array

1	2	6	9	12	6	5	12	5	6
2	3	11	14	10	6	15	9	9	1
2	9	13	4	1	7	10	4	12	11
6	5	15	12	11	4	7	15	8	5
14	15	11	12	4	14	3	10	1	10
6	12	13	8	15	6	13	3	13	11
2	1	14	6	14	4	13	14	7	4
14	8	4	11	14	6	12	10	2	7
6	8	12	13	2	11	6	6	8	7
11	2	15	9	8	12	10	8	6	9

Seams

- Approximately **$Cx3^R$** seams in **CxR** image
- For **10x10**
  - ≈ **590,490** seams
- For **500x500**
  - ≈ **1.81801…x10^{241}** seams (242 digits)
- Age of the Universe
  - **4.3x10^{17}** seconds

Seam carving




- Invented by
  - Shai Avidan (MERL)
  - Ariel Shamir (Interdisciplinary Center, Herzliya)
  - Published at SIGGRAPH 2007
- Very fast
  - `~1` second to find the best seam on `800x533` image
- "Content aware scaling" in Photoshop, others

The Seam carving Algorithm

- Function `find_least_important_seam(vals)`
  - **input**: `vals` is a 2D array of importance values
  - **output**: sequence of column indices that represents a seam

```
[[ - S - - ],
 [ S - - - ],
 [ - S - - ],
 [ - - S - ]]     ⟶     [1, 0, 1, 2]
```

7x7 Importance Array

13	3	1	10	8	11	4
6	10	4	11	12	5	10
1	6	14	10	7	14	7
14	12	10	15	13	3	8
9	3	8	15	1	11	7
6	13	9	5	10	4	14
9	6	7	9	14	7	11

Seam = `[6,5,4,5,4,5,5]`

Data Structures Needed

- `costs`: 2D array filled in from bottom to top
  - `costs[row][col]:` importance of lowest-cost seam starting at `row & col`
- `dirs`: 2D array filled in at the same time as costs
  - `dirs[row][col]:` direction `(-1,0,1)` of next pixel in lowest-cost seam starting at `row & col`

`vals`

3	6	8
5	7	2
4	9	3

`costs`

9	10	
4	9	3

`dirs`

0	1	0
-	-	-

Data Structures Needed

vals

3	6	8
5	7	2
4	9	3

costs

9	10	
4	9	3

dirs

0	1	0
-	-	-

```
costs[row][col] = min(costs[row+1][col-1],
                      costs[row+1][col],
                      costs[row+1][col+1])
                  + vals[row][col]

dirs[row][col] = -1 if min is costs[row+1][col-1]
                  0 if min is costs[row+1][col]
                 +1 if min is costs[row+1][col+1]
```

Simulating seam carving

vals

3	6	8
5	7	2
4	9	3

costs

4	9	3

dirs

Finding Least Important Seam

- Once **`costs`** is completely filled in
  - cell in top row with minimum value is the first pixel in least important seam
- Starting from that pixel
  - follow directions in **`dirs`** to find least important seam
  - and build its column index representation

Seamcarve Pseudocode

```
function find_least_important_seam(vals):
    dirs = 2D array with same dimensions as vals
    costs = 2D array with same dimensions as vals
    costs[height-1] = vals[height-1] // initialize bottom row of costs

    for row from height-2 to 0:
        for col from 0 to width-1:
            costs[row][col] = vals[row][col] +
                              min(costs[row+1][col-1],
                                  costs[row+1][col],
                                  costs[row+1][col+1])
            dirs[row][col] = -1, 0, or 1 // depending on min

    // Find least important start pixel
    min_col = argmin(costs[0]) // Returns index of min in top row

    // Create vertical seam of size 'height' by tracing from top
    seam = []
    seam[0] = min_col
    for row from 0 to height-2:
        seam[row+1] = seam[row] + dirs[row][seam[row]]

    return seam
```

What's **`argmin`**?

- What does **`min`** do?
  - returns minimum output of a function
- What does **`argmin`** do?
  - given function **`f(x)`** returns x that minimizes **`f(x)`**
- **$f(x) = -1+x^2$**
  - **`min f = -1`**
  - **`argmin f = 0`** // value for which f is -1
- Array **`A = [5,4,1,3,9]`**
  - **`min(A) = 1`**
  - **`argmin(A) = 2`** // the index of the minimum value

How fast is this algorithm?

```
function find_least_important_seam(vals):
    dirs = 2D array with same dimensions as vals
    costs = 2D array with same dimensions as vals
    costs[height-1] = vals[height-1] // initialize bottom row of costs

    for row from height-2 to 0:
        for col from 0 to width-1:
            costs[row][col] = vals[row][col] +
                              min(costs[row+1][col-1],
                                  costs[row+1][col],
                                  costs[row+1][col+1])
            dirs[row][col] = -1, 0, or 1 // depending on min

    // Find least important start pixel
    min_col = argmin(costs[0]) // Returns index of min in top row

    // Create vertical seam of size 'height' by tracing from top
    seam = []
    seam[0] = min_col
    for row from 0 to height-2:
        seam[row+1] = seam[row] + dirs[row][seam[row]]

    return seam
```

References

- Slide #5
  - A statue of Muhammad ibn Musa al-Khwarizmi; a persian scholar from the 9th century
  - "Algorithms" is derived from "Algoritmi" which is the Latin translation of his name
  - Worked in mathematics, astronomy and geometry
  - Founded the field of Algebra