CS 4102: Algorithms

Lecture 14: Dynamic Programming

David Wu

Fall 2019

Today’s Keywords

Dynamic Programming

Gerrymandering

Greedy Algorithms

Choice Function

Change Making

CLRS Readings: Chapter 15, 16

Homework

- Midterm take-home due **tonight** at 11pm
  - Individual work

- No office hours / regrade office hours until tomorrow (after midterm)

- **HW5** released later today, due **Thursday, October 24**, 11pm
  - Seam Carving
  - Dynamic Programming (implementation)
  - Java or Python

Dynamic Algorithms Examples

Maximum Sum Continuous Subarray

Tiling Dominoes

Log Cutting

Matrix Chaining

Longest Common Subsequence

Seam Carving

Maximum Sum Continuous Subarray

5	13	9	12	19	4	6	14	0	17	25	0	0	22
5	13	13	13	19	19	19	19	19	19	25	25	25	25
5	8	-4	3	7	-15	2	8	-20	17	8	-50	-5	22
0	1	2	3	4	5	6	7	8	9	10	11	12	13

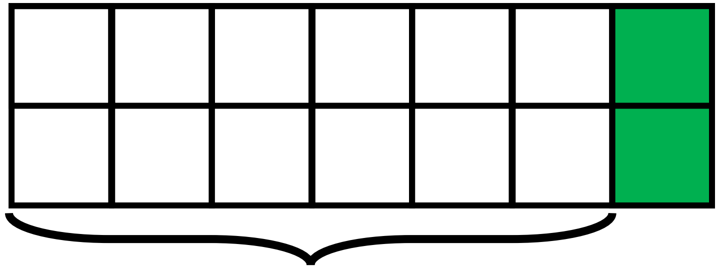
Divide

Observation: No need to recurse! Just maintain <u>two</u> numbers and iterate from 1 to n: best value so far, best value ending at current position

$$BED(n) = \max(BED(n-1) + arr[n], 0)$$
$$BSL(n) = \max\big(BSL(n-1), BED(n)\big)$$

Tiling Dominoes

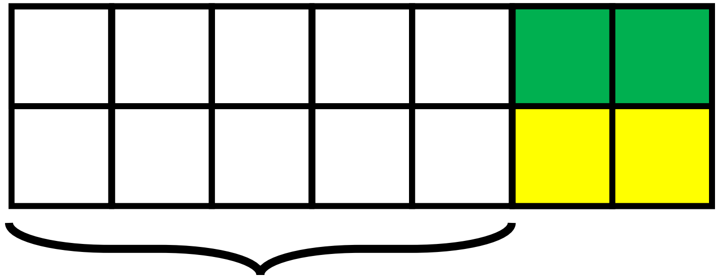
Two ways to fill the final column:



$n-1$

$$\text{Tile}(n) = \text{Tile}(n-1) + \text{Tile}(n-2)$$

$$\text{Tile}(0) = \text{Tile}(1) = 1$$



$n-2$

Log Cutting

$P[i]$ = value of a cut of length i

$\text{Cut}(n)$ = value of best way to cut a log of length n

$$\text{Cut}(n) = \max \begin{cases} \text{Cut}(n-1) + P[1] \\ \text{Cut}(n-2) + P[2] \\ \vdots \\ \text{Cut}(0) + P[n] \end{cases}$$

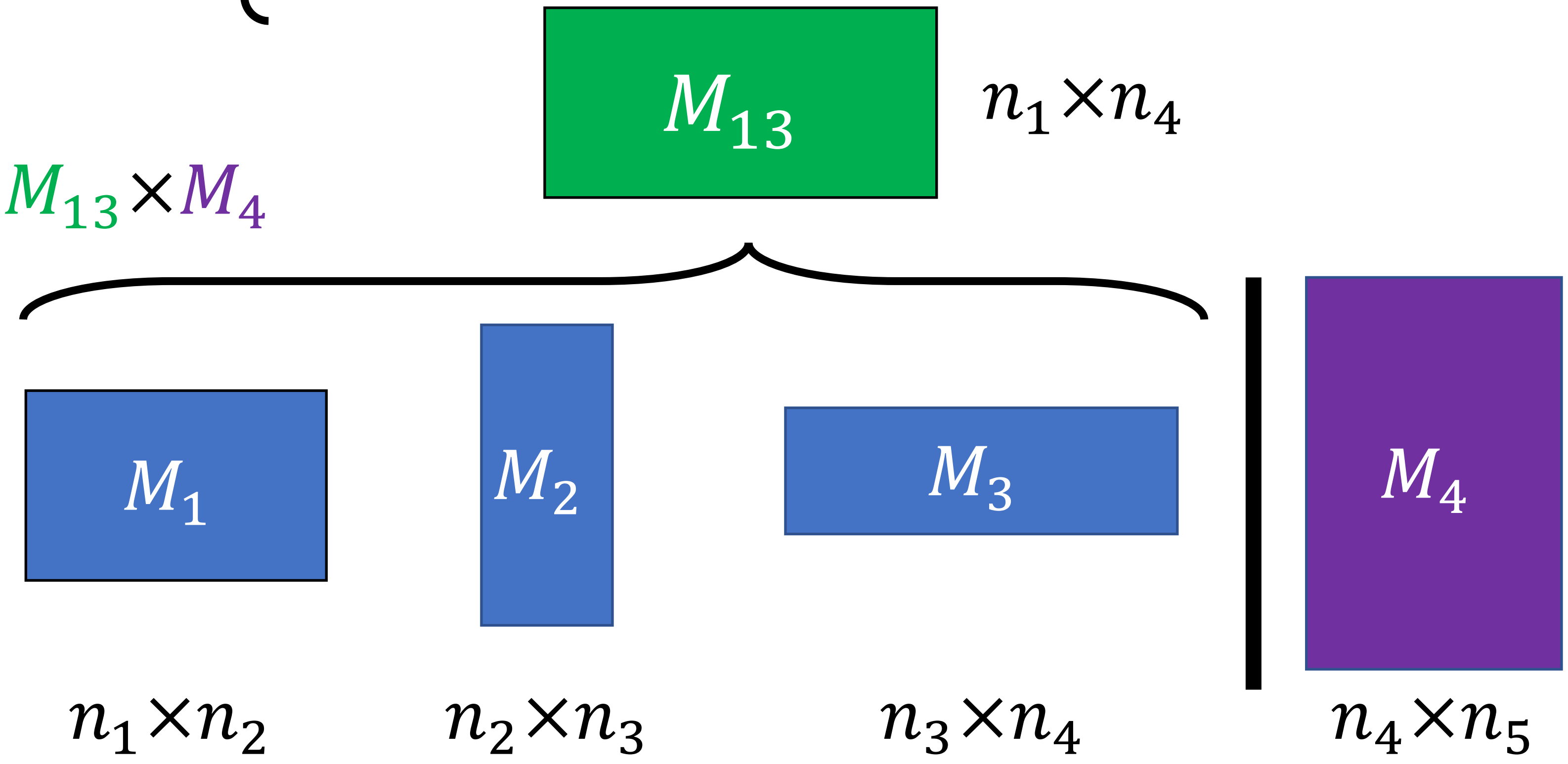
$Cut(n - \ell_n)$ $\qquad$ ℓ_n

Matrix Chaining

$\text{Best}(1, n) =$ cheapest way to multiply together M_1 through M_n

$$\text{Best}(1,4) = \min \begin{cases} \text{Best}(2,4) + n_1 n_2 n_5 \\ \text{Best}(1,2) + \text{Best}(3,4) + n_1 n_3 n_5 \\ \text{Best}(1,3) + n_1 n_4 n_5 \end{cases}$$

Last product: $M_{13} \times M_4$



Longest Common Subsequence

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 $|X| = n$, $|Y| = m$)

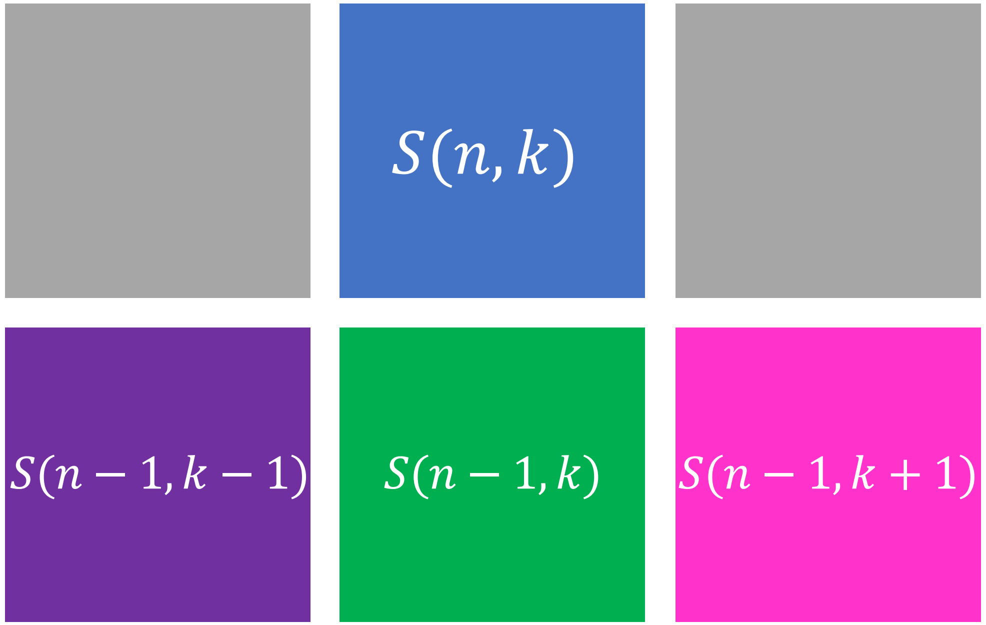
Seam Carving

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)

$p_{n,k}$

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



Gerrymandering

Manipulating electoral district boundaries to favor one political party over others

Coined in an 1812 political cartoon after Governor Gerry signed a bill that redistricted Massachusetts to benefit his Democratic-Republican Party

The Gerrymander

Gerrymandering

Supreme Court Associate Justice Anthony Kennedy gave no sign that he has abandoned his view that extreme partisan gerrymandering might violate the Constitution. | Eric Thayer/Getty Images

Supreme Court eyes partisan gerrymandering

Anthony Kennedy is seen as the swing vote that could blunt GOP's map-drawing successes.

Gerrymandering

SUPREME COURT OF THE UNITED STATES

Syllabus

VIRGINIA HOUSE OF DELEGATES ET AL. *v.*

SUPREME COURT OF THE UNITED STATES

Syllabus

Next Gerrymandering Battle in North Carolina: Congress

A North Carolina court threw out the state's legislative map as an illegal gerrymander. Now the same court could force the state to redraw the state's congressional districts as well.

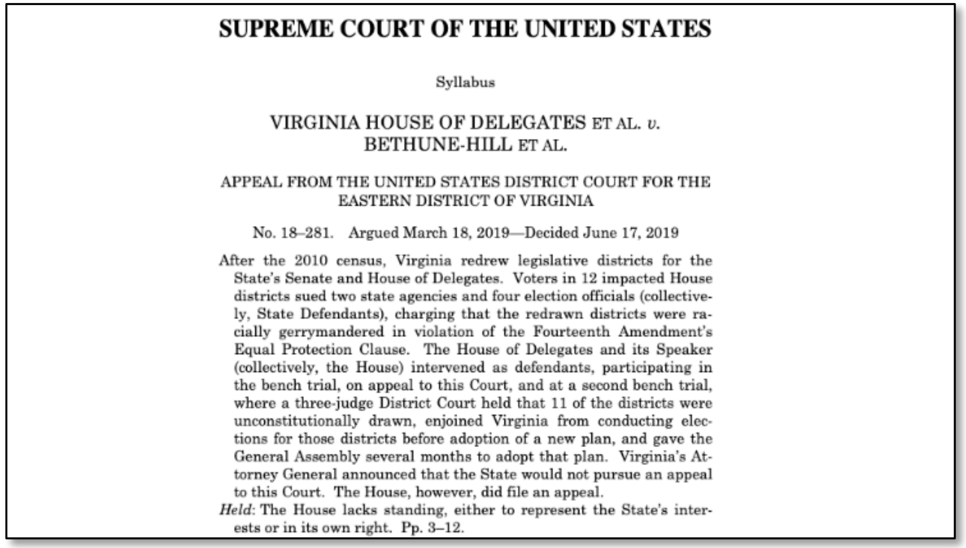
According to the Supreme Court…

Gerrymandering cannot be used to:

- Disadvantage racial/ethnic/religious groups

It can be used to:

- Disadvantage political parties



SUPREME COURT OF THE UNITED STATES

Syllabus

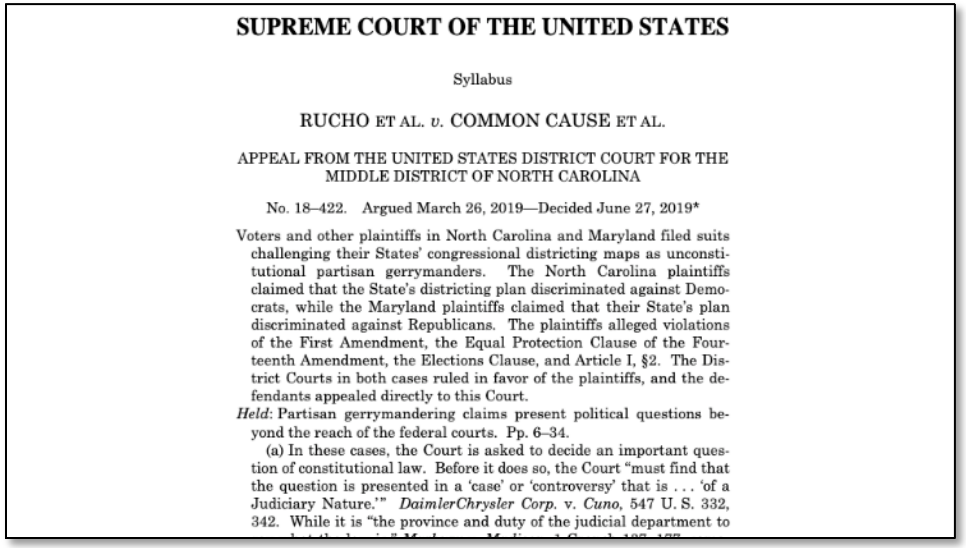
VIRGINIA HOUSE OF DELEGATES ET AL. *v.* BETHUNE-HILL ET AL.

APPEAL FROM THE UNITED STATES DISTRICT COURT FOR THE EASTERN DISTRICT OF VIRGINIA

No. 18–281. Argued March 18, 2019—Decided June 17, 2019

After the 2010 census, Virginia redrew legislative districts for the State's Senate and House of Delegates. Voters in 12 impacted House districts sued two state agencies and four election officials (collectively, State Defendants), charging that the redrawn districts were racially gerrymandered in violation of the Fourteenth Amendment's Equal Protection Clause. The House of Delegates and its Speaker (collectively, the House) intervened as defendants, participating in the bench trial, on appeal to this Court, and at a second bench trial, where a three-judge District Court held that 11 of the districts were unconstitutionally drawn, enjoined Virginia from conducting elections for those districts before adoption of a new plan, and gave the General Assembly several months to adopt that plan. Virginia's Attorney General announced that the State would not pursue an appeal to this Court. The House, however, did file an appeal.

Held: The House lacks standing, either to represent the State's interests or in its own right. Pp. 3–12.



SUPREME COURT OF THE UNITED STATES

Syllabus

RUCHO ET AL. *v.* COMMON CAUSE ET AL.

APPEAL FROM THE UNITED STATES DISTRICT COURT FOR THE MIDDLE DISTRICT OF NORTH CAROLINA

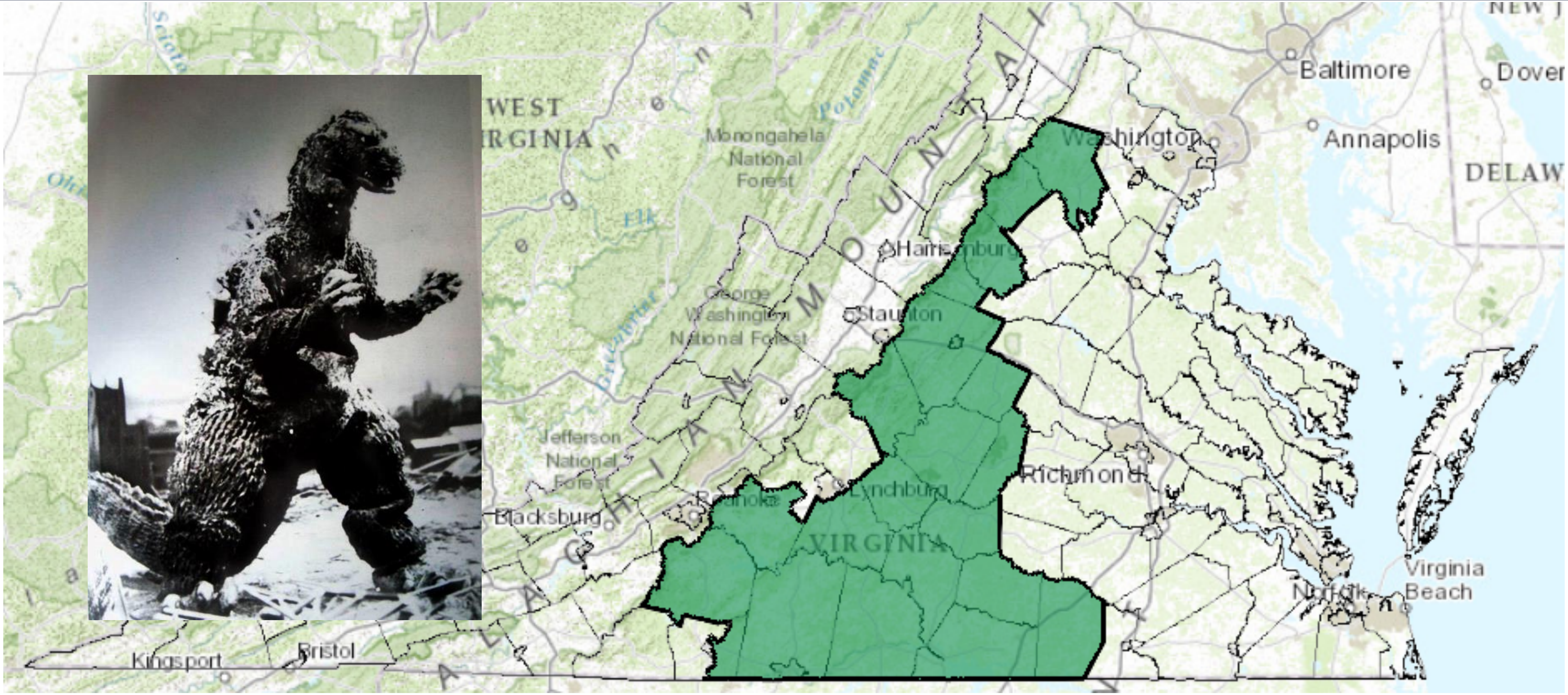
No. 18–422. Argued March 26, 2019—Decided June 27, 2019*

Voters and other plaintiffs in North Carolina and Maryland filed suits challenging their States' congressional districting maps as unconstitutional partisan gerrymanders. The North Carolina plaintiffs claimed that the State's districting plan discriminated against Democrats, while the Maryland plaintiffs claimed that their State's plan discriminated against Republicans. The plaintiffs alleged violations of the First Amendment, the Equal Protection Clause of the Fourteenth Amendment, the Elections Clause, and Article I, §2. The District Courts in both cases ruled in favor of the plaintiffs, and the defendants appealed directly to this Court.

Held: Partisan gerrymandering claims present political questions beyond the reach of the federal courts. Pp. 6–34.

(a) In these cases, the Court is asked to decide an important question of constitutional law. Before it does so, the Court "must find that the question is presented in a 'case' or 'controversy' that is . . . 'of a Judiciary Nature.'" *DaimlerChrysler Corp.* v. *Cuno*, 547 U. S. 332, 342. While it is "the province and duty of the judicial department to

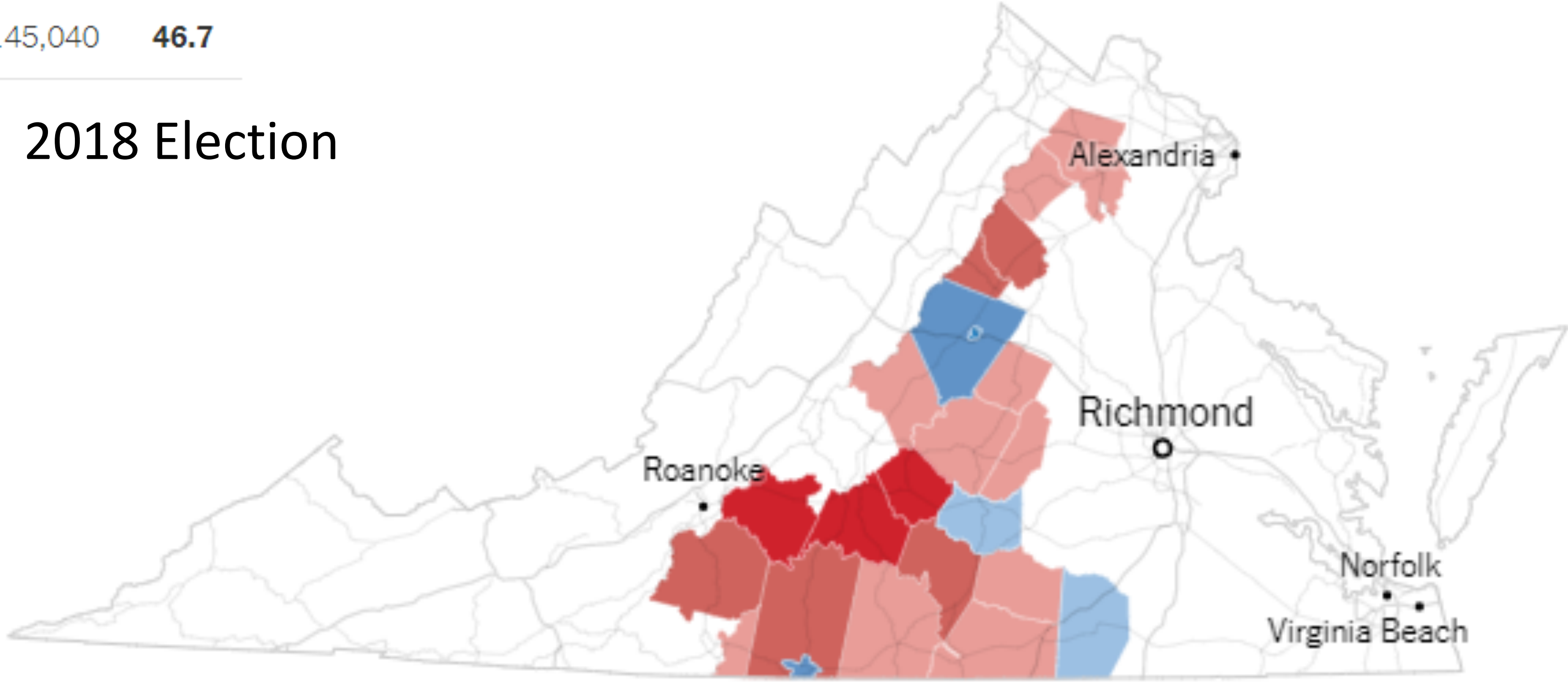
VA 5th District



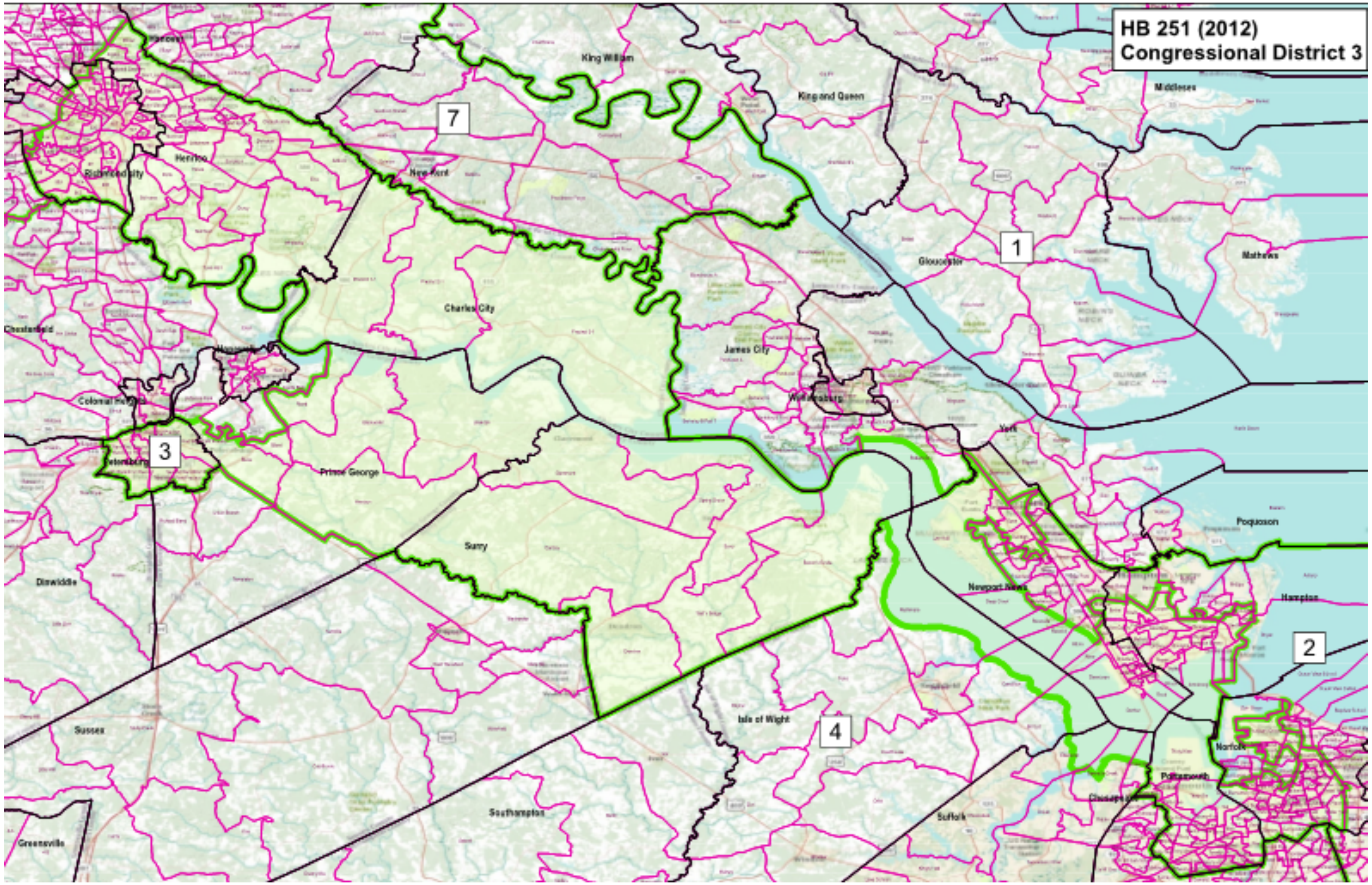
VA 5th District

	Votes	Pct.
	165,339	**53.3%**
	145,040	**46.7**

2018 Election

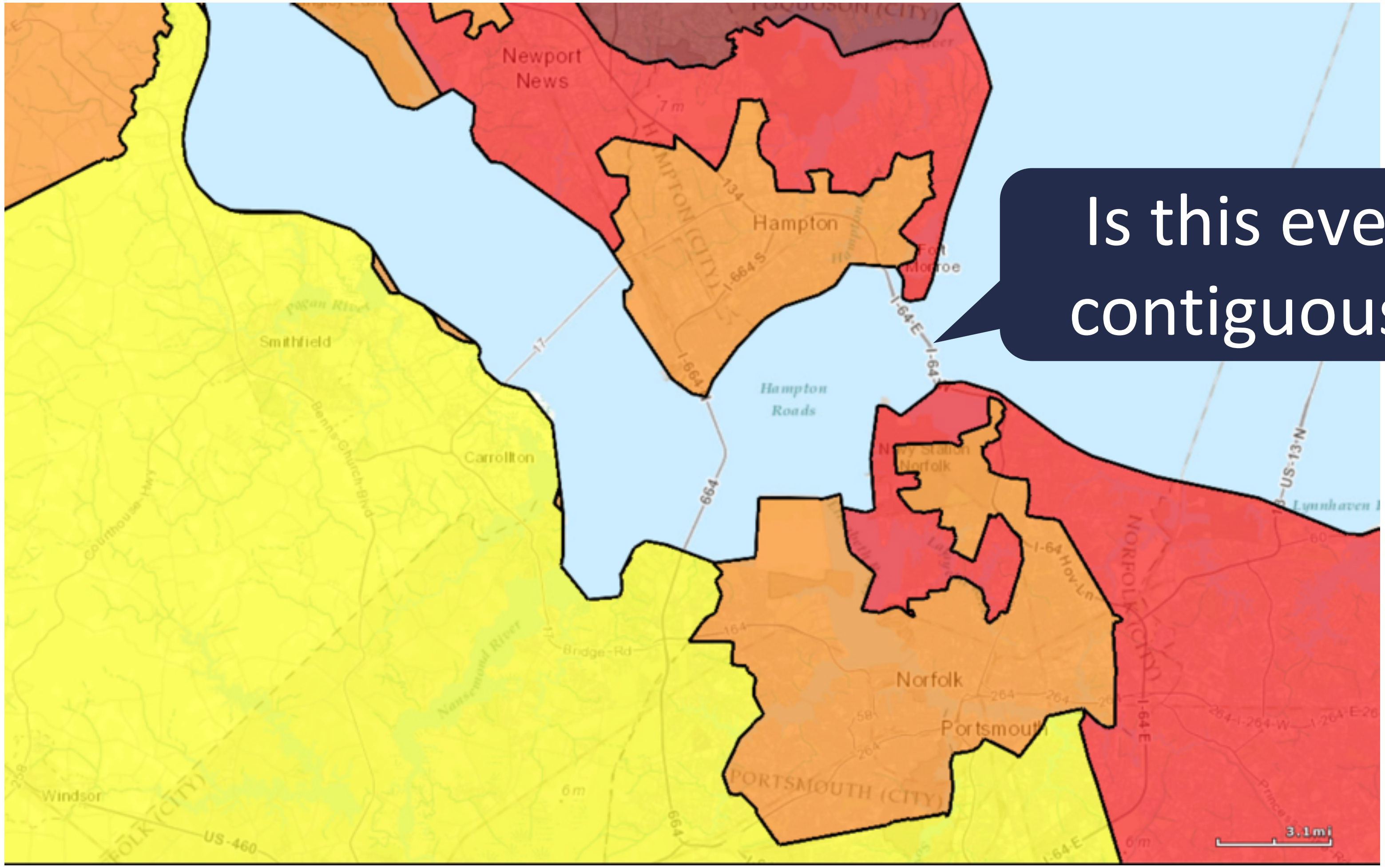


Gerrymandering Today



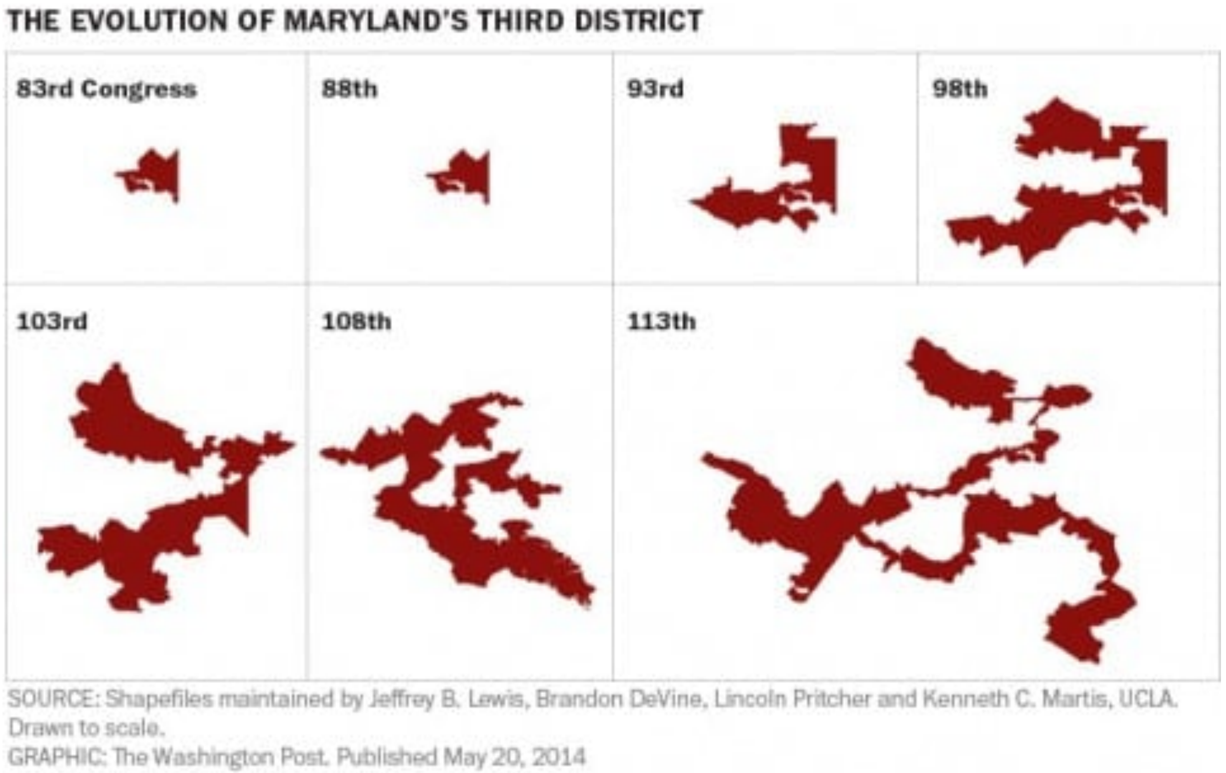
Gerrymandering Today

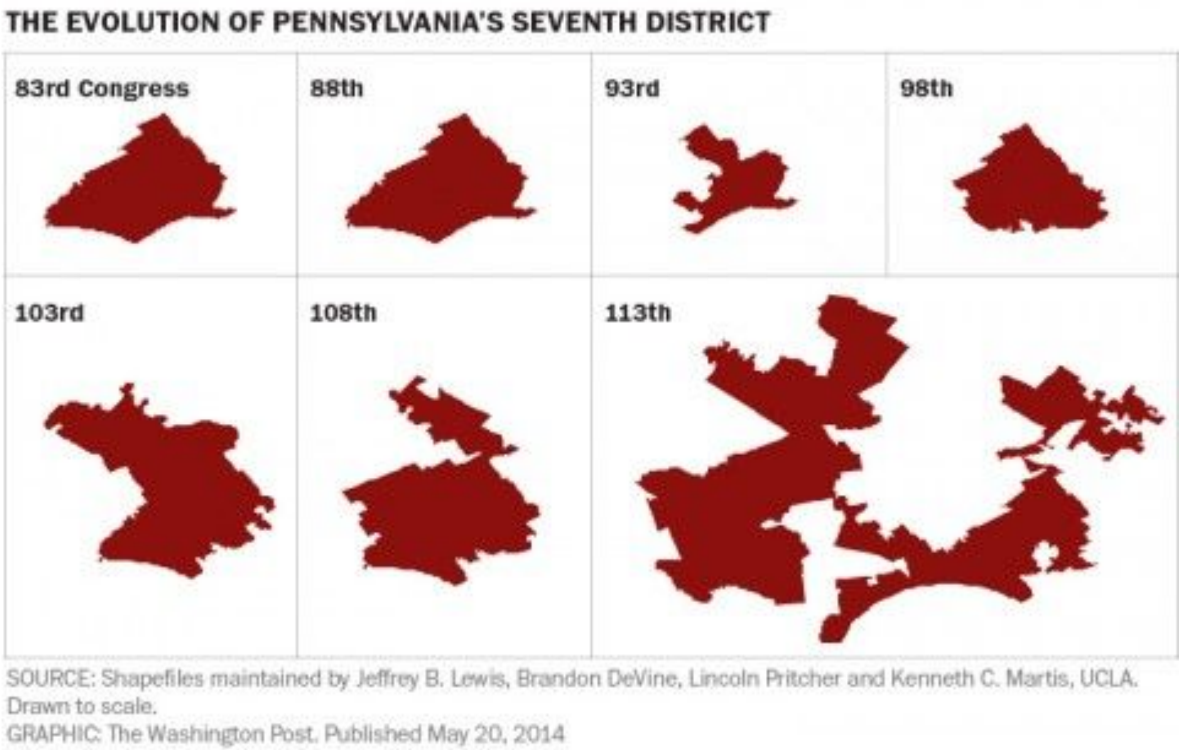
Computers make it very effective



Is this even contiguous?

Gerrymandering Today





How Does it Work?

- States are broken into precincts
- All precincts have the same number of people
- We know voting preferences of each precinct
- Group precincts into districts to maximize the number of districts won by my party

Each district should have roughly the same number of people

Overall: R:217 D:183

R:65 D:35	R:45 D:55
R:60 D:40	R:47 D:53

100 voters per precinct

(R)

vs.

(D)

How Does it Work?

- States are broken into precincts
- All precincts have the same number of people
- We know voting preferences of each precinct
- Group precincts into districts to maximize the number of districts won by my party

Each district should have roughly the same number of people




Overall: R:217 D:183

R:65 D:35	R:45 D:55
R:60 D:40	R:47 D:53

R:125 R:92

R:65 D:35	R:45 D:55
R:60 D:40	R:47 D:53

R:112 R:105

R:65 D:35	R:45 D:55
R:60 D:40	R:47 D:53

100 voters per precinct

Gerrymandering Problem Statement

Given:

- A list of precincts: $p_1, p_2, \ldots, p_n$
- Each precinct contains exactly m voters

Output districts $D_1, D_2 \subset \{p_1, p_2, \ldots, p_n\}$ where:

- $|D_1| = |D_2|$
- $R(D_1), R(D_2) > \frac{mn}{4}$ where

$R(D_i)$ is the number of "Regular Party" voters in D_i

mn voters in total

Assign precincts to districts

Districts have the same size

Party has majority of voters in the district (at least $mn/4$ voters since each district has $mn/2$ voters)

If no such assignment is possible, output `impossible`

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

Consider the <u>Last</u> Precinct

After assigning the first $n-1$ precincts $p_1, \dots, p_{n-1}$

Observation: succeed if there is a way to assign k precincts from $\{p_1, \dots, p_{n-1}\}$ to D_1 with x voters in D_1 and y voters in D_2 such that either

- $k+1 = n/2$ and $x + R(p_n) > mn/4$ and $y > mn/4$; or
- $k = n/2$ and $x > mn/4$ and $y + R(p_n) > mn/4$

k precincts
x voters for R

District D_1

$n-k-1$ precincts
y voters for R

District D_2

p_n

assign p_1 to D_1

assign p_2 to D_2

$k+1$ precincts
$x + R(p_n)$ voters for R

District D_1

Valid gerrymandering if:
$k+1 = n/2$,
$x + R(p_n), y > mn/4$

$n-k$ precincts
$y + R(p_n)$ voters for R

District D_2

Valid gerrymandering if:
$n-k = n/2$,
$x, y + R(p_n) > mn/4$

Define Recursive Structure

Recursive substructure: can we achieve a specific split of the precincts?

Observation: succeed if there is a way to assign k precincts from $\{p_1, \dots, p_{n-1}\}$ to D_1 with x voters in D_1 and y voters in D_2 such that either

- $k + 1 = n/2$ and $x + R(p_n) > mn/4$ and $y > mn/4$; or
- $k = n/2$ and $x > mn/4$ and $y + R(p_n) > mn/4$

$S(j, k, x, y) =$ True if from among the first $\boldsymbol{j}$ precincts:

$\boldsymbol{k}$ are assigned to D_1

exactly $\boldsymbol{x}$ vote for R in D_1

exactly $\boldsymbol{y}$ vote for R in D_2

Goal: see if there exists $x, y > mn/4$ such that $S(n, n/2, x, y)$ is true

Define Recursive Structure

Recursive substructure: can we achieve a specific split of the precincts?

Observation: succeed if there is a way to assign k precincts from $\{p_1, \ldots, p_{n-1}\}$ to D_1 with x voters in D_1 and y voters in D_2 such that either

- $k + 1 = n/2$ and $x + R(p_n) > mn/4$ and $y > mn/4$; or
- $k = n/2$ and $x > mn/4$ and $y + R(p_n) > mn/4$

$S(j, k, x, y) =$ True if from among the first $\boldsymbol{j}$ precincts:

$\boldsymbol{k}$ are assigned to D_1

exactly $\boldsymbol{x}$ vote for R in D_1

exactly $\boldsymbol{y}$ vote for R in D_2

4-dimensional dynamic programming!

Size of the memory?

$$n \times n \times mn \times mn$$

Identify Recursive Structure

$S(j,k,x,y) =$ True if from among the first $\boldsymbol{j}$ precincts:
$\boldsymbol{k}$ are assigned to D_1
exactly $\boldsymbol{x}$ vote for R in D_1
exactly $\boldsymbol{y}$ vote for R in D_2

Two possibilities: assign p_j to D_1 or assign p_j to D_2

Case 1: assign p_j to D_1

$S(j,k,x,y)$ is true if we can assign $k-1$ out of the first $j-1$ precincts to D_1 such that:

- exactly $x - R(p_j)$ vote for R in D_1
- exactly y vote for R in D_2

$$S\left(j-1, k-1, x-R(p_j), y\right)$$

Case 2: assign p_j to D_2

$S(j,k,x,y)$ is true if we can assign k out of the first $j-1$ precincts to D_1 such that:

- exactly x vote for R in D_1
- exactly $y - R(p_j)$ vote for R in D_2

$$S\left(j-1, k, x, y-R(p_j)\right)$$

Identify Recursive Structure

$S(j,k,x,y)$ = True if from among the first $\boldsymbol{j}$ precincts:
$\boldsymbol{k}$ are assigned to D_1
exactly $\boldsymbol{x}$ vote for R in D_1
exactly $\boldsymbol{y}$ vote for R in D_2

Two possibilities: assign p_j to D_1 or assign p_j to D_2

$$S(j,k,x,y) = S\left(j-1, k-1, x-R(p_j), y\right) \text{ OR } S\left(j-1, k, x, y-R(p_j)\right)$$

Base Case: $S(0,0,0,0) = \text{True}$

$S(0,k,x,y) = \text{False for all } k,x,y$

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

Find a Good Ordering

$S(j,k,x,y) =$ True if from among the first $\boldsymbol{j}$ precincts:
$\boldsymbol{k}$ are assigned to D_1
exactly $\boldsymbol{x}$ vote for R in D_1
exactly $\boldsymbol{y}$ vote for R in D_2

Two possibilities: assign p_j to D_1 or assign p_j to D_2

$$S(j,k,x,y) = S(j-1, k-1, x-R(p_j), y) \text{ OR } S\left(j-1, k, x, y-R(p_j)\right)$$

Base Case: $S(0,0,0,0) = \text{True}$

$S(0,k,x,y) = \text{False for all } k,x,y$

Observation: Values with j only depend on values with $j-1$ (start with first component and fill in rest in order)

Final Algorithm

$$S(j,k,x,y) = S(j-1,k-1,x-R(p_j),y) \vee S\left(j-1,k,x,y-R(p_j)\right)$$

```
initialize S[0, 0, 0, 0] = True and False elsewhere
for j = 1,...,n:
    for k = 1,...,n:
        for x = 0,...,mn:
            for y = 0,...,mn:
                S[j, k, x, y] =
                    S[j - 1, k - 1, x - R[j], y] |
                    S[j - 1, k, x, y - R[j]]

return True if exists x > mn/4, y > mn/4 where
    S[n, n/2, x, y] = True
```

Can early terminate some of these loops, but same asymptotics

Value of $R(p_j)$

Running Time

$$S(j,k,x,y) = S(j-1,k-1,x-R(p_j),y) \vee S\left(j-1,k,x,y-R(p_j)\right)$$

```
initialize S[0, 0, 0, 0] = True and False elsewhere        O(m^2 n^4)
for j = 1,...,n:                                           O(n)
    for k = 1,...,n:                                       O(n)
        for x = 0,...,mn:                                  O(mn)
            for y = 0,...,mn:                              O(mn)
                S[j, k, x, y] =
                    S[j - 1, k - 1, x - R[j], y] |
                    S[j - 1, k, x, y - R[j]]

return True if exists x > mn/4, y > mn/4 where            O(m^2 n^2)
    S[n, n/2, x, y] = True
```

Overall Running Time: $O(m^2 n^4)$

Running Time

Is this an efficient algorithm?

Overall Running Time: $O(m^2 n^4)$

efficient = "polynomial time"

To be <u>efficient</u>, running time would have to be of the form $n^s(\log m)^t$ for constants s, t. But $m^2 = (\log m)^{\frac{2 \log m}{\log \log m}}$. We call this a "pseudo-polynomial" time algorithm.

Inputs to algorithm: $R(p_1), \ldots, R(p_n), m$

Length of inputs: $O(n \log m)$

Running time is <u>exponential</u> in *length* of input

In fact: Gerrymandering is NP-complete

Mental Stretch

Given access to an unlimited number of pennies, nickels dimes, and quarters, give an algorithm which gives change for a target value x using the <u>fewest</u> number of coins.

Change Making Algorithm

Given: target value x, list of coins $C = [c_1, \ldots, c_n]$
(in this case $C = [1, 5, 10, 25]$)

Repeatedly select the largest coin less than the remaining target value:

while $x > 0$:
  let $c = \max(c_i \in \{c_1, \ldots, c_n\} \mid c_i \leq x)$
  add c to list L
  $x = x - c$
output L

Example of a **greedy algorithm**:
always choose the “optimal” choice

Mental Stretch

Suppose we added a new coin worth 11 cents. In conjunction with pennies, nickels, dimes, and quarters, find the minimum number of coins needed to give 90 cents of change.

Greedy Solution

90 cents

Optimal Solution

90 cents

When can we use the greedy solution?

Greedy Algorithms

Requires optimal substructure

- Solution to larger problem contains the solution to a smaller one
- Only a single subproblem to consider

General Blueprint:

1. Identify a greedy choice property
   - Show that this choice is guaranteed to be included in some optimal solution
2. Repeatedly apply the choice property until no subproblems remain

Greedy vs Dynamic Programming

Dynamic Programming:

- Require optimal substructure
- Optimal choice can be one of <u>multiple</u> smaller subproblems

Greedy:

- Require optimal substructure
- Only a <u>single</u> choice and a single subproblem

Change Making Choice Property

Largest coin less than or equal to target value must be part of some optimal solution (for standard U.S. coins)

To be continued…