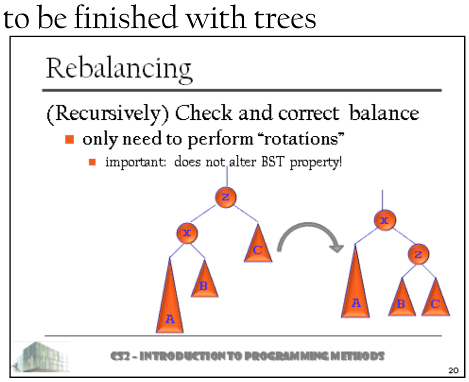
CS 2

Introduction to Programming Methods

Last Time

Intro to data structures

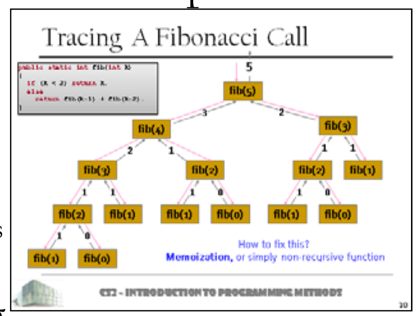
- how to store your data in a convenient form
  - missing one lecture to be finished with trees



Remember Fibonacci?

Example where recursion is hurtful

- Recursion w/ slightly smaller subproblems
  - Fib(n) calls Fib(n-1)
    - not like merge sort!
  - sort: log depth of tree
  - Fib: linear depth
    - exponential # of nodes
    - and many nodes are repeats



Dynamic programming

- used for certain optimization problems

When Is Dynamic Prog. Useful?

Optimal substructure

- solution can be constructed efficiently from (optimal) solutions to subproblems
  - includes merge sort or towers of Hanoi, for instance

Overlaping subproblems

- recursions like Fibonacci, with redundancy

DP proceeds much faster

- through various improvements
  - including memoization (i.e., storing previous results)

More Interesting Problem

How to reformat an image?

- scaling or cropping no good
- need to keep relevant parts

More Interesting Problem

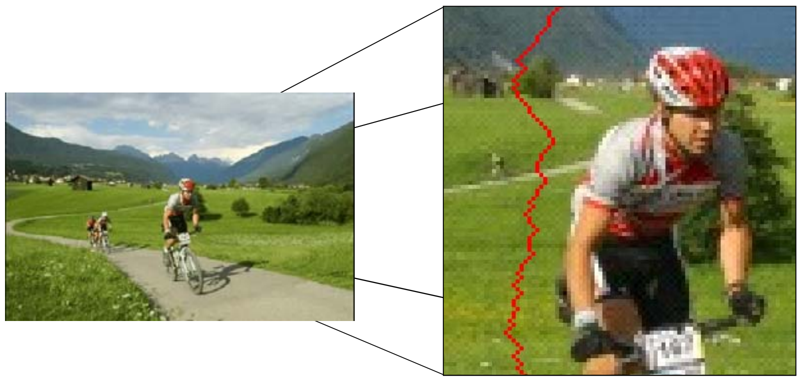
Ho to reformat an image?

- scaling or cropping no good
- need to keep relevant parts
  - or to remove old partners…

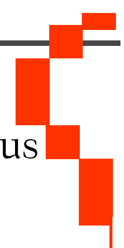
Simple Idea: Seam Carving

Remove a continuous line of pixels

- least conspicuous, from top to bottom



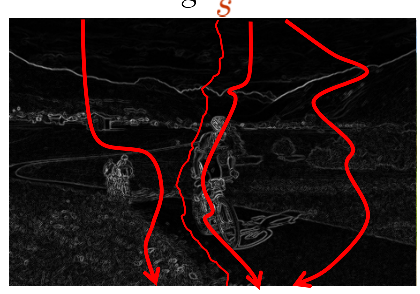
The Optimal Seam



What is a (vertical) seam?

- one pixel per row; all pixels contiguous
- with least "saliency"
  - saliency map derived from color image
  - score = sum of pixels
  - best seam = min score

s^*



$$E(s) = \sum_i \text{saliency}(s_i)$$

$$E(s^*) = \min_s \ E(s)$$

Computing Saliency Map

Many ways, but gradient norm G enough

- linear combos of 8 neighbors of each pixel

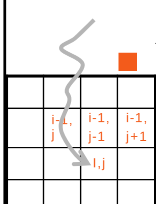
$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

How to Find the Optimal Path

Seemingly, like a needle in a haystack

- for a NxN image, lots of possible seams
- brut force approach? Try and score them all
  - could use recursion
    - ex: $M(i,j) = \text{saliency}(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$
    - but HUGE redundancy; & how to get seam from best score?



- can find it by **dynamic programming**
  - minimum path found by storing the optimal paths to reach *each pixel* in a cost table
  - table constructed by top-down memoizing
  - final path inferred by back tracing through the table.
    - typical when lots of overlapping sub problems

Construction of Cost Table

Top to bottom (for vertical seams)

- from second to last row

5	8	3	9
9	2	3	9
7	3	4	2
4	5	7	8

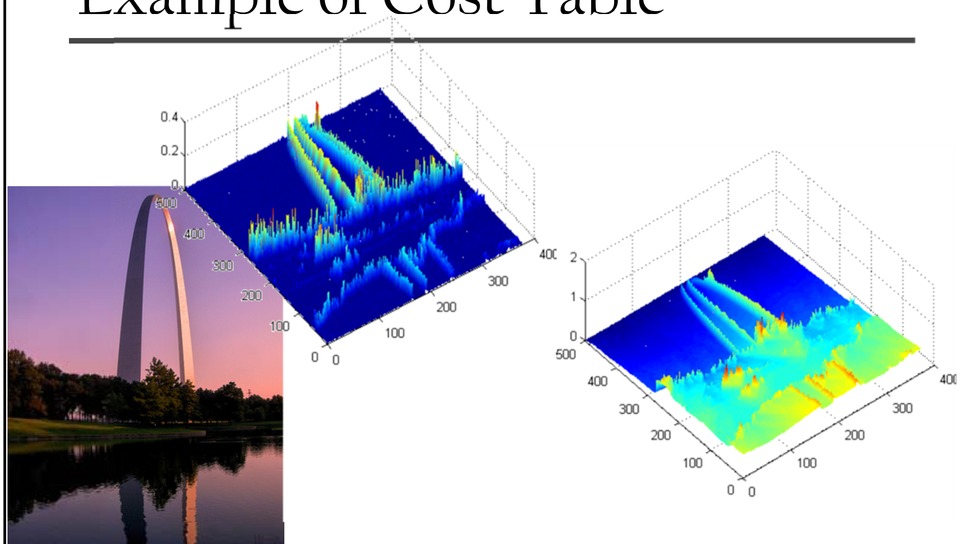
saliency

5	8	3	9
14	5	6	12
12	8	9	8
12	13	15	16

cost

$$M(i,j) = \text{saliency}(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

Example of Cost Table



Finding Optimal Seam

Now just backtrace!

- start with min value on last row
  - which is the optimal cost, by the way
- … and walk up from value to value

5	8	3	9
14	5	6	12
14	8	9	8
12	13	15	16

Typical Seams

Expanding an Image?

Same idea

- find optimal seam
- insert pixels (local averages)


Other Dynamic Prog Examples

Find min # of (nonUS) coins to make an amount

- to find the solution for 13¢,
  - solve for all of 1¢, 2¢, 3¢, ..., 12¢
  - choose best among {solution for i¢ + solution for 13-i ¢ }

Knapsack problem

- various item types of various values & weights
- bag of limited total weight→how to make most valuable?

Gene Sequence Alignment

Shortest Path

- more on this later