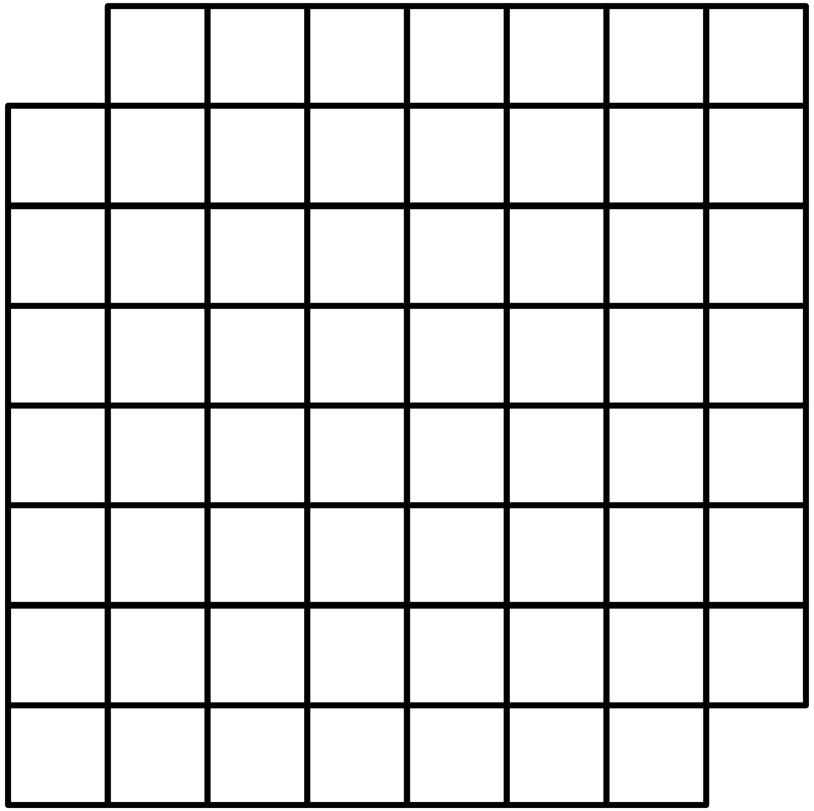
CS4102 Algorithms

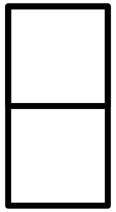
Spring 2022

Can you fill a 8×8 board with the corners missing using dominoes?

Can you tile this?

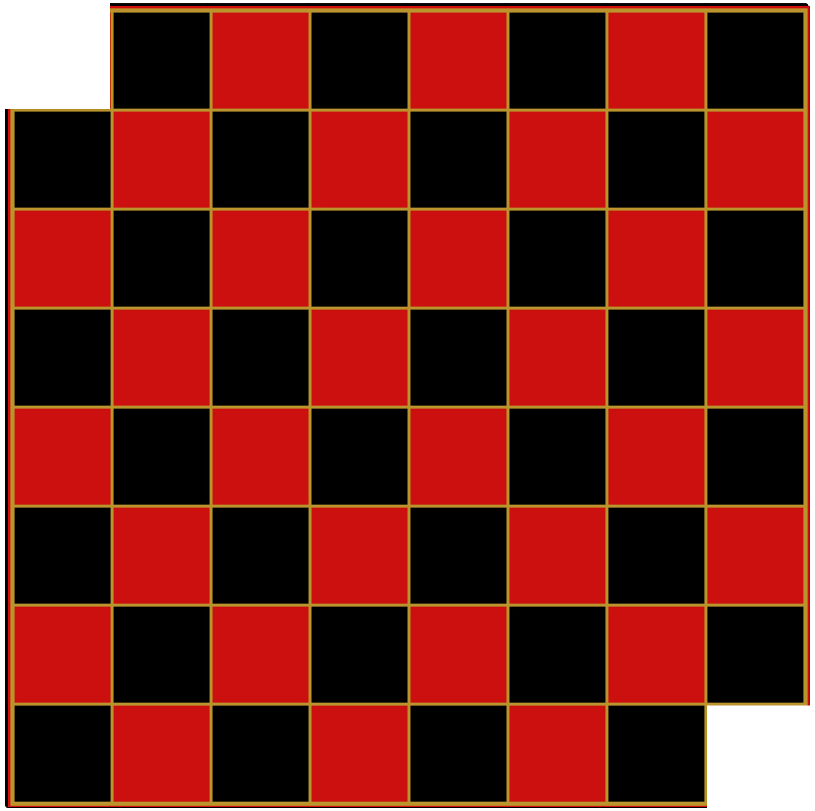


With these?

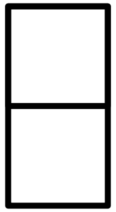


Can you fill a 8×8 board with the corners missing using dominoes?

Can you tile this?



With these?



Today's Keywords

- Dynamic Programming
- Gerrymandering

Announcements

- Unit B Adv and Programming due Friday 4/8
- Unit C Adv and Programming coming soon

Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
- Idea:
  1. Identify the recursive structure of the problem
     - What is the "last thing" done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest

Generic Top-Down Dynamic Programming Soln

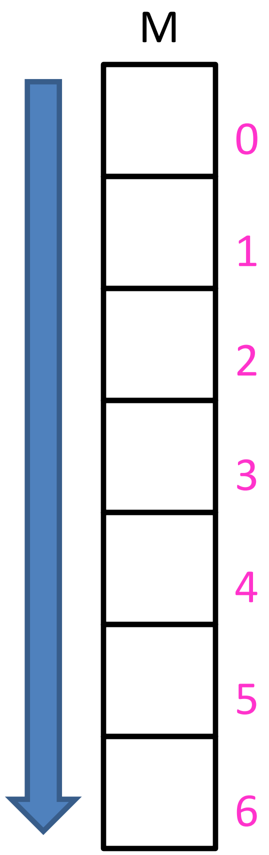
```
mem = {}
def myDPalgo(problem):
    if mem[problem] not blank:
        return mem[problem]
    if baseCase(problem):
        solution = solve(problem)
        mem[problem] = solution
        return solution
    for subproblem of problem:
        subsolutions.append(myDPalgo(subproblem))
    solution = OptimalSubstructure(subsolutions)
    mem[problem] = solution
    return solution
```

DP Algorithms so far

- $2 \times n$ domino tiling (Fibonacci)
- Log cutting
- Matrix Chaining
- Longest Common Subsequence
- Seam Carving (Unit C Programming Problem)

Domino Tiling

```
Tile(n):
    Initialize Memory M
    M[0] = 0
    M[1] = 0
    for i = 0 to n:
        M[i] = M[i-1] + M[i-2]
    return M[n]
```



Log Cutting

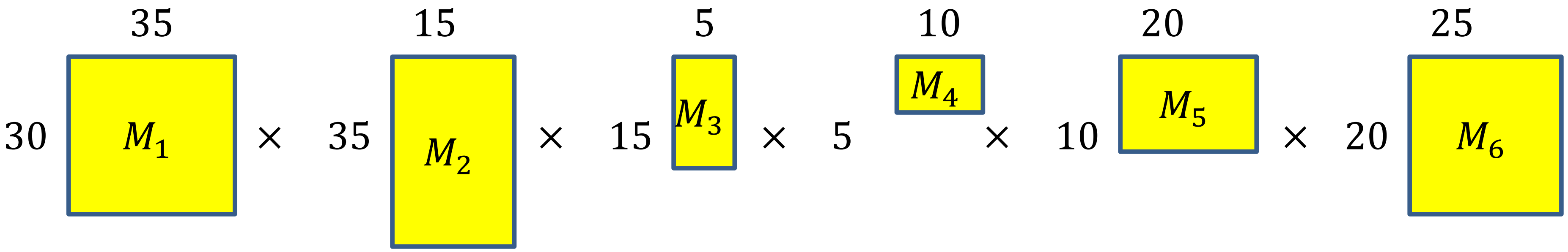
Solve Smallest subproblem first

$$Cut(4) = \max \begin{cases} Cut(3) + P[1] \\ Cut(2) + P[2] \\ Cut(1) + P[3] \\ Cut(0) + P[4] \end{cases}$$

$Cut(i)$:	0										
Length:	0	1	2	3	4	5	6	7	8	9	10

4

Matrix Chaining



$$Best(i,j) = \min_{k=1}^{j-1}\left(Best(i,k) + Best(k+1,j) + r_i r_{k+1} c_j\right)$$

$$Best(i,i) = 0$$

i \ j	1	2	3	4	5	6
1	0	15750	7875	9375	11875	15125
2		0	2625	4375	7125	10500
3			0	750	2500	5375
4				0	1000	3500
5					0	5000
6						0

$$Best(1,6) = \min \begin{cases} Best(1,1) + Best(2,6) + r_1 r_2 c_6 \\ Best(1,2) + Best(3,6) + r_1 r_3 c_6 \\ Best(1,3) + Best(4,6) + r_1 r_4 c_6 \\ Best(1,4) + Best(5,6) + r_1 r_5 c_6 \\ Best(1,5) + Best(6,6) + r_1 r_6 c_6 \end{cases}$$

Longest Common Subsequence

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

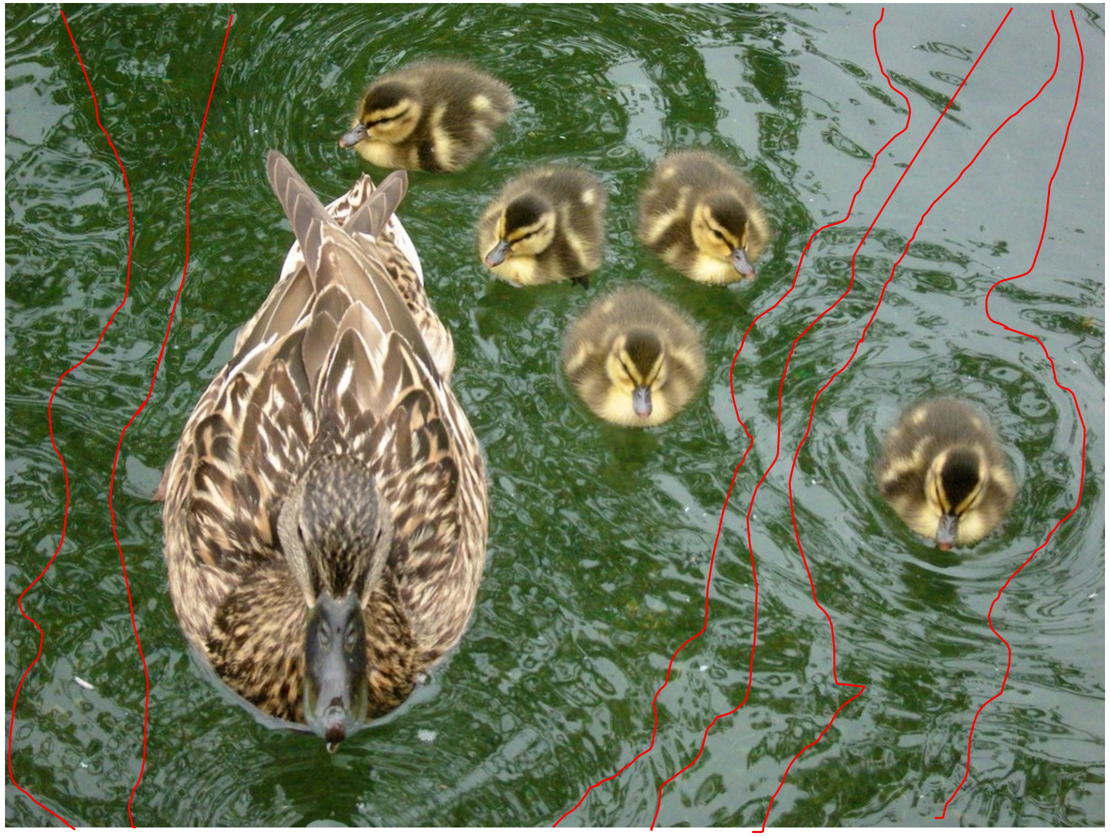
$X =$, $Y =$

		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**

To fill in cell (i, j) we need cells $(i-1, j-1), (i-1, j), (i, j-1)$
Fill from Top->Bottom, Left->Right (with any preference)

Seam Carving

- Removes "least energy seam" of pixels
- http://rsizr.com/, https://www.aryan.app/seam-carving/



Carved

Energy of a Seam

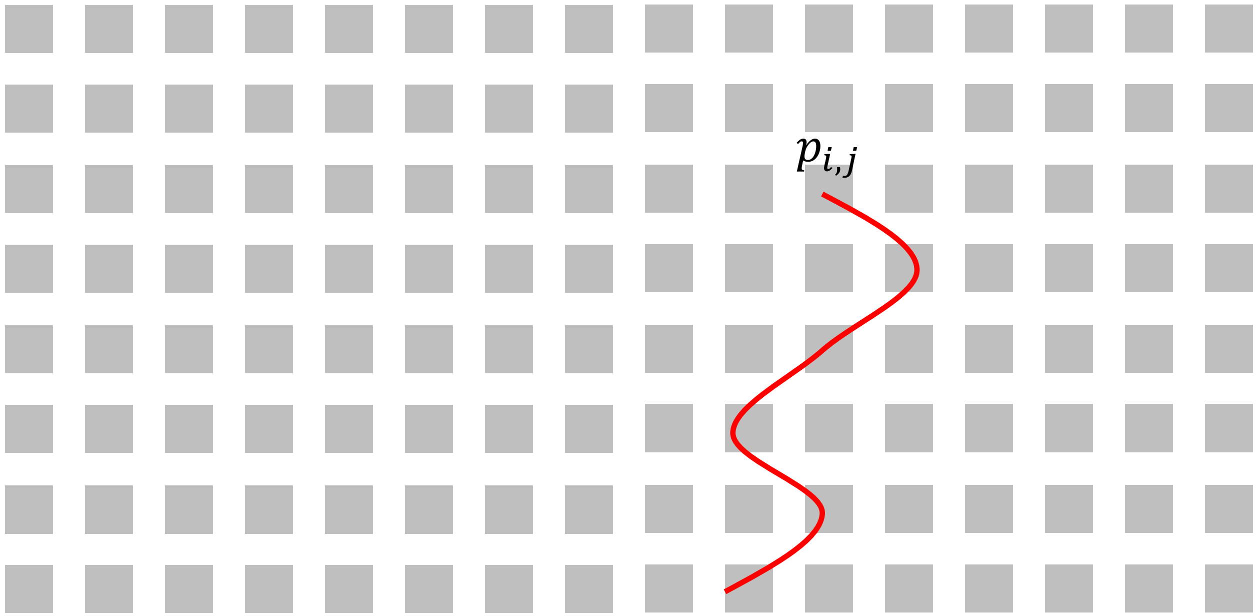
- Sum of the energies of each pixel
  - $e(p)$ = energy of pixel p
- Many choices
  - E.g.: change of gradient (how much the color of this pixel differs from its neighbors)
  - Particular choice doesn't matter, we use it as a "black box"

Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
  - Or: If S is an optimal solution to a problem, then the components of S are optimal solutions to sub-problems
- Idea:
  1. Identify the recursive structure of the problem
     - What is the “last thing” done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     - “Top Down”: Solve each recursively
     - “Bottom Up”: Iteratively solve smallest to largest

Identify Recursive Structure

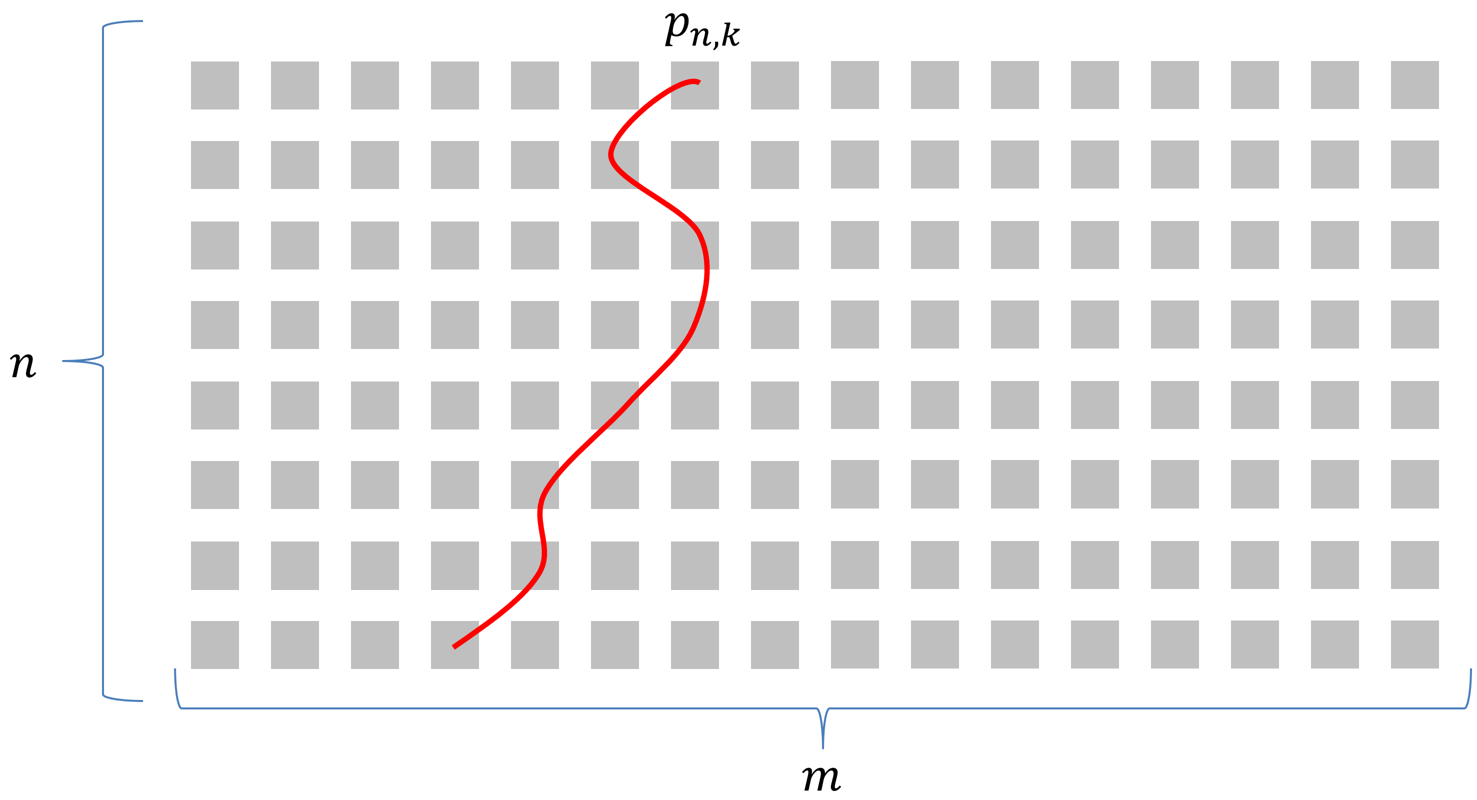
Let $S(i, j)$ = least energy seam from the bottom of the image up to pixel $p_{i,j}$



Finding the Least Energy Seam

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

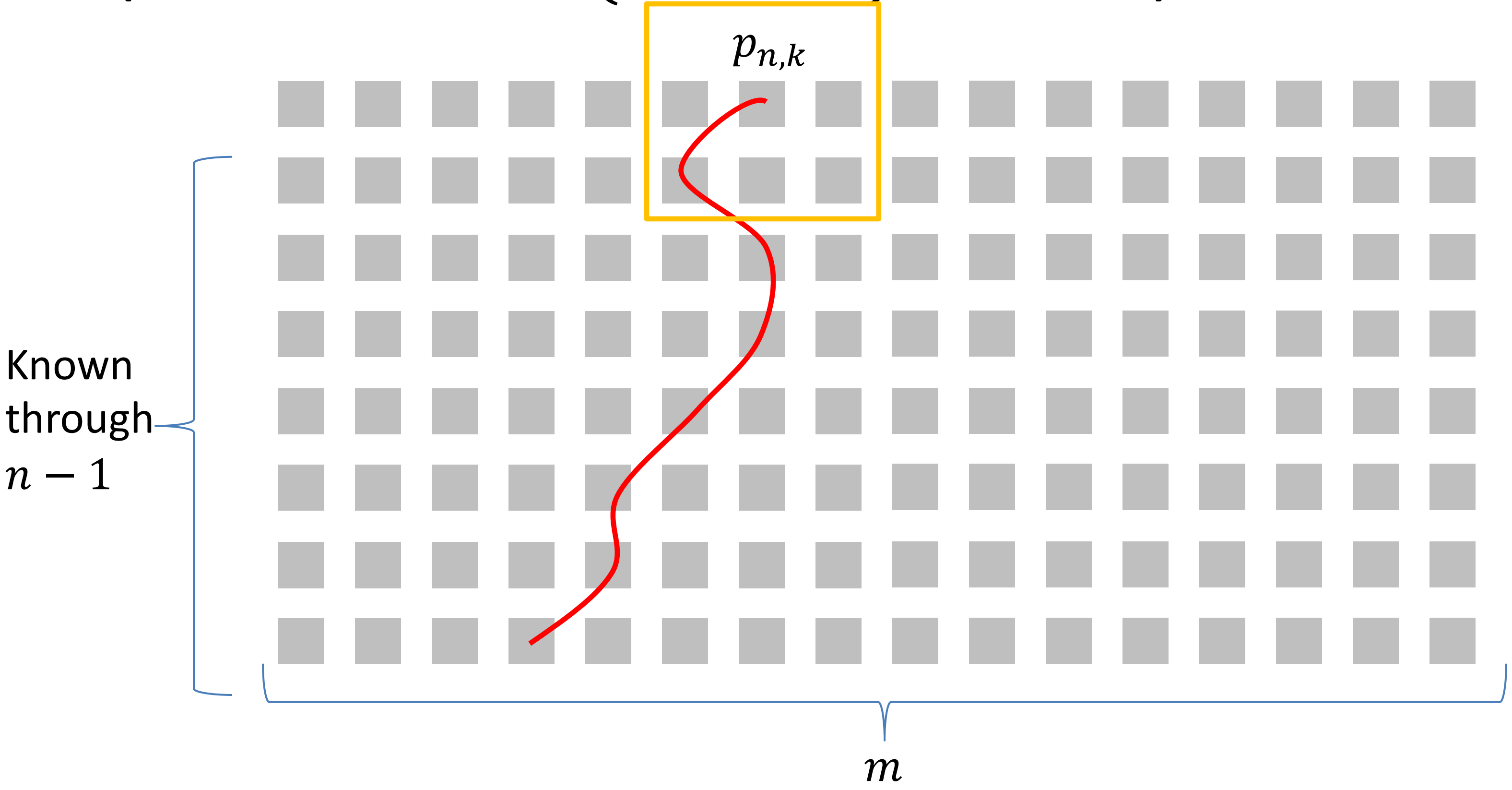
$$\min_{k=1}^{m}(S(n,k))$$



Computing $S(n,k)$

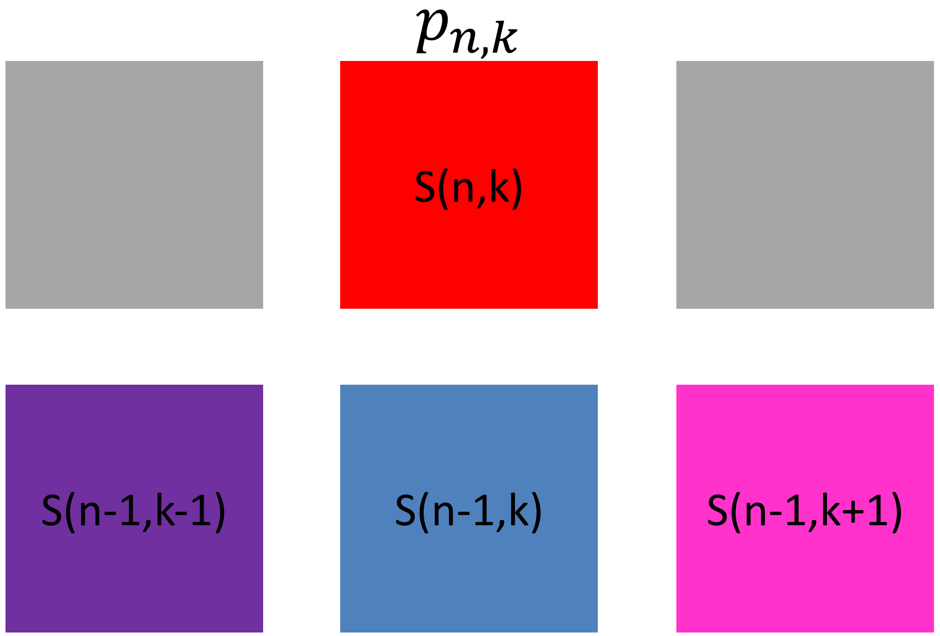
Assume we know the least energy seams for all of row $n-1$

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



Computing $S(n,k)$

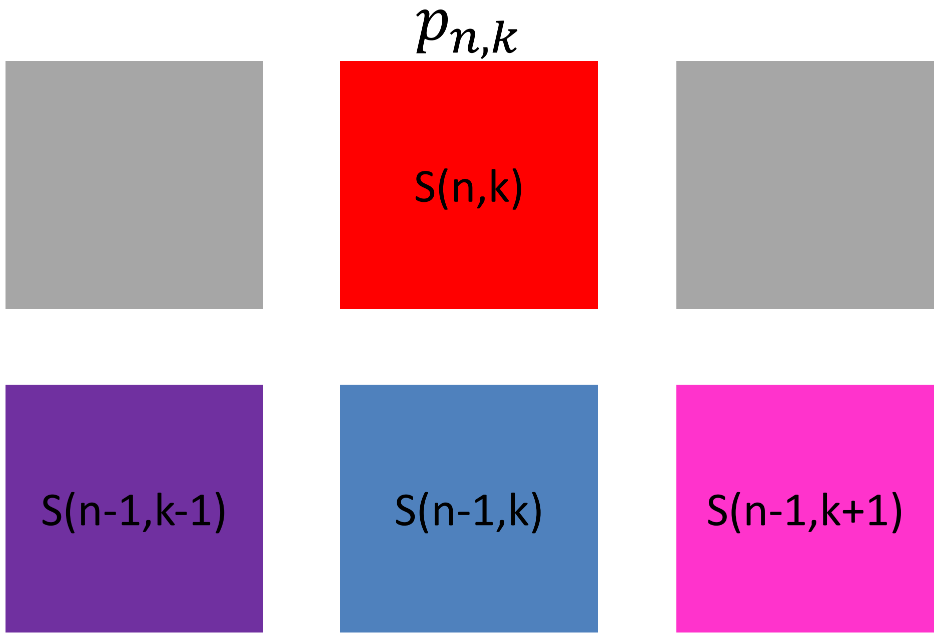
Assume we know the least energy seams for all of row $n-1$ (i.e. we know $S(n-1,\ell)$ for all ℓ)



Computing $S(n,k)$

Assume we know the least energy seams for all of row $n-1$ (i.e. we know $S(n-1,\ell)$ for all ℓ)

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



Seam Carving

- Details left to you! Unit C Programming assignment
  - Note: Python or Java implementations only this time

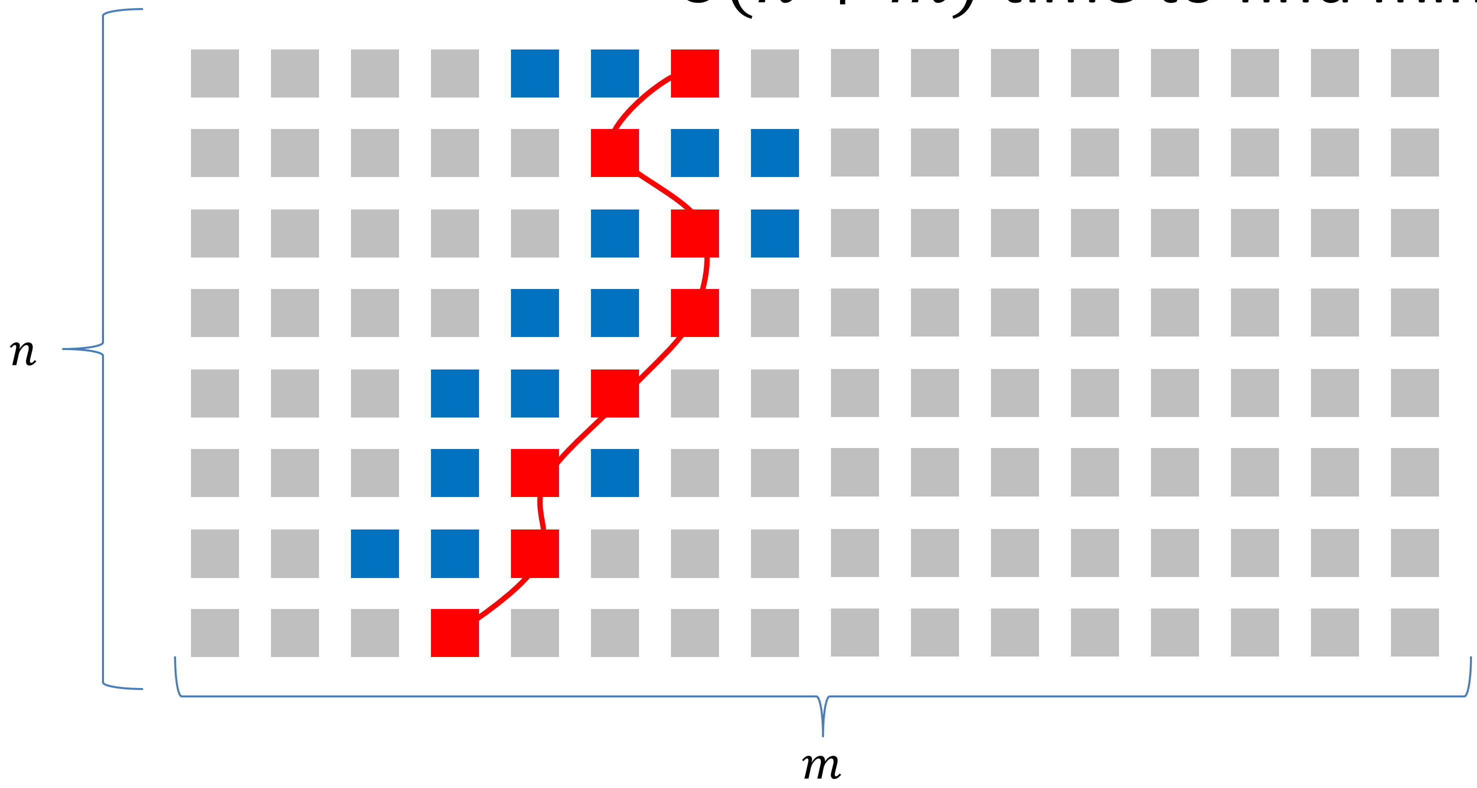
Repeated Seam Removal

Only need to update pixels dependent on the removed seam

$2n$ pixels change

$\Theta(2n)$ time to update pixels

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



Supreme Court Associate Justice Anthony Kennedy gave no sign that he has abandoned his view that extreme partisan gerrymandering might violate the Constitution. I 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.

SUPREME COURT OF THE UNITED STATES

Syllabus

VIRGINIA HOUSE OF DELEGATES ET AL. *v.*

SUPREME COURT OF THE UNITED STATES

Syllabus

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

Next Gerrymandering Battle in North Carolina: Congress

A North Carolina cou
an illegal gerrymande
state to redraw the sta

LAW

Supreme Court Rules Partisan Gerrymandering Is Beyond The Reach Of Federal Courts

June 27, 2019 · 10:17 AM ET
Heard on All Things Considered

NINA TOTENBERG | DOMENICO MONTANARO | MILES PARKS

4-Minute Listen | + PLAYLIST

Gerrymandering

- Manipulating electoral district boundaries to favor one political party over others
- Coined in an 1812 Political cartoon
- Governor **Elbridge Gerry** signed a bill that redistricted Massachusetts to benefit his Democratic-Republican Party

The Gerrymander

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

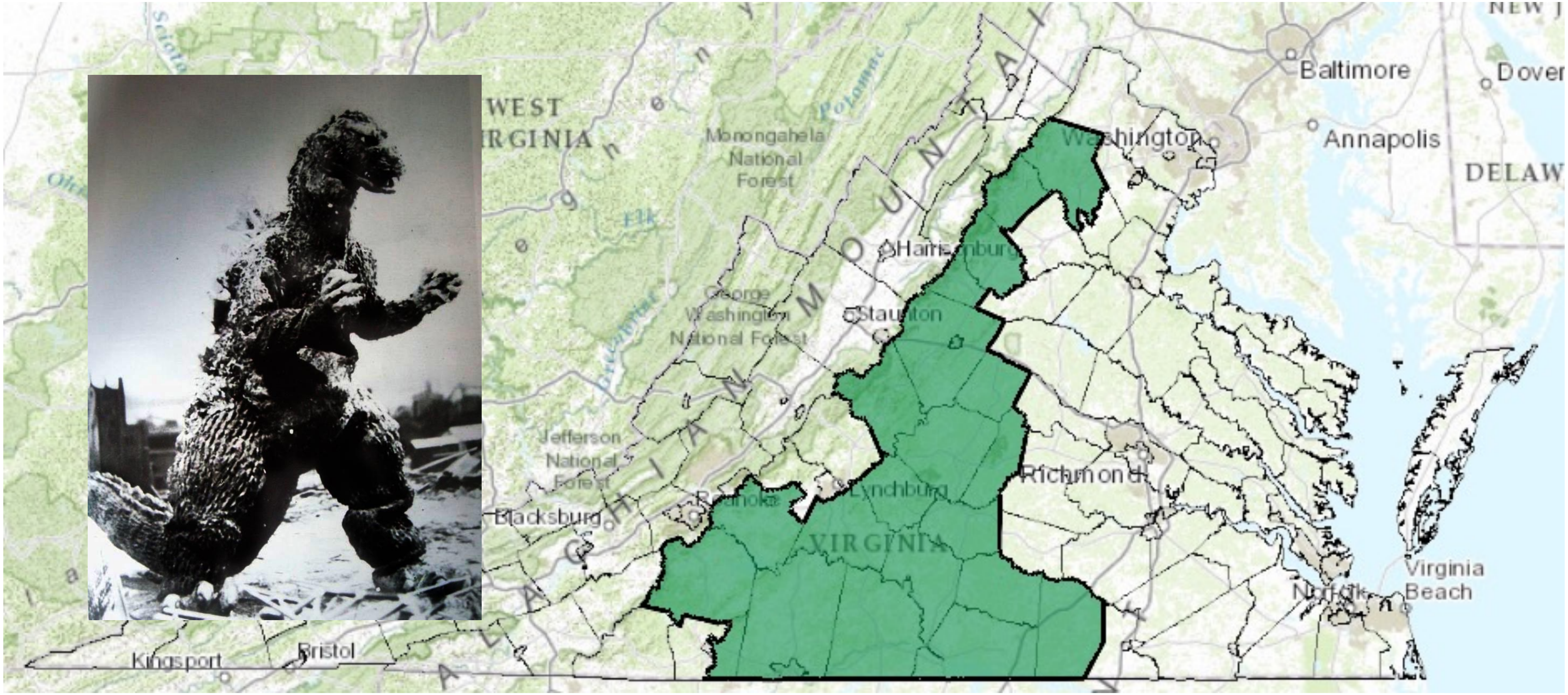
No. 18–422. Argued March 26, 2019—Decided June 27, 2019*

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

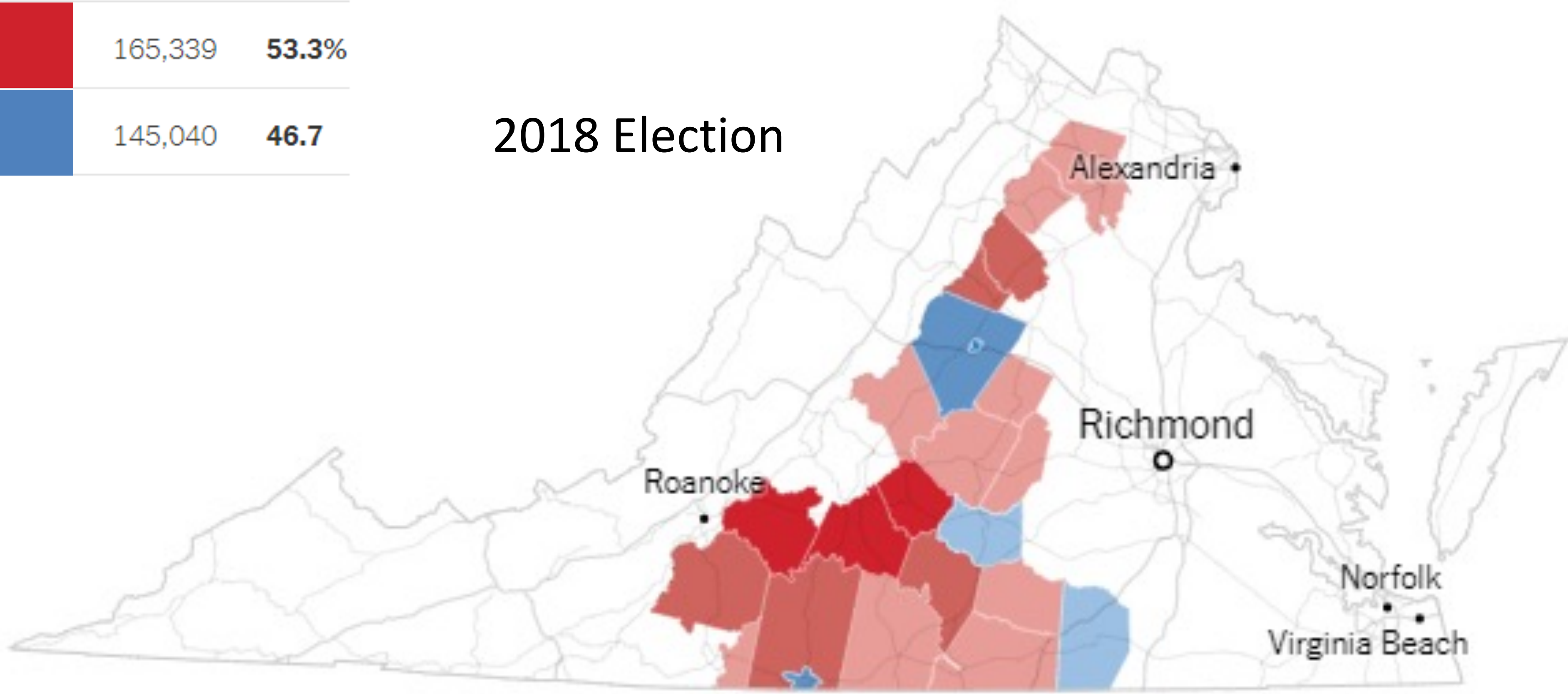
VA 5th District



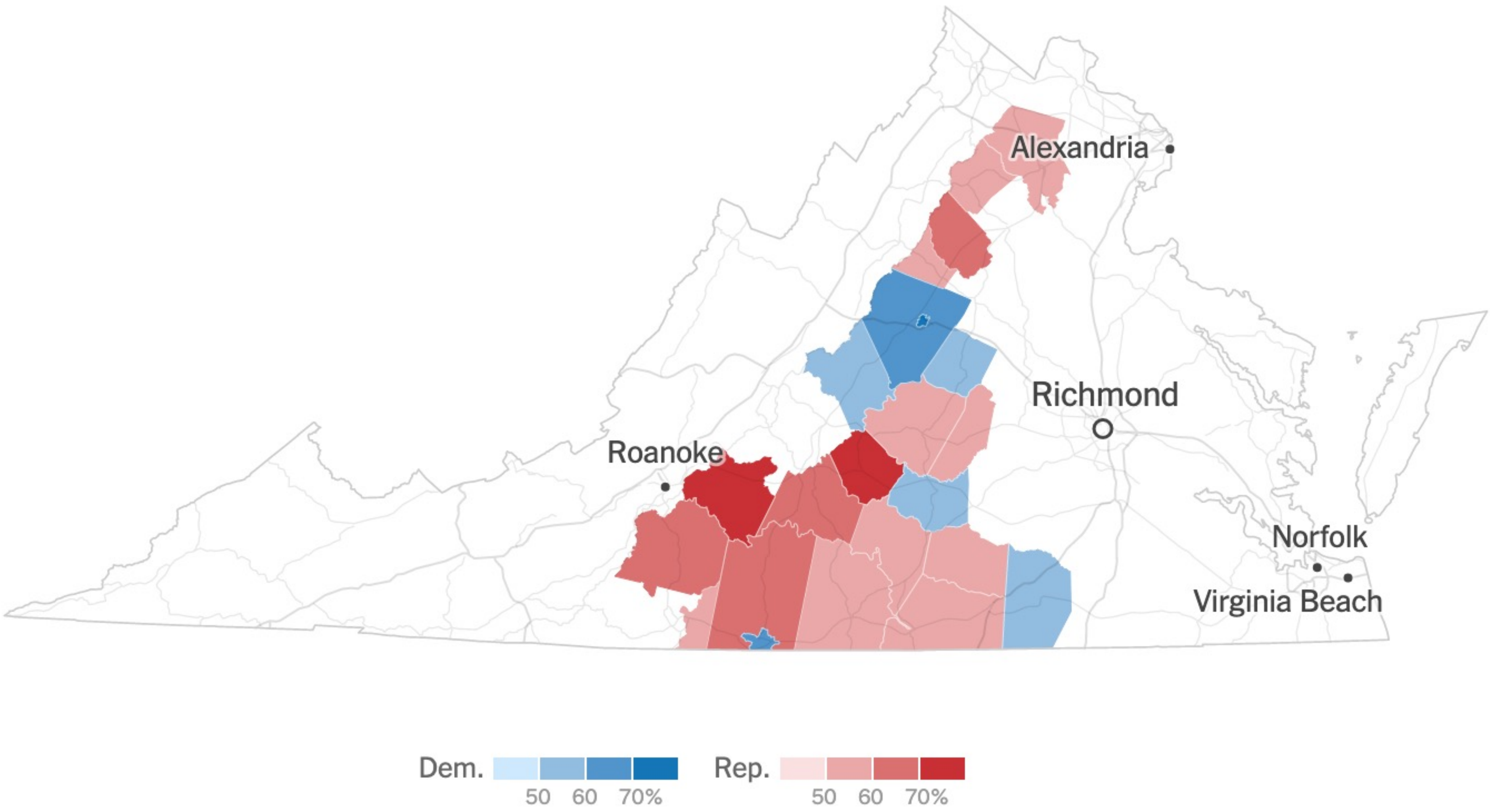
VA 5th District

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

2018 Election



5th District 2020 HR Results



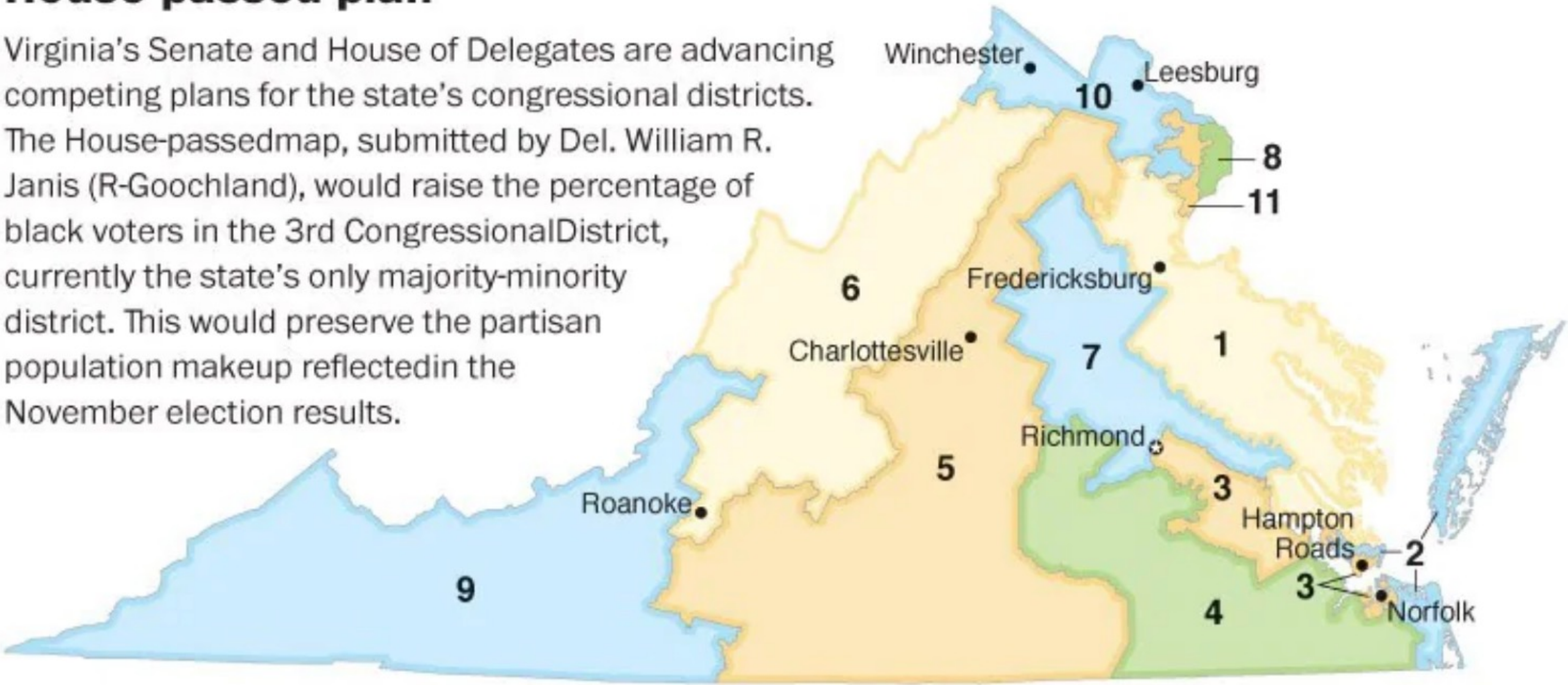
Some Thinking Before 2012 VA Congressional Redistricting

From: "Incumbents, not voters, shaping Virginia's congressional districts." Washington Post, 2011

Political Reality and 2012 VA Congressional Redistricting

House-passed plan

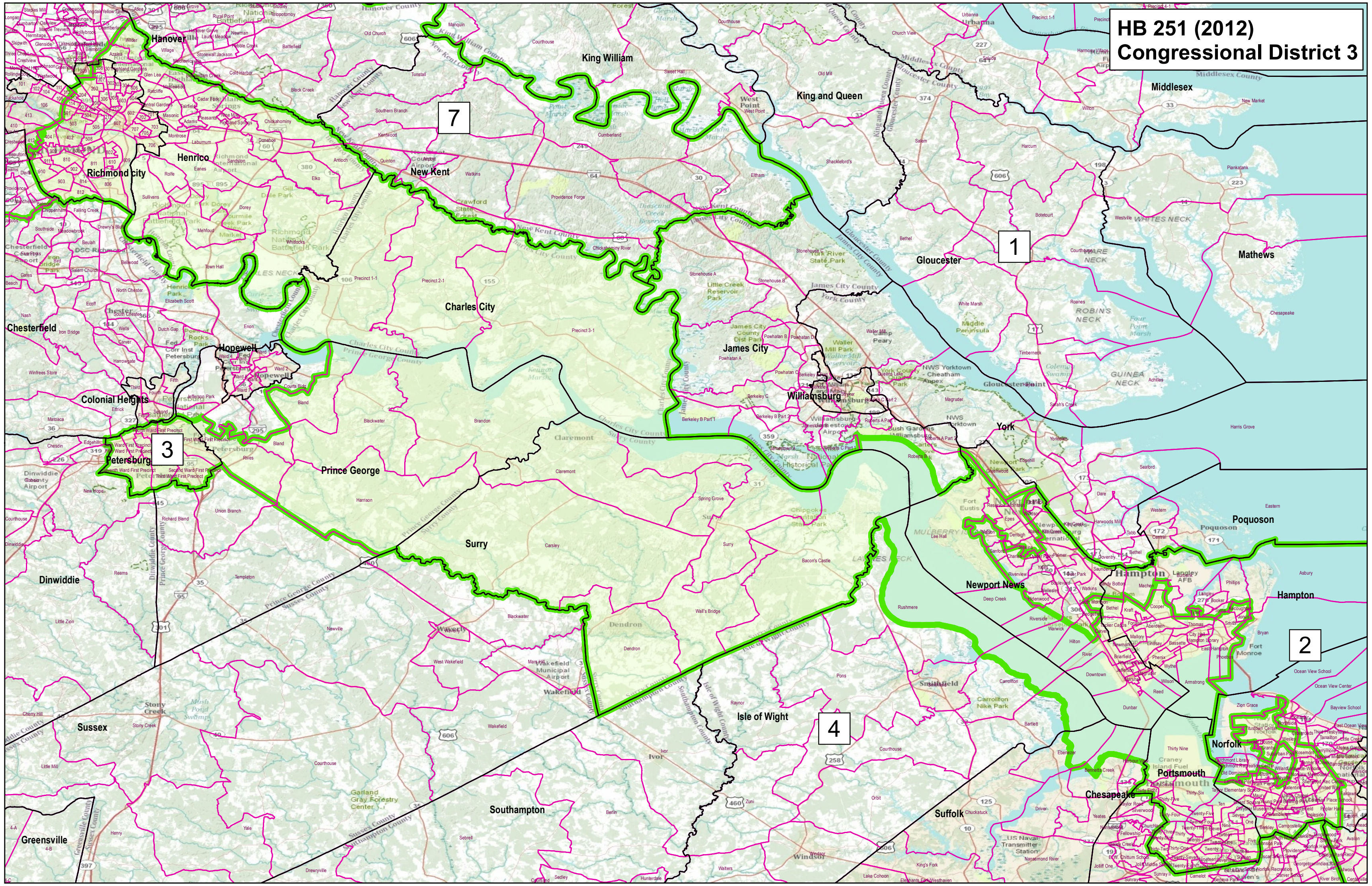
Virginia's Senate and House of Delegates are advancing competing plans for the state's congressional districts. The House-passedmap, submitted by Del. William R. Janis (R-Goochland), would raise the percentage of black voters in the 3rd CongressionalDistrict, currently the state's only majority-minority district. This would preserve the partisan population makeup reflectedin the November election results.



Christine Schoenberg April 15, 2011

Gerrymandering *Today*

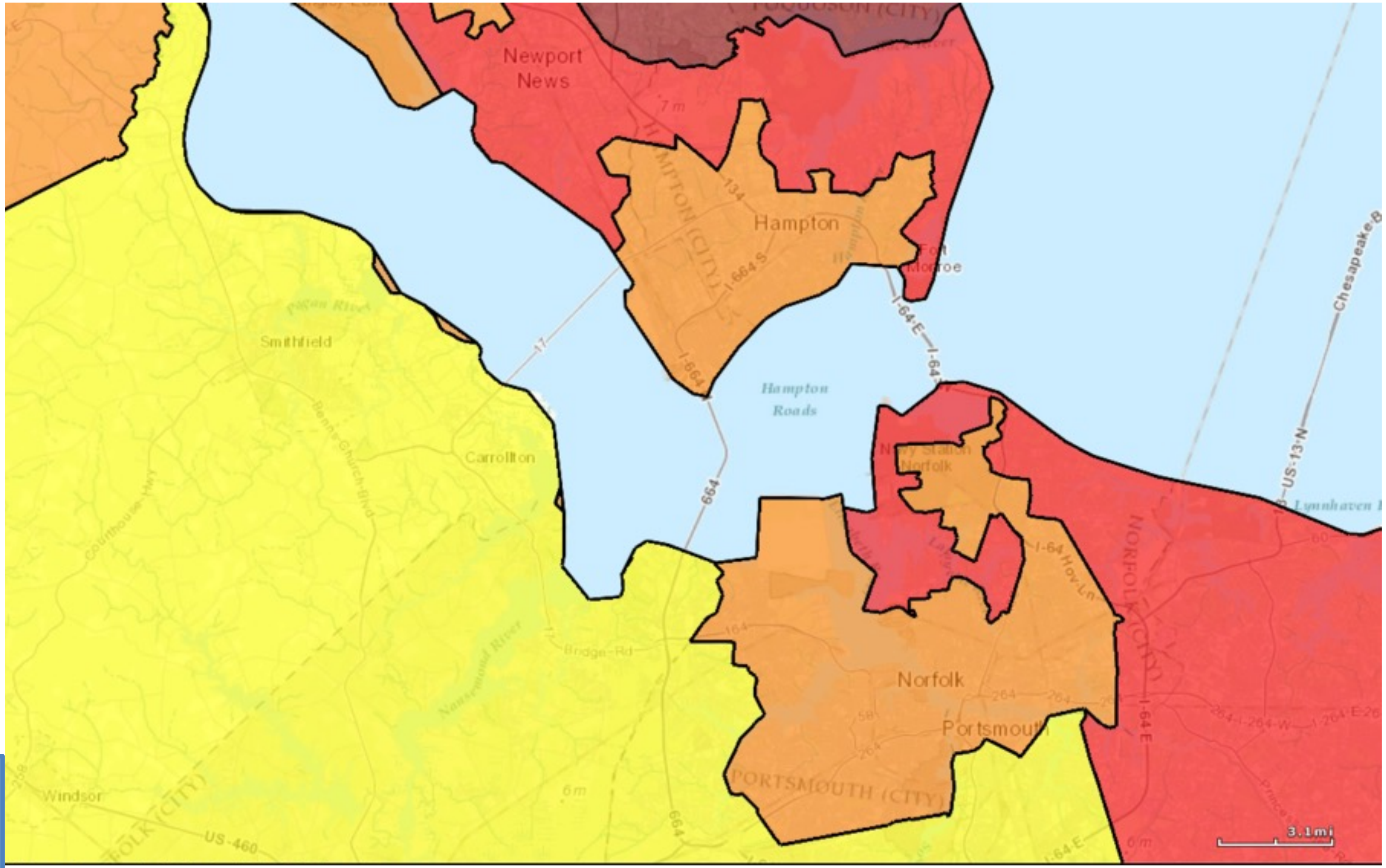
- Computers make it really effective



Gerrymandering *Today* – Seriously?

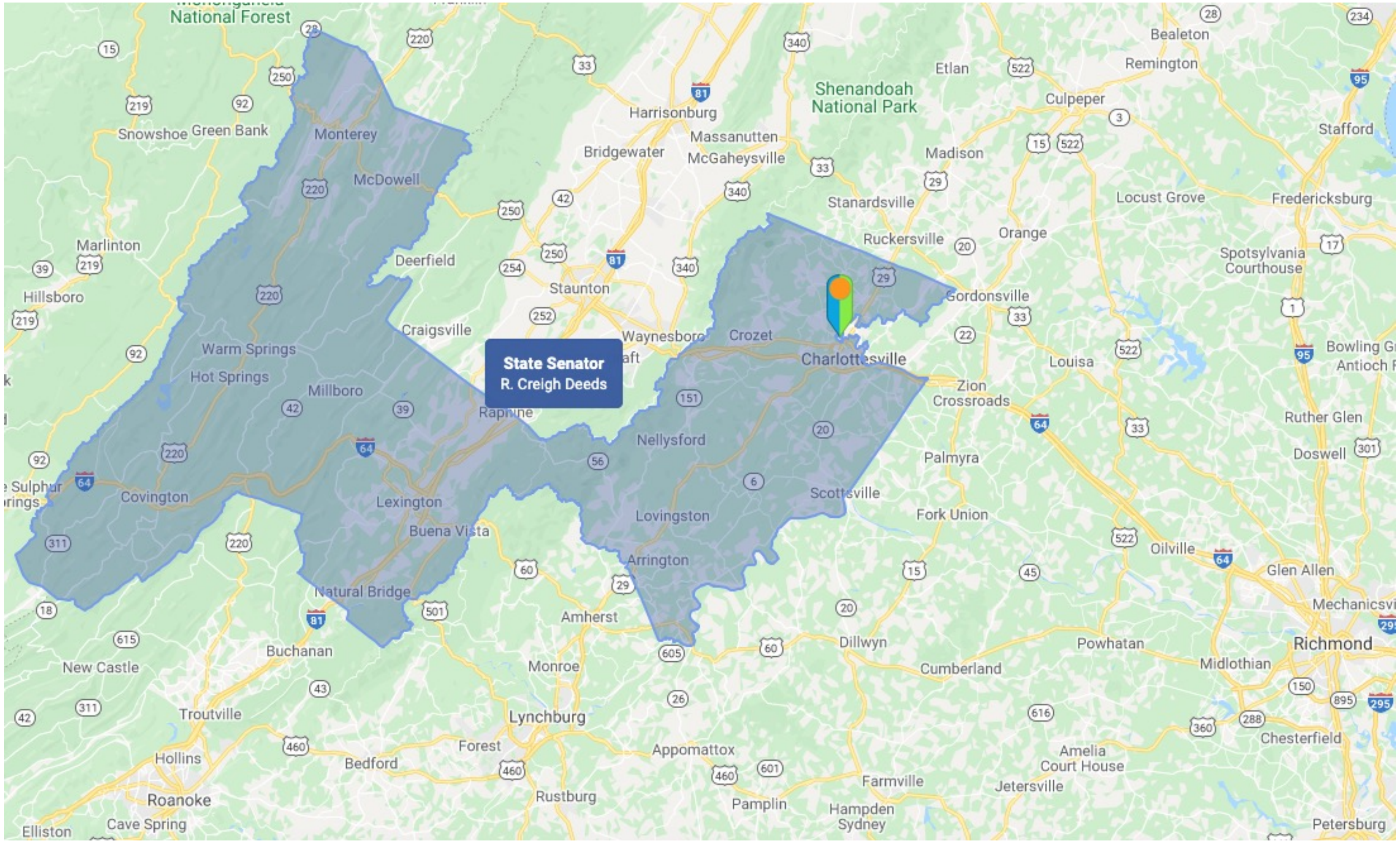
- Computers make it really effective
  - Close-up on part of 3rd District
  - This was 2013-2017; court ordered it changed

- Virginia will do redistricting soon under a new system



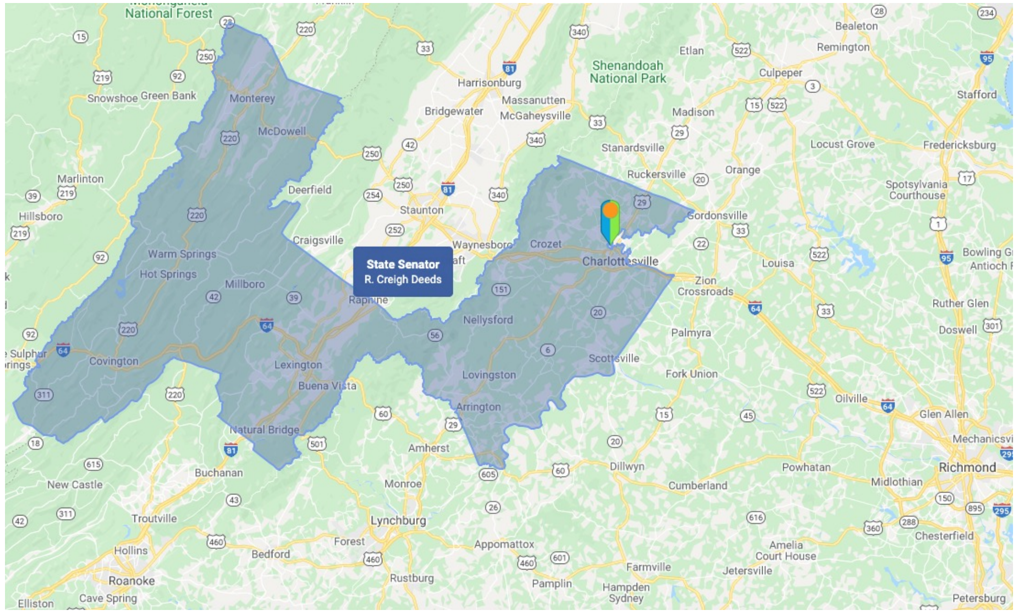
Learn More about VA redistricting:
https://www.vpap.org/redistricting/
https://redistricting.dls.virginia.gov/

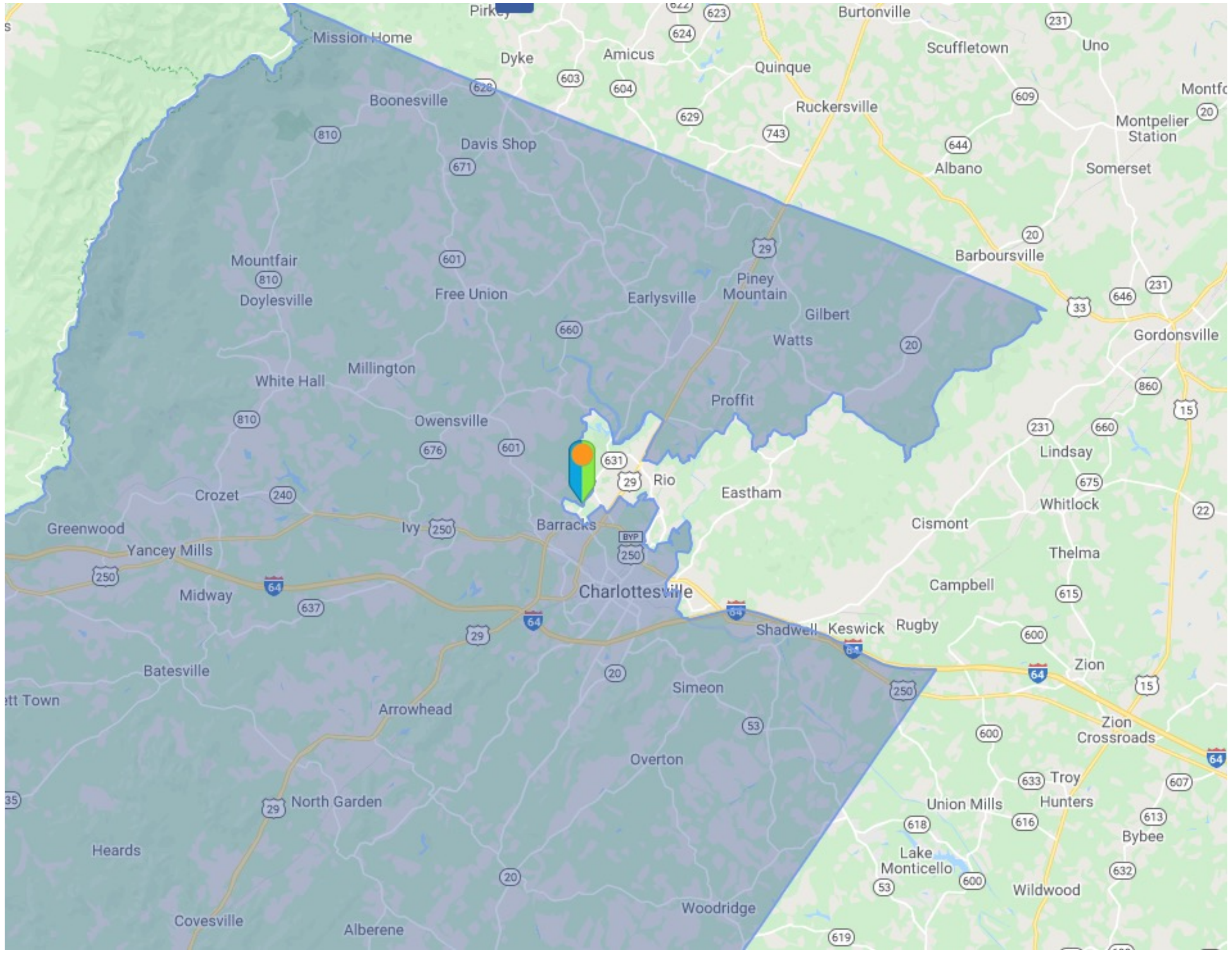
VA State Senate District 25 (2020)



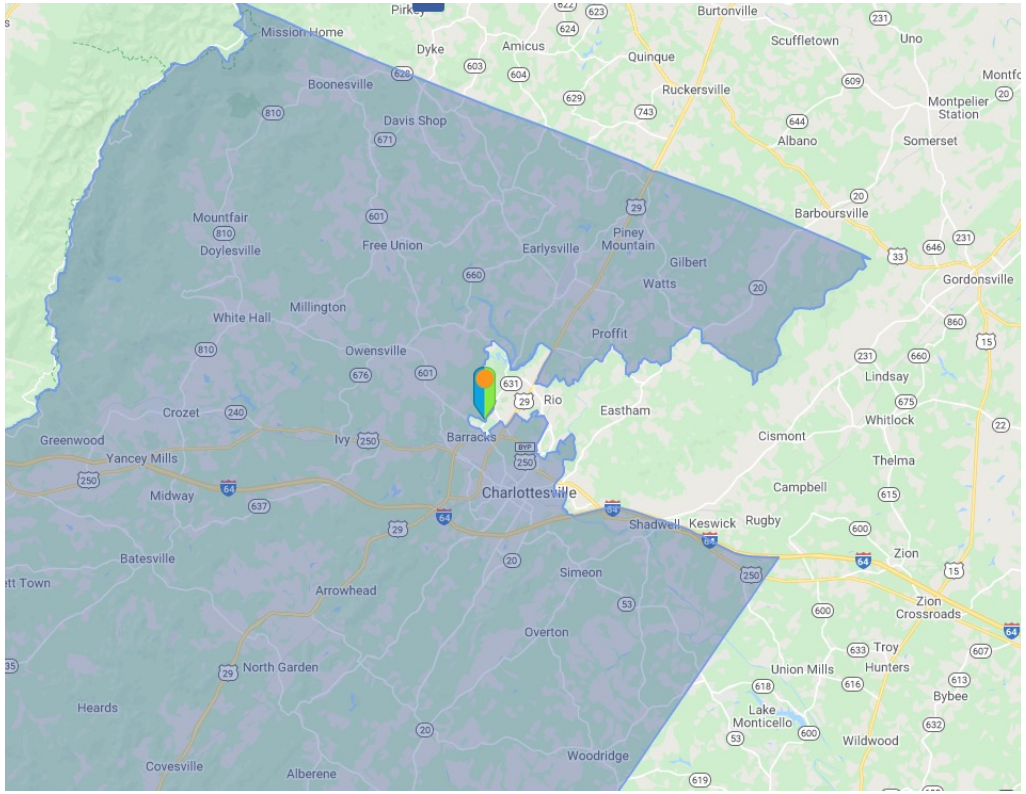
My house marked!

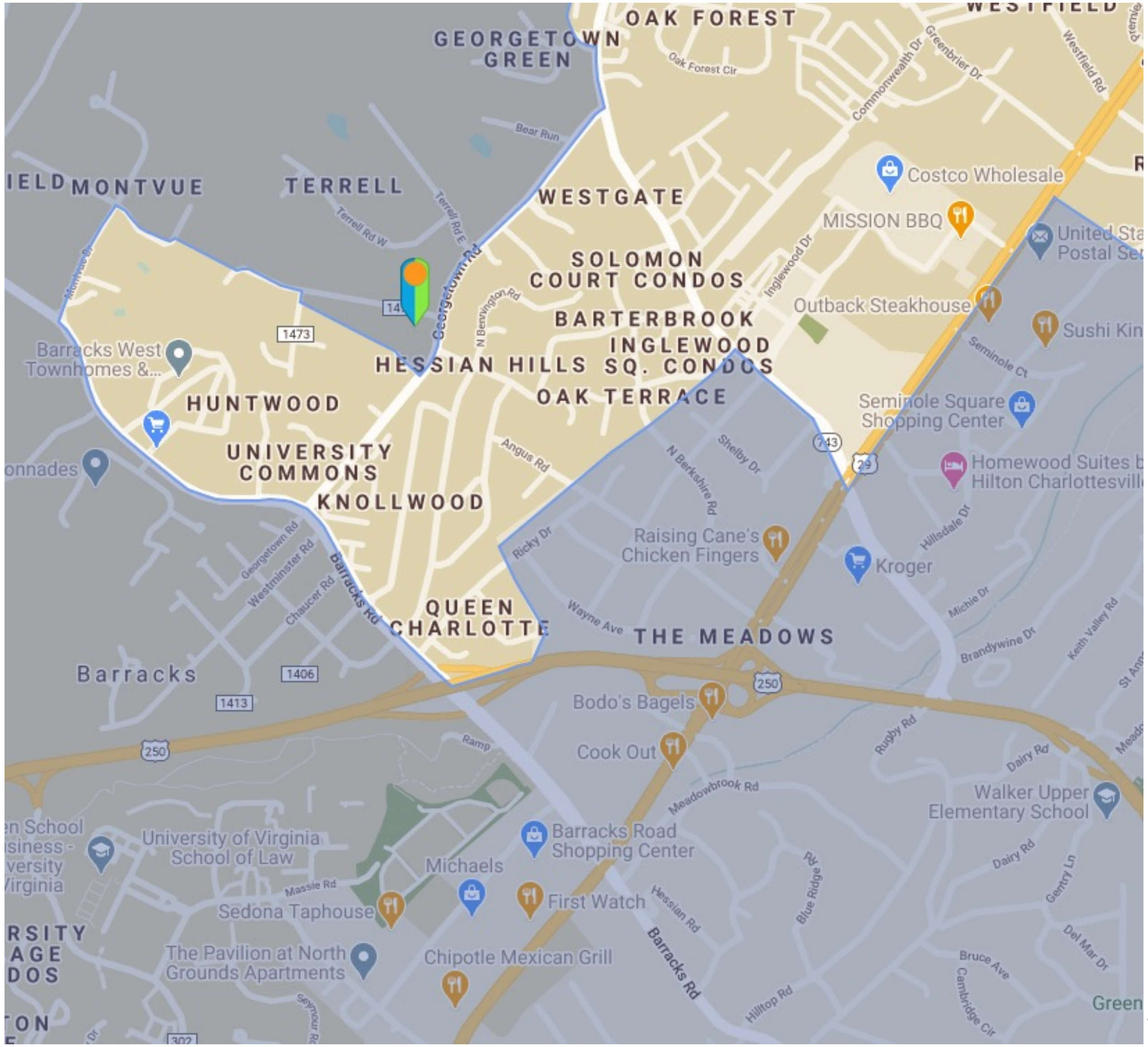
VA State Senate District 25 (2020)





VA State Senate District 25 (2020)

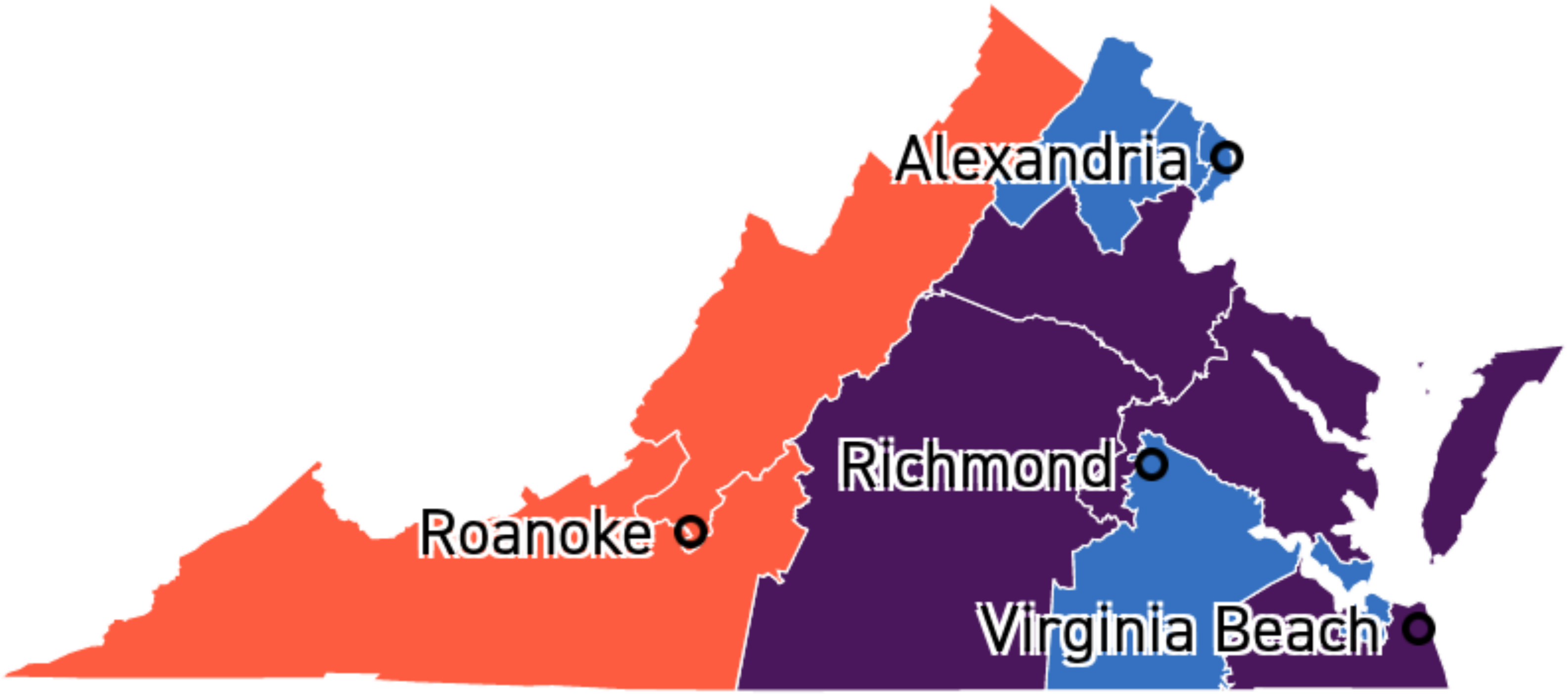




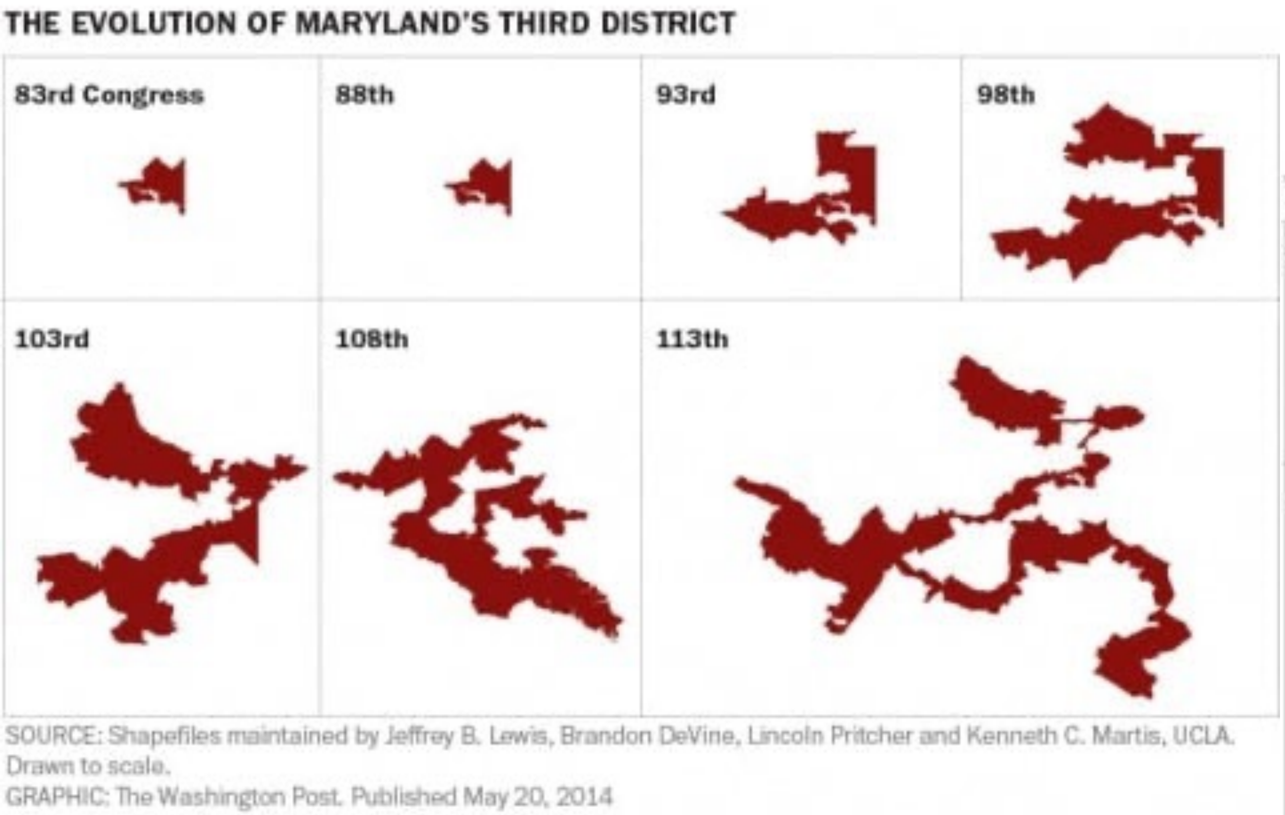
Learn More about VA redistricting:

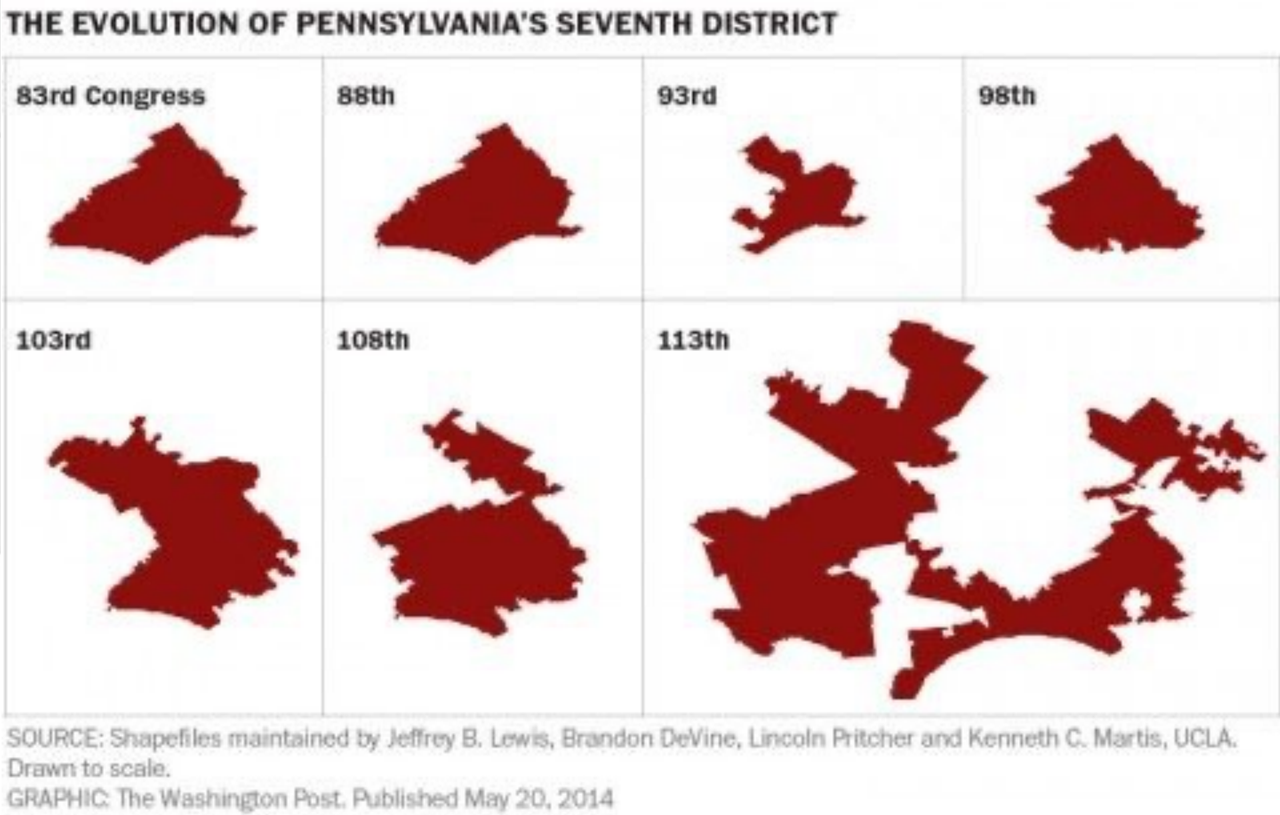
- https://www.vpap.org/redistricting/
- https://redistricting.dls.virginia.gov/

VA Redistricting – 2022-



Gerrymandering Today





An Algorithm to Gerrymander

It’s really a bit more complicated than this…

- States are broken into precincts
- All precincts have the same size
- We know the votes for 2 parties in each precinct
- Group precincts into districts to maximize the number of districts won by my party

Four precincts; 100 votes in each; Group into 2 districts

Overall: R:217 D:183

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

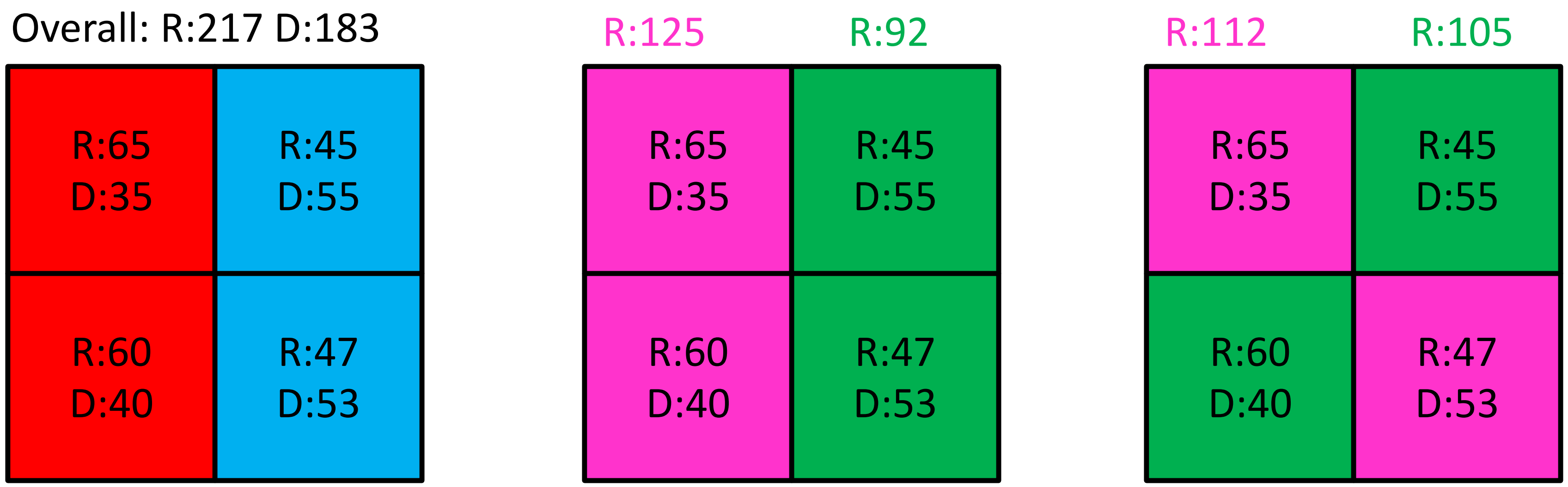
The “Regular” Party

VS

The “Diet” Party

How does it work?

- States are broken into precincts
- All precincts have the same size
- We know the votes for 2 parties in each precinct
- Group precincts into districts to maximize the number of districts won by my party



Gerrymandering Problem Statement

- Given:
  - A list of precincts $p_1, p_2, \dots, p_n$ and $R(p_i)$, number of votes for "Regular Party"
  - Each precinct has exactly m voters (So mn total voters)
- Output:
  - Two districts $D_1, D_2 \subset \{p_1, p_2, \dots, p_n\}$
  - Where $|D_1| = |D_2|$
    - So exactly $\frac{mn}{2}$ votes per district
  - $R(D_1) > \frac{mn}{4}$ and $R(D_2) > \frac{mn}{4}$
    - $R(D_i)$ gives number of "Regular Party" voters in D_i
    - $R(D_i) > \frac{\text{mn}}{4}$ means D_i is majority "Regular Party"
  - "failure" if no such solution is possible

Valid Gerrymandering: Both districts go to Regular Party!

More than 50% of the $\frac{mn}{2}$ votes

Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
- Idea:
  1. Identify the recursive structure of the problem
     - What is the "last thing" done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest

Dynamic Programming

- Requires Optimal Substructure
  - Solution to larger problem contains the solutions to smaller ones
- Idea:
  1. Identify the recursive structure of the problem
     - What is the "last thing" done?
  2. Save the solution to each subproblem in memory
  3. Select a good order for solving subproblems
     - "Top Down": Solve each recursively
     - "Bottom Up": Iteratively solve smallest to largest

Consider the last precinct

After assigning the first $n-1$ precincts $p_1, p_2, \dots, p_{n-1}$

D_1
k precincts
x voters for R

D_2
$n-k-1$ precincts
y voters for R

p_n

If we assign p_n to D_1

D_1
$k+1$ precincts
$x+R(p_n)$ voters for R

Valid gerrymandering if:

$$k+1=\frac{n}{2},$$

$$x+R(p_n), y > \frac{mn}{4}$$

If we assign p_n to D_2

D_2
$n-k$ precincts
$y+R(p_n)$ voters for R

Valid gerrymandering if:

$$\mathrm{n}-k=\frac{n}{2},$$

$$x, y+R(p_n) > \frac{mn}{4}$$

World One

D_1
$k+1$ precincts
$x+R(p_n)$ voters for R

D_2
$n-k-1$ precincts
y voters for R

World Two

D_1
k precincts
x voters for R

D_2
$n-k$ precincts
$y+R(p_n)$ voters for R

Define 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

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

4D Dynamic Programming!!!

Two ways to satisfy $S(j,k,x,y)$:

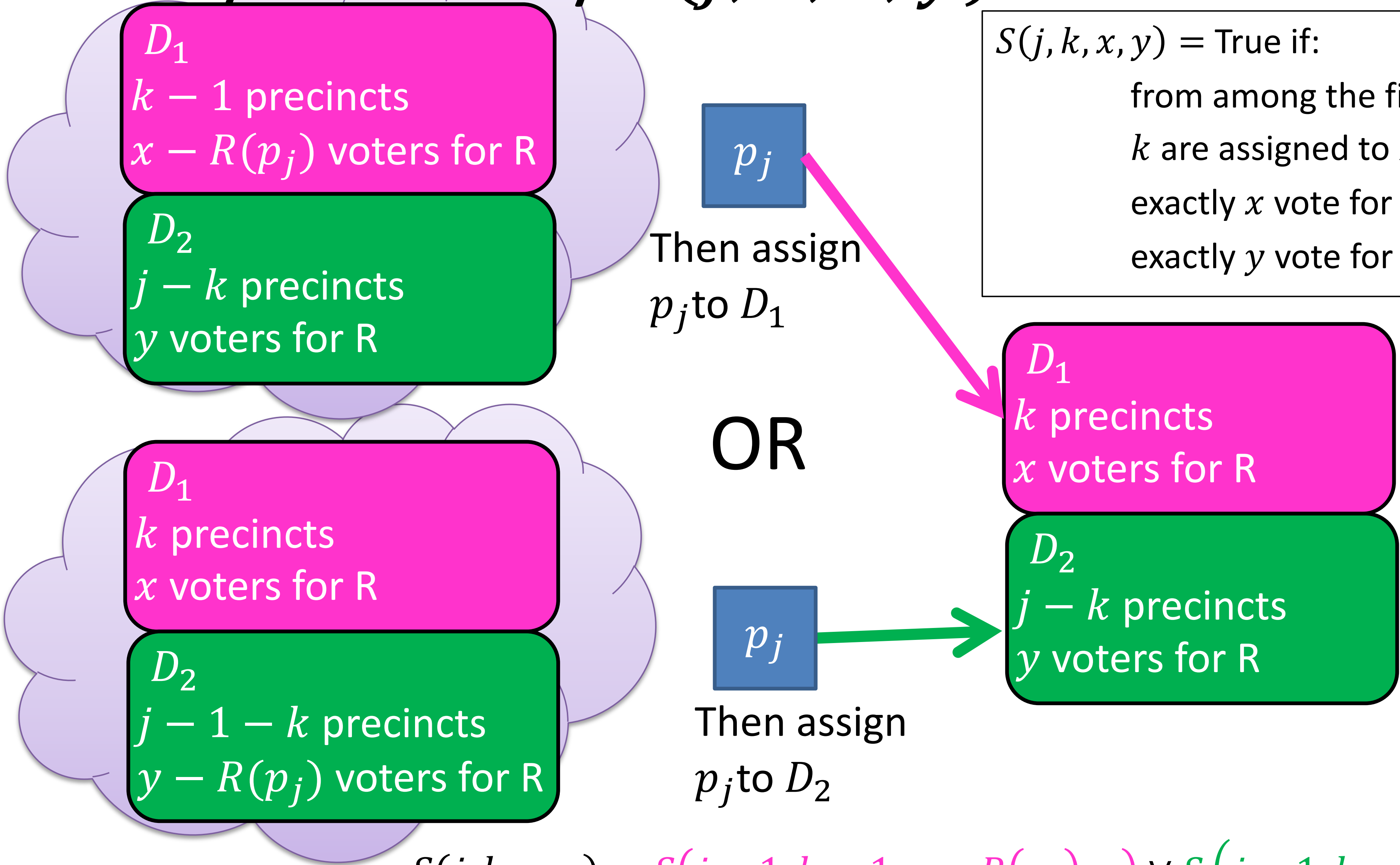
$S(j,k,x,y) =$ True if:

from among the first j precincts

k are assigned to D_1

exactly x vote for R in D_1

exactly y vote for R in D_2



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

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 $S(0,0,0,0) = \text{True}$
for $j = 1, \dots, n$:
  for $k = 1, \dots, \min(j, \frac{n}{2})$:
    for $x = 0, \dots, jm$:
      for $y = 0, \dots, jm$:
        $S(j,k,x,y) =$
        $S(j-1,k-1,x-R(p_j),y) \vee S\left(j-1,k,x,y-R(p_j)\right)$

Search for True entry at $S(n, \frac{n}{2}, > \frac{mn}{4}, > \frac{mn}{4})$

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

Run Time

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

Initialize $S(0,0,0,0) = \text{True}$

n for $j = 1, \ldots, n$:

$\frac{n}{2}$ for $k = 1, \ldots, \min(j, \frac{n}{2})$:

nm for $x = 0, \ldots, jm$:

nm for $y = 0, \ldots, jm$:

$S(j,k,x,y) =$

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

Search for True entry at $S(n, \frac{n}{2}, > \frac{mn}{4}, > \frac{mn}{4})$

$\Theta(n^4 m^2)$

Can We Visualize this 4D “Table”?

$S(j, k, x, y)$ = True if:

from among the first j precincts n

k are assigned to D_1 $n/2$

exactly x vote for R in D_1 nm

exactly y vote for R in D_2 nm

To get a solution: search for True entry at $S(n, \frac{n}{2}, > \frac{mn}{4}, > \frac{mn}{4})$

$\Theta(n^4 m^2)$

- This looks big! Yes, and it's interesting too! ☺
- Inputs:
  - List (size n) of precincts and counts of voters for Regular Party, $R(p_i)$
  - Number of voters (integer m)
- n is a size of one of the inputs
  - If n doubles, twice as many items in the list that's our input
- But m is an input value (not a size)
  - If m doubles, it's still one integer, one input item
  - But the amount of work grows
  - The complexity depends on the size of this single integer

Size of a Numeric Input-Value

Question: How do we measure the size of an integer?

Answer: the number of bits to represent it.

Example:
The value 4 (decimal) in binary is 100, so the size of “value 4” is 3.

If the size grows by 1, that’s 4 bits. With 4 bits, the value could be 1000 or 8 decimal.

Wait, what? Size of input grows by 1, and the value doubles (4 to 8).
That sounds like exponential! 2^n vs. 2^{n+1}

Pseudo-Polynomial Time

Yes, the *inputSize* (in bits) of value m is $\log_2 m$

$inputSize = \log_2 m$
$m = 2^{inputSize}$
So $m^2 = (2^{inputSize})^2 = 2^{2 \cdot inputSize}$

Gerrymandering's run-time is <u>exponential</u> because of *size* of input m

- Because run-time $\Theta(n^4 m^2)$ written in terms of the *value* of m, not the *size* of m
- Input size is really $n + |m| = n + \log m$

This is called **pseudo-polynomial time** (https://en.wikipedia.org/wiki/Pseudo-polynomial_time)
We've seen others like this! Knapsack DP $\Theta(n \cdot C)$ and Coin-changing DP $\Theta(n \cdot A)$