

PET PROGRAMS

RUSS PAUL-JONES

MAY 21, 2020

NORTH CREEK AND WOODINVILLE HIGH
SCHOOLS



Russ Paul-Jones Pet Programs
Reverse / Array Rotation / Mazes /
Sudoku / Word Search / Flood Fill /
Music / Mandelbrot Sets

PET PROGRAMS

- I suppose every programmer has some favorite programs that they are fond of over a long time. These are some of my pet programs, some of which I've been writing for over forty years. They are familiar programs that I can use to learn a new programming language or computer environment, and most of them have found their way into my collection of interview questions over the years. Let me tell you why these are some of my favorites, and I'll tell some stories behind these programs that have made them my special pets.
- I hope you walk away with the understanding that you, too, could have some pet programs to help you navigate your continued education and enjoyment with Computer Science.



RUSS PAUL-JONES

- I have been programming since 1977, when I found the computer lab at Cascade High School. I working with computers ever since, sometimes as a student, and mostly as a professional.
- I received a degree in Computer Science from the University of Washington in 1987.
- I have worked at companies in size from startups to Microsoft, as a programmer, architect, and development manager. I am currently working as a Development Manager at the Tableau Division of Salesforce.
- After all these years, elegant code still makes me very happy, and I am lucky that I get to work with it every day.



RUSS PAUL-JONES

- russ@paul-jones.org
- I'm on [LinkedIn](#)
- Github at <https://github.com/russpi>



WHAT ARE PET PROGRAMS FOR?

- They are just fun
- Learning a new programming language is easier if you have a familiar project to use as a tutorial
- I've collected interesting algorithms and data structures to use for interview questions
 - I like to written solutions to any questions I ask, since that gives me the experience to guide and evaluate the applicant
- And sometimes I've learned them in interview questions



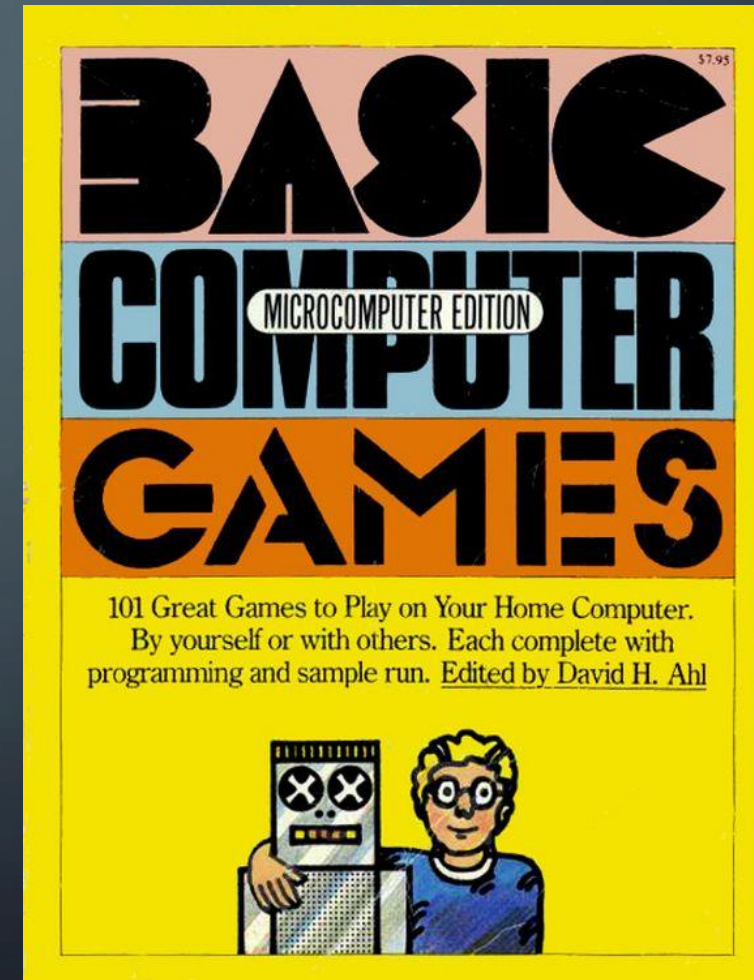
TABLE OF CONTENTS

- Maze programs
 - Maze Generators in BASIC
 - Maze Solver in BASIC
 - Learning Java with Mazes
- Array Rotation
 - The first interview question I invented
- Word Search in Python
- Flood Fill in Python
- Computer Music in C
- Mandelbrot Sets in C++



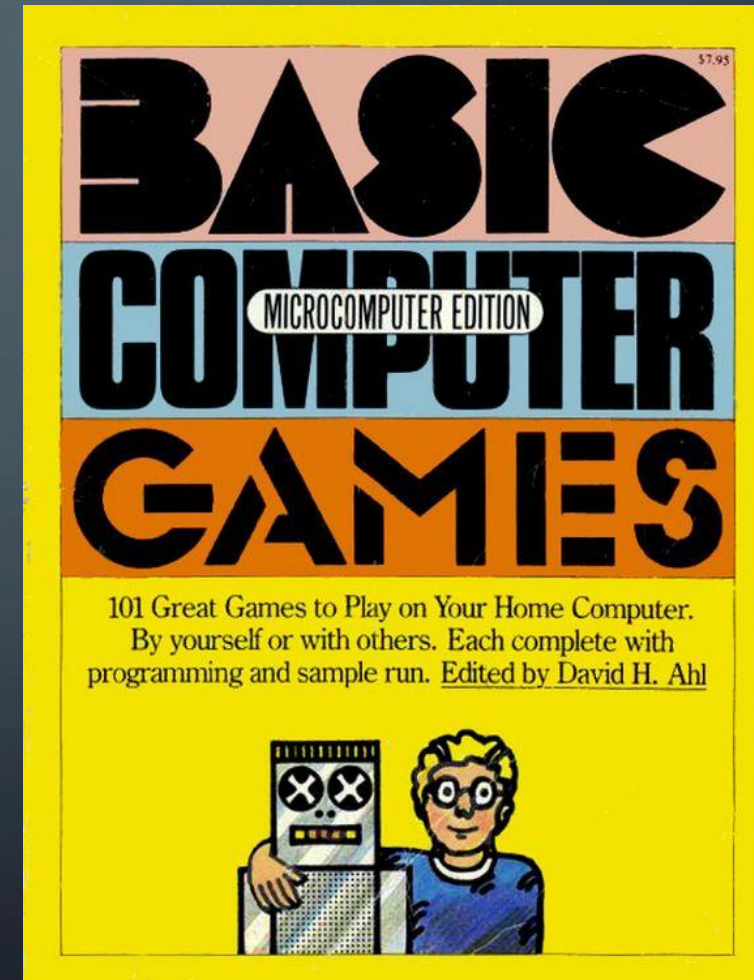
HIGH SCHOOL

- Diving into BASIC
 - We didn't have the internet, we had books
 - Number guessing, adventure games, Amazing
 - All text-based



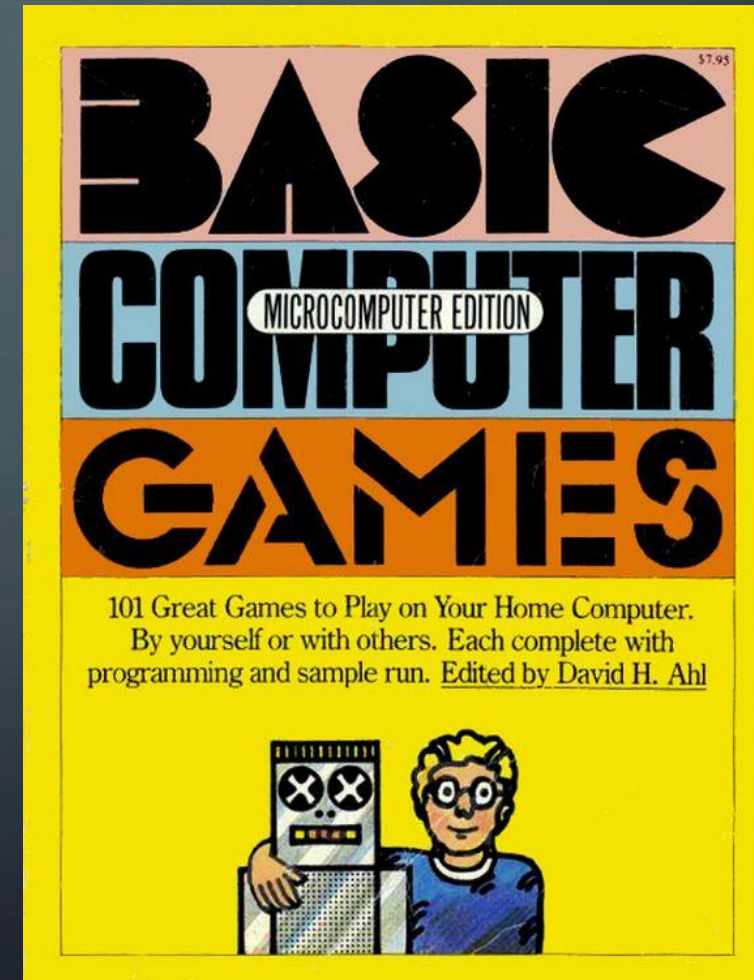
AMAZING

- The IBM-PC was released in 1981
- I had access to them through work, University, and friends
- Included BASIC programming language
- Now there is an [emulator](#) written for Windows
- I [ported](#) the version from the book to the IBM version of BASIC
- But I never understood the algorithm (the code is very hard to read)
- Demo with original teletype friendly version



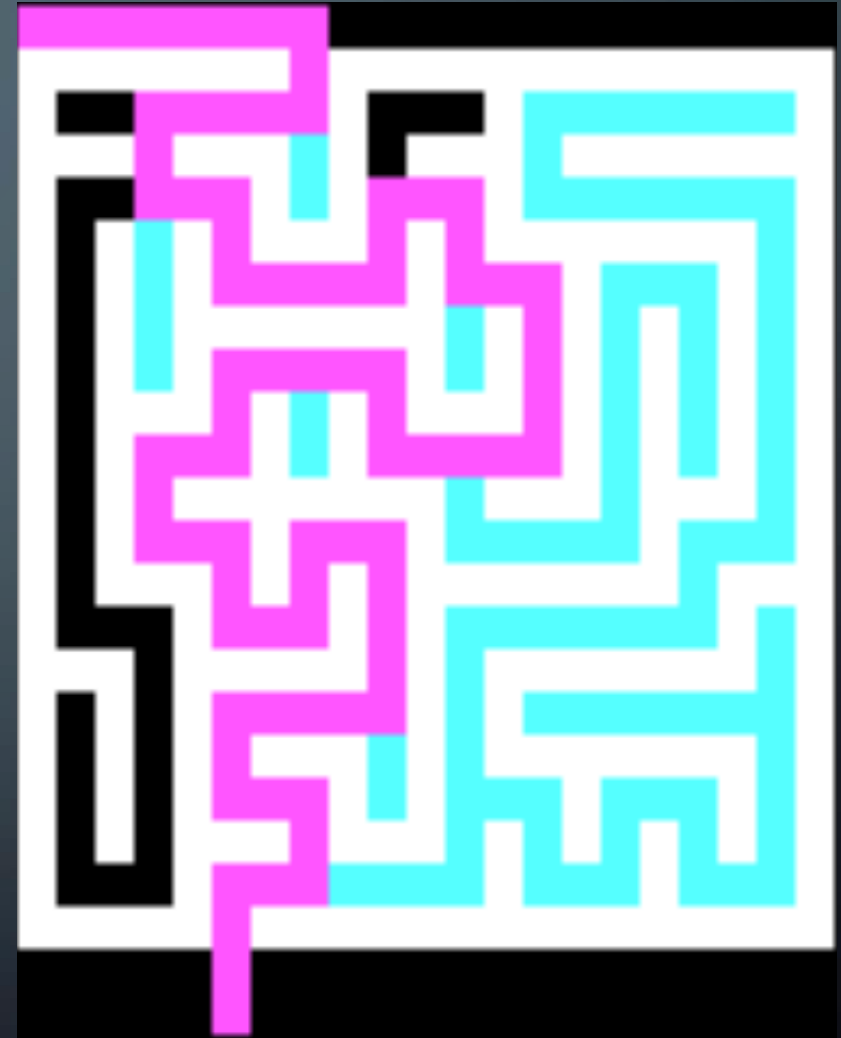
AMAZING

- Around 1984 I converted it to use the screen graphics available at the time
- 320x200 pixels, four glorious colors
- And I added my own code, to allow the computer to solve the maze
- It's often easier to understand your own code
- Demonstration



MAZE SOLVER

- Solving a maze is an example of a search with backtracking
 - At each junction, follow one of the paths you have not tried before
 - If you get to a dead end, back up to the previous junction
 - If there are paths you haven't tried, try one.
 - If at any point, you get out, you have solved the maze!
 - Otherwise, keep backtracking and trying other solutions
 - Follow the right-hand wall is a way of automating that process
 - I use this at corn mazes to solve the maze without looking at the map



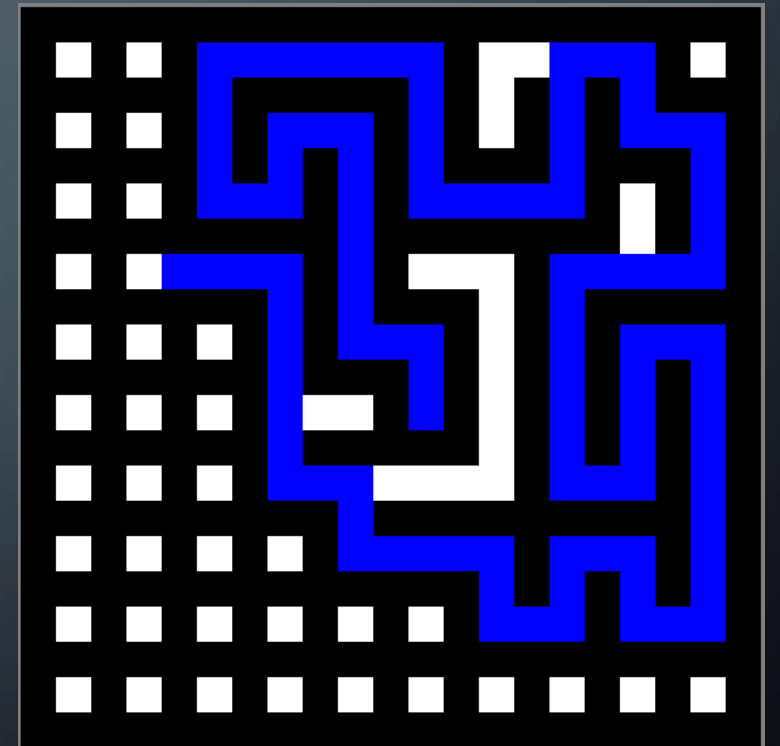
DR. STEPHEN GARLAND

- Last year, when I recovered my version of the BASIC Amazing program, I did some research on its origins
- The version from the book is very hard to read and understand, but I found another version with comments that explain what the variables are, and what steps the code sections represent
- I also found an additional author, S. Garland, and further research lead me to an email address for him
- I have had some interesting email conversations with him. He is still interested in mazes and follows the right hand wall when solving life-sized ones!



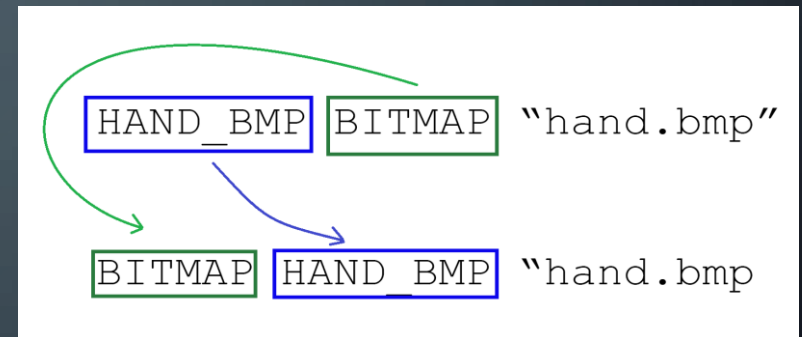
JAVA VERSION

- Recently I wanted to learn the Java programming language
- I had learned something about the theory behind mazes in University
 - Start with a set of rooms (*nodes*) that are not connected.
 - Start with some room. Test the rooms that are up/down, left/right. If one of those rooms has not been visited, make a passage (*edge*) to that room. Go into that room, and repeat.
 - If there are no unvisited rooms nearby, backtrack until you find a room with unvisited neighbors.
 - This creates a *spanning tree* of this *graph*, which is *singly-connected* (there is only one path from any two rooms in the maze)
- I wrote a [Java program](#) that created mazes (and solved them, of course)
- And just a few months later, I was asked the maze solving question in an interview
 - I told them I had been working on that for over 30 years!
- Demonstration



REVERSING AND ROTATION

- This is the first interview question I invented, around 1990 or so
- Based on an actual problem that I had at work:
 - We needed resource files for two different operating systems
 - Some lines were almost the same, just with the words in a different order
- Given a text string, *HAND_BMP BITMAP 'hand.bmp'*
- Produce the string, *BITMAP HAND_BMP 'hand.bmp'*
- I realized that if I could just rotate the characters in the string, I could convert one line into the other
- In those days, memory was dear, and computers were slow, so we tried not to use extra memory. The idea here was to do the rotation in place.



REVERSING AND ROTATION – WHY PYTHON?

- Python is an interesting programming language for scripting, text processing, and more general computing.
 - For example, that PC-BASIC interpreter I used earlier was written in Python
- I had already written these in Python before, but for these demonstrations, I wanted animations of the algorithms
- I found a library with good graphics that works across platforms
 - I even have these demos on my phone
- Many programming languages and environments are available on many platforms today, for free
- Demonstration



SUDOKU SOLVER IN PYTHON

- Sudoku can also be solved with a [backtracking program](#)
- Rules of Sudoku
 - Sudoku is a 9x9 grid, split up into 9 3x3 squares
 - You place a number between 1 and 9 in each location on the grid
 - Numbers cannot match another on their row
 - Numbers cannot match another in their column
 - Numbers cannot match another in their 3x3 square

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

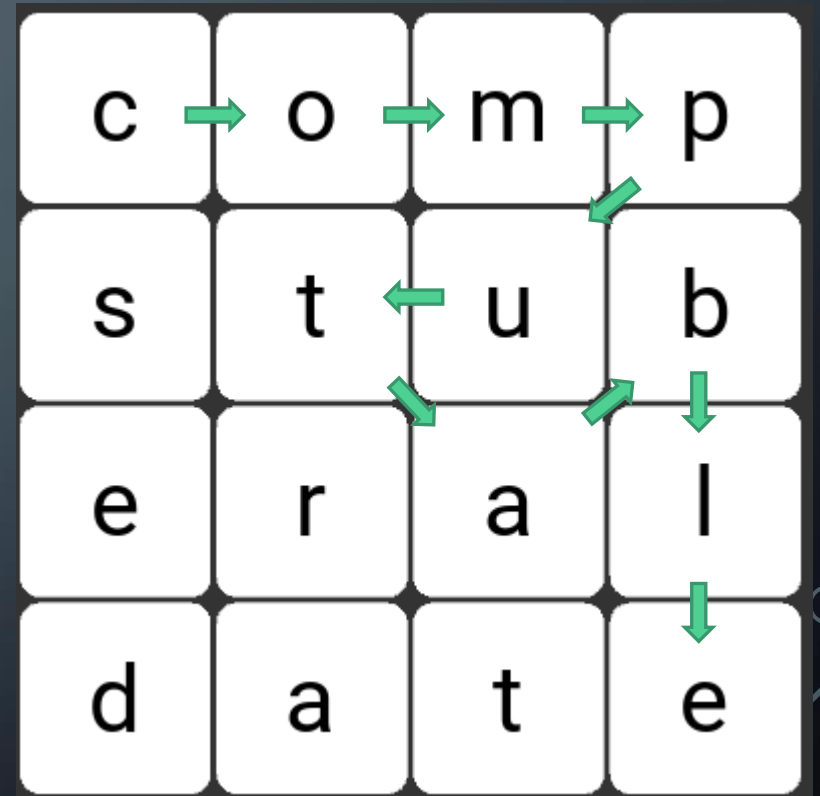
SUDOKU SOLVER IN PYTHON

- Python is an interesting programming language for scripting, text processing, and more general computing.
 - For example, that PC-BASIC interpreter I used earlier was written in Python
- I had written some of these up in Python before, but for these demonstrations, I wanted animations of the algorithms
- I found a library with good graphics that works across platforms
 - I even have these demos on my phone
- The lesson here is that today, many programming languages and environments are available on many platforms
- Demonstration

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

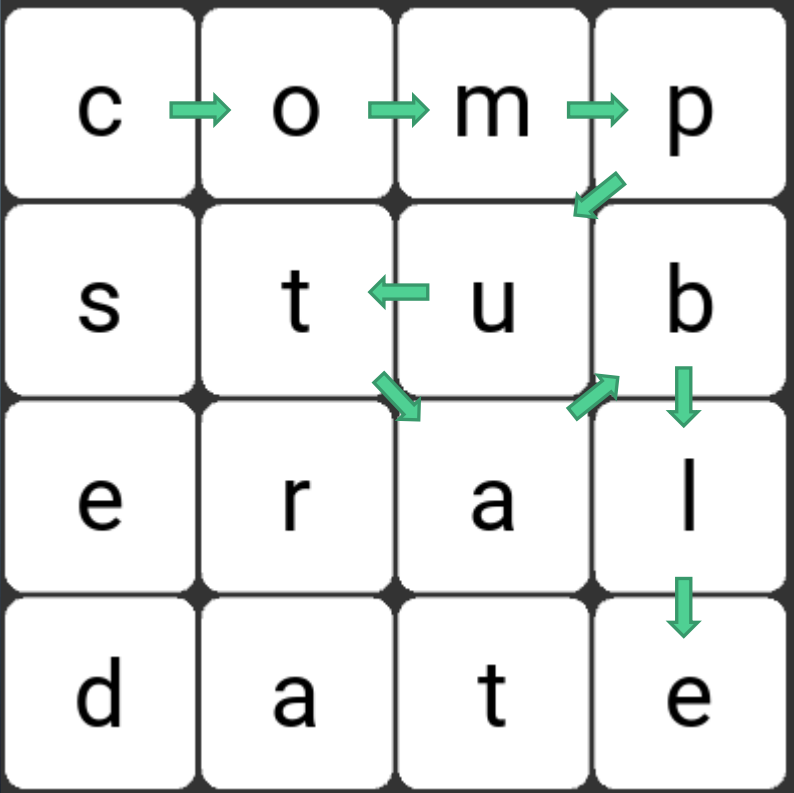
WORD SEARCH IN PYTHON

- Boggle is a word search game
- Rules
 - 16 letter blocks are placed at random on a 4x4 grid
 - You can connect letters up/down, left/right, and diagonally
 - The goal is to connect letters to form real words
 - Scoring is based on the length of the word, or the value of the letters
 - WordBlitz is a variation of this game on Facebook; I play Wordament on my Android phone

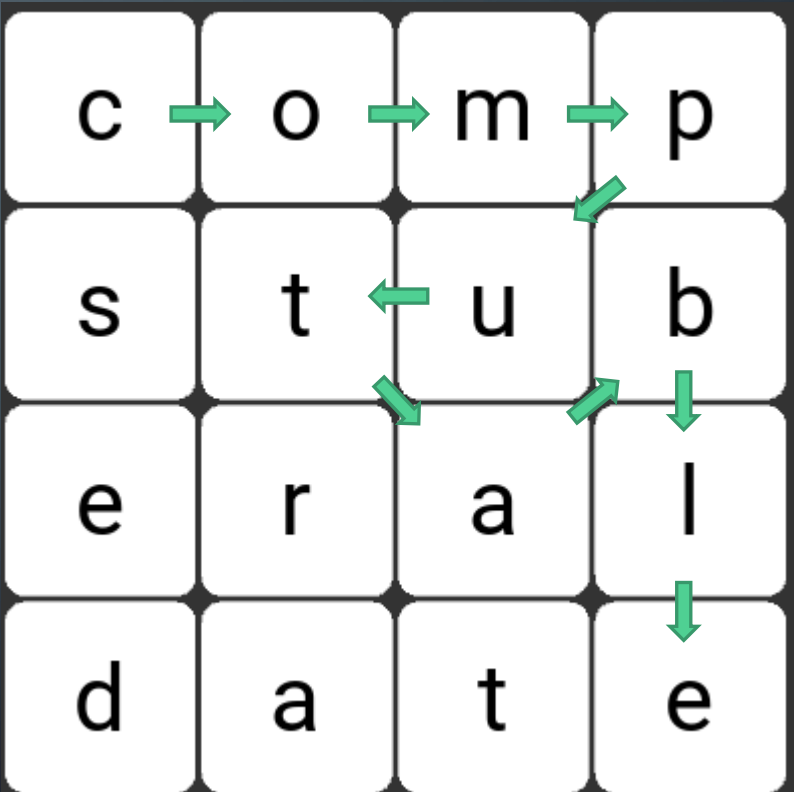


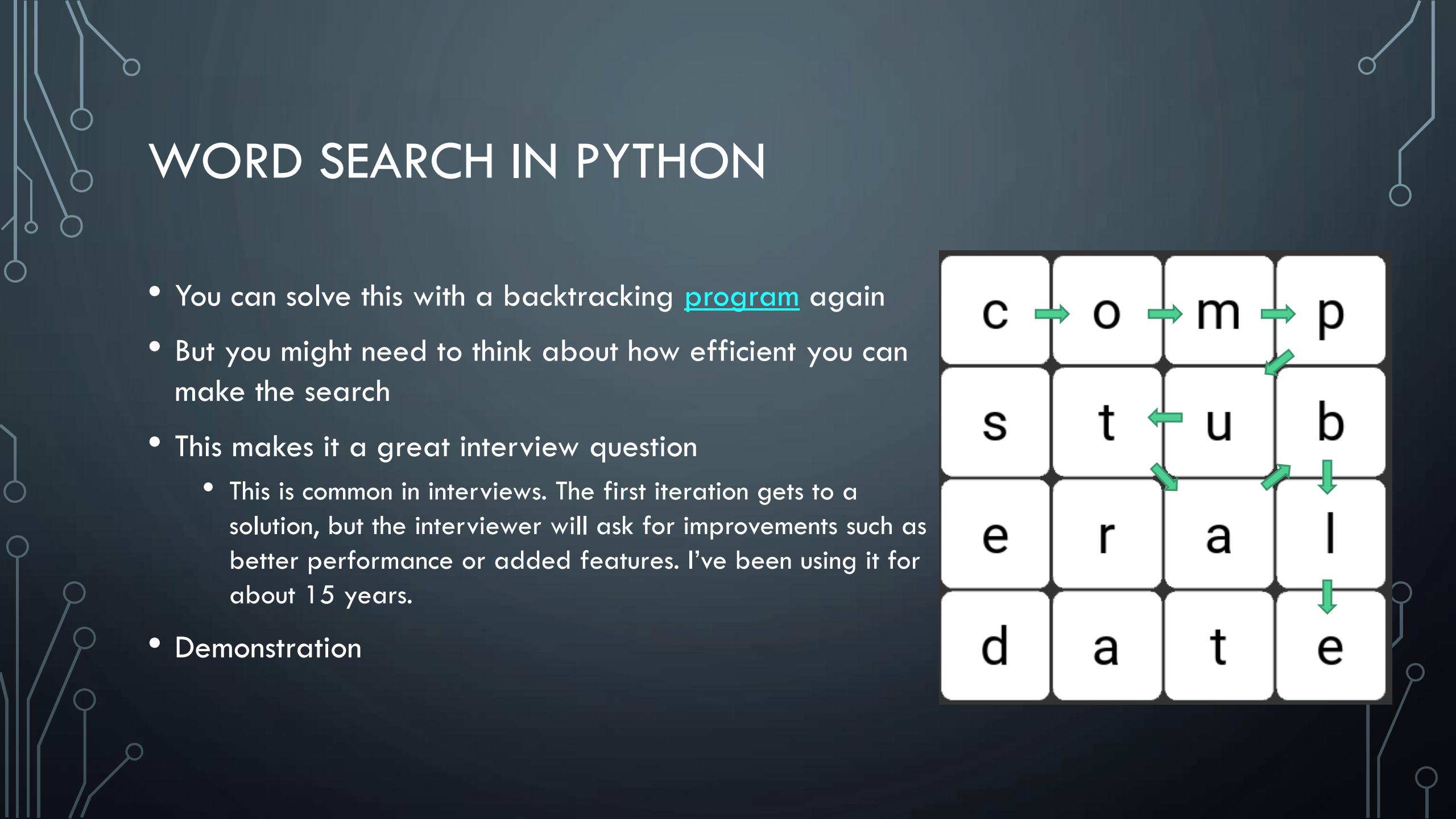
WORD SEARCH IN PYTHON

- You can solve this with a backtracking [program](#) again
- But you might need to think about how efficient you can make the search
- This makes it a great interview question
 - This is common in interviews. The first iteration gets to a solution, but the interviewer will ask for improvements such as better performance or added features. I've been using it for about 15 years.
- Demonstration



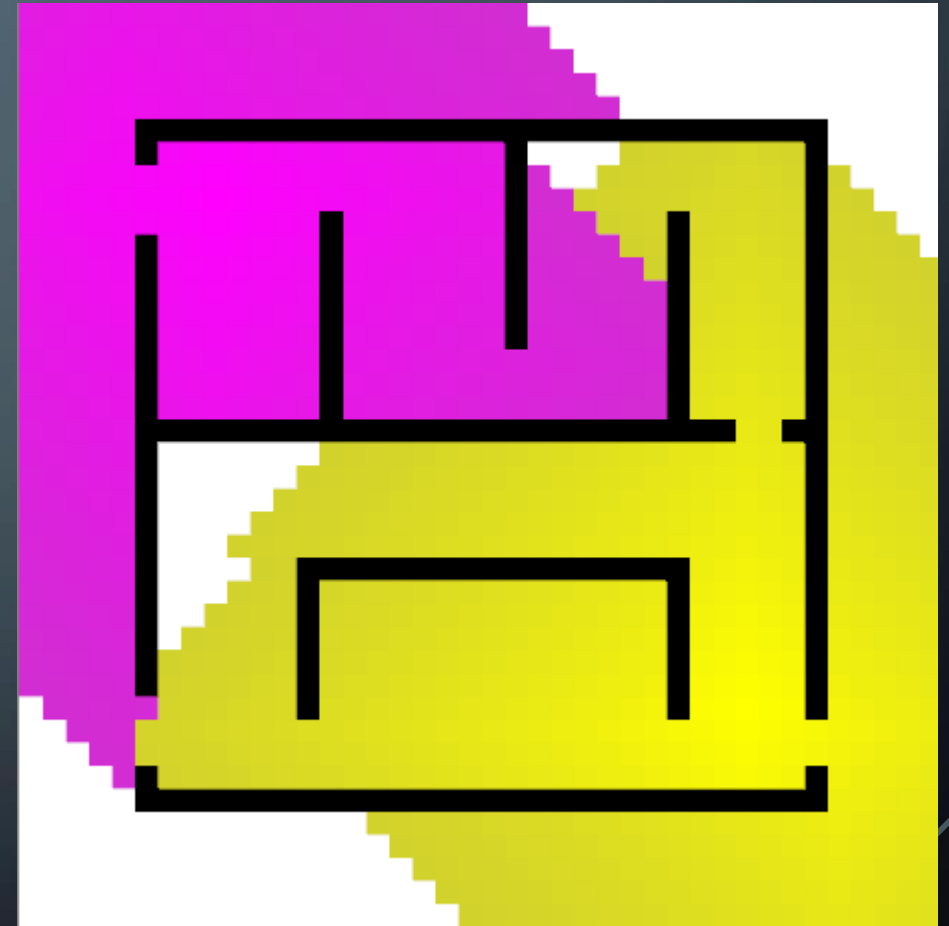
c	o	m	p
s	t	u	b
e	r	a	l
d	a	t	e

- # WORD SEARCH IN PYTHON
- You can solve this with a backtracking [program](#) again
 - But you might need to think about how efficient you can make the search
 - This makes it a great interview question
 - This is common in interviews. The first iteration gets to a solution, but the interviewer will ask for improvements such as better performance or added features. I've been using it for about 15 years.
 - Demonstration
- 
- | | | | |
|---|---|---|---|
| c | o | m | p |
| s | t | u | b |
| e | r | a | l |
| d | a | t | e |



FLOOD FILL IN PYTHON

- These backtracking solutions have been examples of depth-first searching
- [Flood fill](#) is an example of a breadth-first search
- Breadth-first search is important in social networks, maps, and other applications where finding shortest paths (or paths of a specific length) is important
- Demonstration

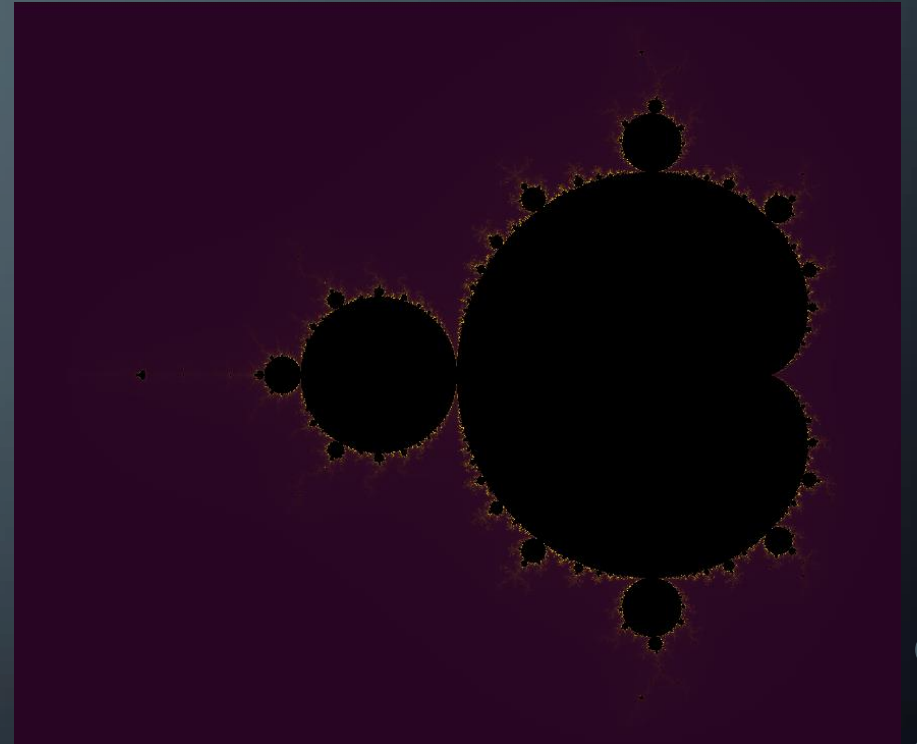


COMPUTER MUSIC IN C

- Programming should be fun
- As an elective in University, I took a computer music class
- We generated digital music samples with a program system written in FORTRAN
 - I generated the score for this composition in BASIC, of course
- The digital output was converted to analog and recorded on cassette tape
- I have a copy of the tape, but not the original score
- The Music program we were using has been converted to C
 - Lately I've been going on from where I was, recreating the score and instruments (using Python for generating the score)
 - It's a work in progress

MANDELBROT SETS IN C++

- Programming should be fun
- [Mandelbrot sets](#) are formed by looking at the repetition of the complex function, $f_c(z) = z^2 + c$
- Fractals were very popular in the 80s
- I wrote a Mandelbrot Set generator in Modula-2
 - But it was slow. Runs would take hours (for 640x350 16 color graphics), so I often left my work computer overnight. I also used my colleagues', so they would come in in the morning with striking graphics on their machines as a little gift
- Recently, I wanted to write a native Windows program in C++, and thought of this
 - This is another [work in progress](#), and the performance is not very good yet. But it is creating 2000x1440 2048 color images in seconds, because processing has gotten that much faster in 34 years.



The image features a dark blue gradient background. In the corners, there are decorative white line art elements resembling circuit boards or neural networks, with lines and small circles connecting them.

QUESTIONS?

EDUCATION QUESTIONS

- Where did you study and why did you choose that place?
- How did you know computer science/programming was something you wanted to do with your life and once you knew, how did you get to where you are now?
- What was college life like for Computer Science? How did you get into Computer Science and relate into your everyday life?
- How did you know what you wanted to do?
- What skills should we as high school students work on to help us later on when we get jobs? How do we start and develop personal projects while also learning new things?
- What methods do you use to learn a topic through self-studying?
- How does technology impact our lives today and what does it can AI do for us in the future?
- How do you advance yourself alongside technological advances? Is it possible to lose your job because you're not knowledgeable enough in a certain topic that wasn't expected when you started your job?

HISTORY QUESTIONS

- What programs does Russ mainly use for coding?
- What coding languages stand out to you and why? Are there any new languages that you have been wanting to learn and work with but haven't had the time to yet?
- What are some challenges you faced along the way?
- How is computer science different now than back then?
- How does it feel watching technology evolve to this day, how different is it?

JOB QUESTIONS

- How hard is it to get a job?
- How does the work environment and daily life differ between startups and larger companies?
- Any tip for students looking to start/join a startup?
- Did you do any internships, what types of coding languages do you know and what languages are being used in companies often so its worth knowing?
- What skills and talents helped you succeed and/or would you look for in a potential employee?
- What types questions do you ask in an interview?
- What is the hardest part of your job?

PROGRAM QUESTIONS

- How do you choose what pet projects to start? How do you keep yourself motivated with a pet project? Do you use them to fluff up your github (lol)
- What is your favorite Pet Project
- What is your favorite program?
- What are pet programs and why are they important?
- What are some of your pet projects, and how did you come up with the ideas for these pet projects?
- What is a pet project? Why are they important?
- The specific details on why we should have pet programs
- How did you start building your pet programs?