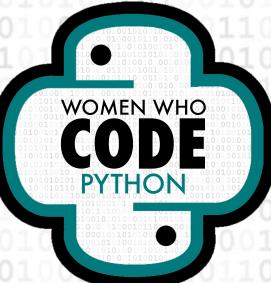


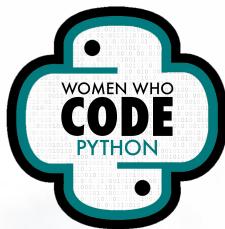
Welcome everyone!

- You can find these slides on GitHub here:
<https://github.com/WomenWhoCode/WWCodePython>
- Some housekeeping rules:
 - Everyone will be muted throughout the webinar, but there will be opportunities for participation!
 - Please share your thoughts on the chat and/or ask questions in the Q&A.
 - The entire team is here today. Please reach out to us with any technical questions!



WELCOME WOMEN WHO CODE





Women Who Code Python

**Intro to Data Structures
with Python:
Ace the Technical Interview**



Session #8: Graphs

WOMEN WHO
CODE[®]

MEET YOUR TEAM



Rishika Singh
Track Lead



Jasmeen Rajpal
Evangelist

WOMEN WHO
CODE

OUR MISSION

Inspiring women to
excel in technology
careers.

WOMEN WHO
CODE



OUR VISION

A world where women are representative as technical executives, founders, VCs, board members and software engineers.

WOMEN WHO
CODE



OUR TARGET

Engineers with two or more years of experience looking for support and resources to strengthen their influence and levelup in their careers.



CODE OF CONDUCT

WWCode is an inclusive community, dedicated to providing an empowering experience for everyone who participates in or supports our community, regardless of gender, gender identity and expression, sexual orientation, ability, physical appearance, body size, race, ethnicity, age, religion, socioeconomic status, caste, creed, political affiliation, or preferred programming language(s).

Our events are intended to inspire women to excel in technology careers, and anyone who is there for this purpose is welcome. We do not tolerate harassment of members in any form. Our **Code of Conduct** applies to all WWCode events and online communities.

Read the full version and access our incident report form at womenwhocode.com/codeofconduct

230,000 Members

70 networks in 20 countries
Members in 97+ countries
10K+ events
\$1025 daily Conference tickets
\$2M Scholarships
Access to [jobs](#) + [resources](#)
Infinite connections



OUR MOVEMENT

As the world changes, we can be a connecting force that creates a sense of belonging while the world is being asked to isolate.



Upcoming Events

FRI
25
JUN

✨ Intro to Data Structures with Python: Ace the Technical Interview (Session #8:
Graphs) ✨ *Featured*

📍 Online | Python | 9:00 AM - 10:30 AM KST (UTC+0900)

[Register](#)

WED
30
JUN

✨ Ask Me Anything with Jane Street Software Engineers ✨ *Featured*

📍 Online | Python | 12:00 AM - 1:00 AM KST (UTC+0900)

[Register](#)

FRI
09
JUL

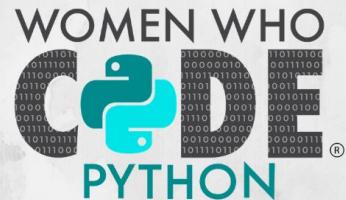
✨ Intro to Data Structures with Python: Ace the Technical Interview (Session #9: Maps &
Hash Tables) ✨ *Featured*

📍 Zoom | Python | 9:00 AM - 10:30 AM KST (UTC+0900)

[Register](#)



Stay Connected



WOMEN WHO
CODE
PYTHON®

JOIN US ON SOCIAL MEDIA!

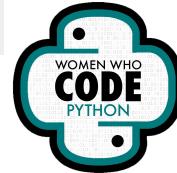
@WWCODEPYTHON

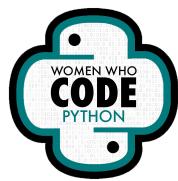
WOMENWHOCODE.COM/PYTHON



Nayeon Shin

Sophomore CS & New Media Art Major | WWCode Python Track General Volunteer



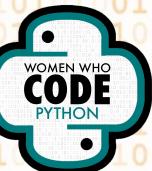


Today's Agenda



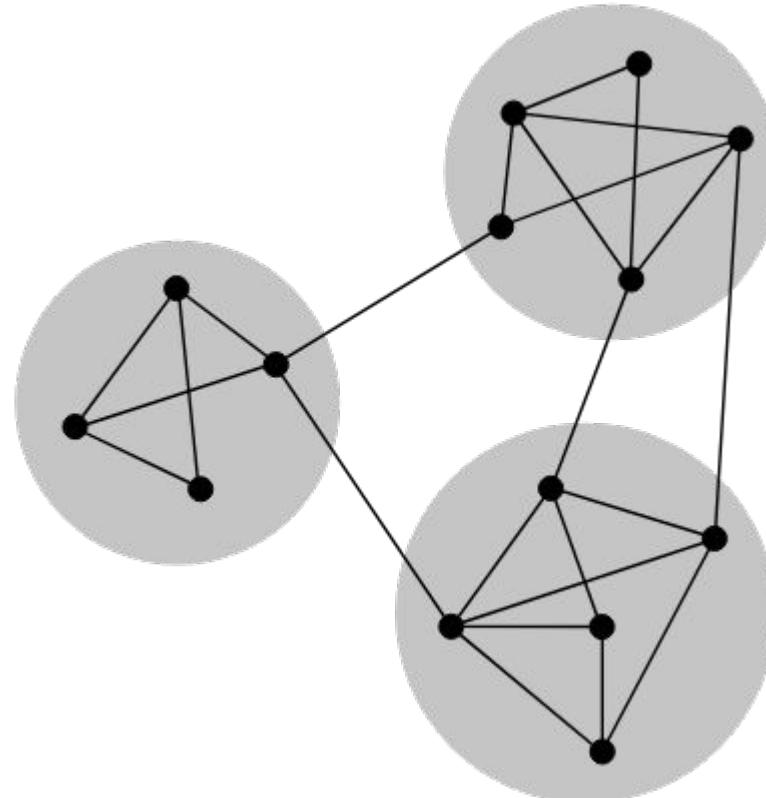
1. Definition
2. Applications
3. Concepts
4. Implementation
 - a. Adjacency List
 - b. Adjacency Matrix
5. Traversals
 - a. Breadth-First Search
 - b. Depth-First Search
6. Q&A
7. Live Coding

Definition

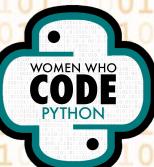


Definition - Graphs

- Not “charts”
- Relationships between data items
- Consists of nodes & edges



Applications

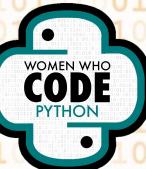


Applications - Graphs

- Many!
 - Networking
 - How the Internet is connected
 - Transportation
 - System of roads
 - Airline flights from city to city
 - Game playing
 - AI apps

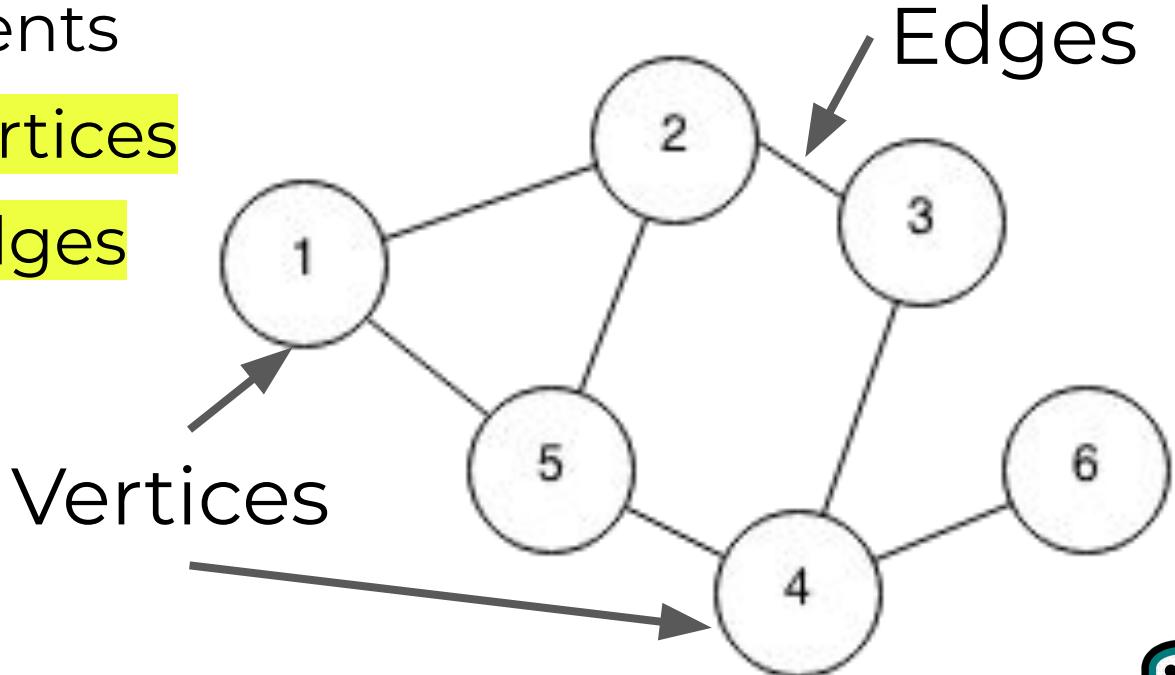


Concepts



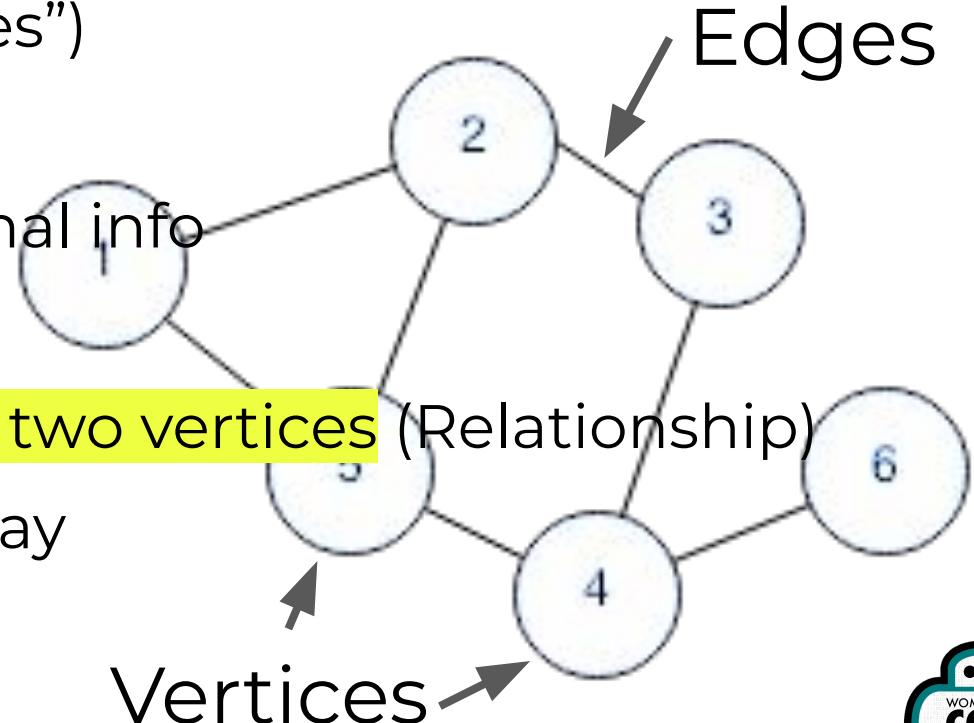
Components - Graphs

- Two components
 - V : Set of vertices
 - E : Set of Edges



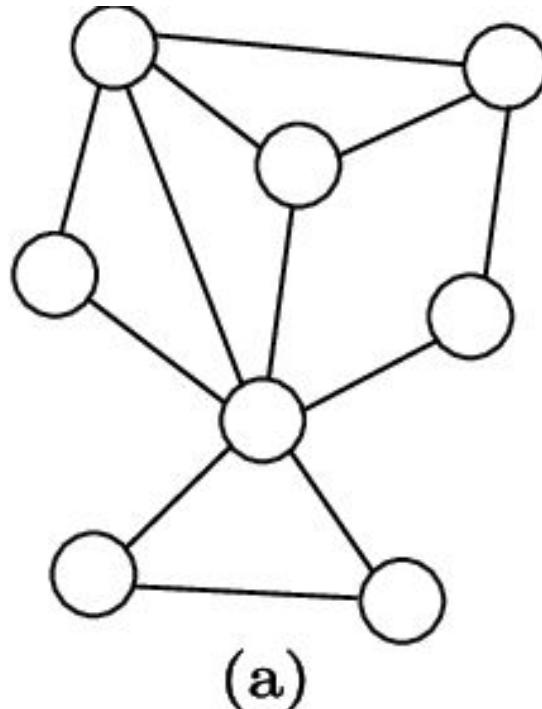
Components - Graphs

- **Vertex** (plural: “Vertices”)
 - “Key”: Name
 - “Payload”: Additional info
- **Edge** (a.k.a.: “Arc”)
 - An edge **connects two vertices** (Relationship)
 - One-way or two-way

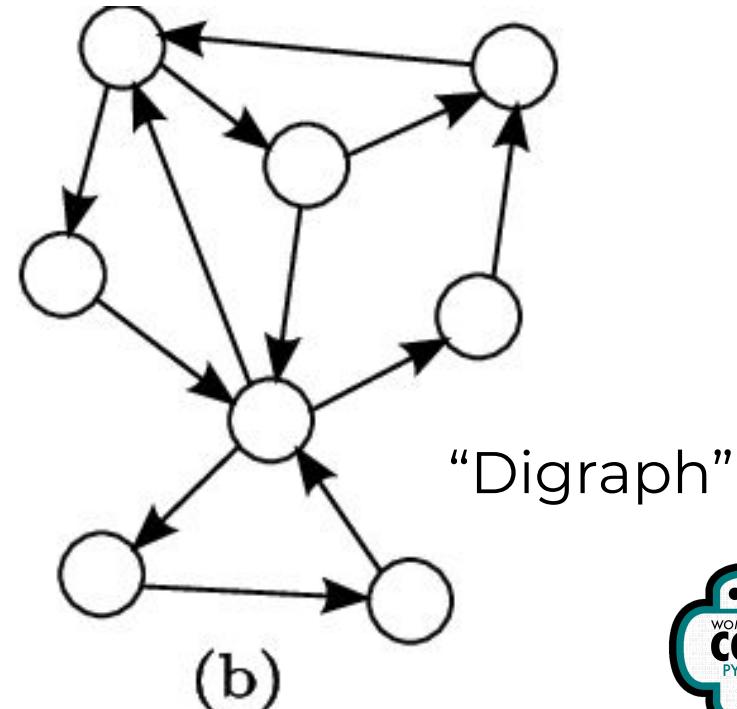


Undirected vs Directed Graphs

Undirected Graph (Two-way)



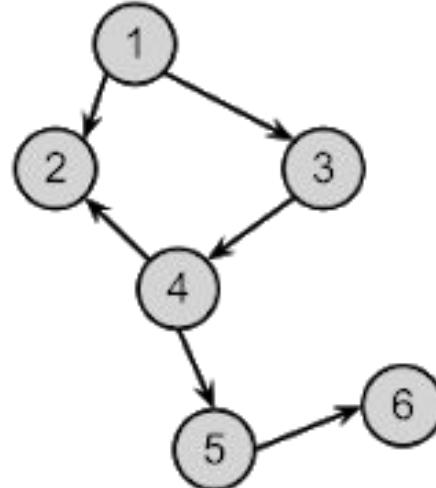
Directed Graph (One-way)



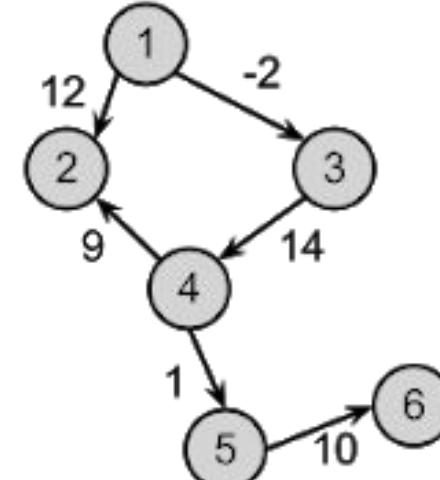
Unweighted vs Weighted Graphs

- Weight: Cost to go from one vertex to another
 - Example: Distance between two cities in graph of roads

Unweighted Graph

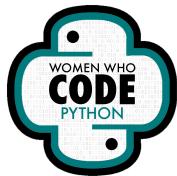


Weighted Graph



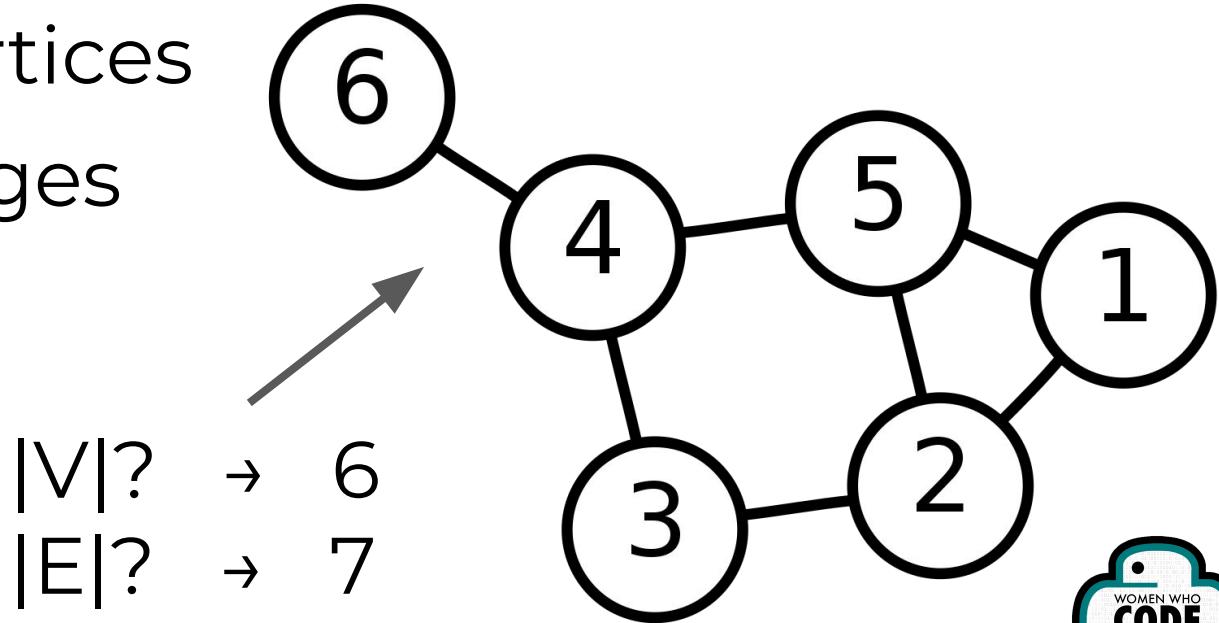
Additional Concepts - Graphs

- Cardinality
- Degree
- Adjacent Vertices
- Path
- Cycle



Cardinality - Graphs

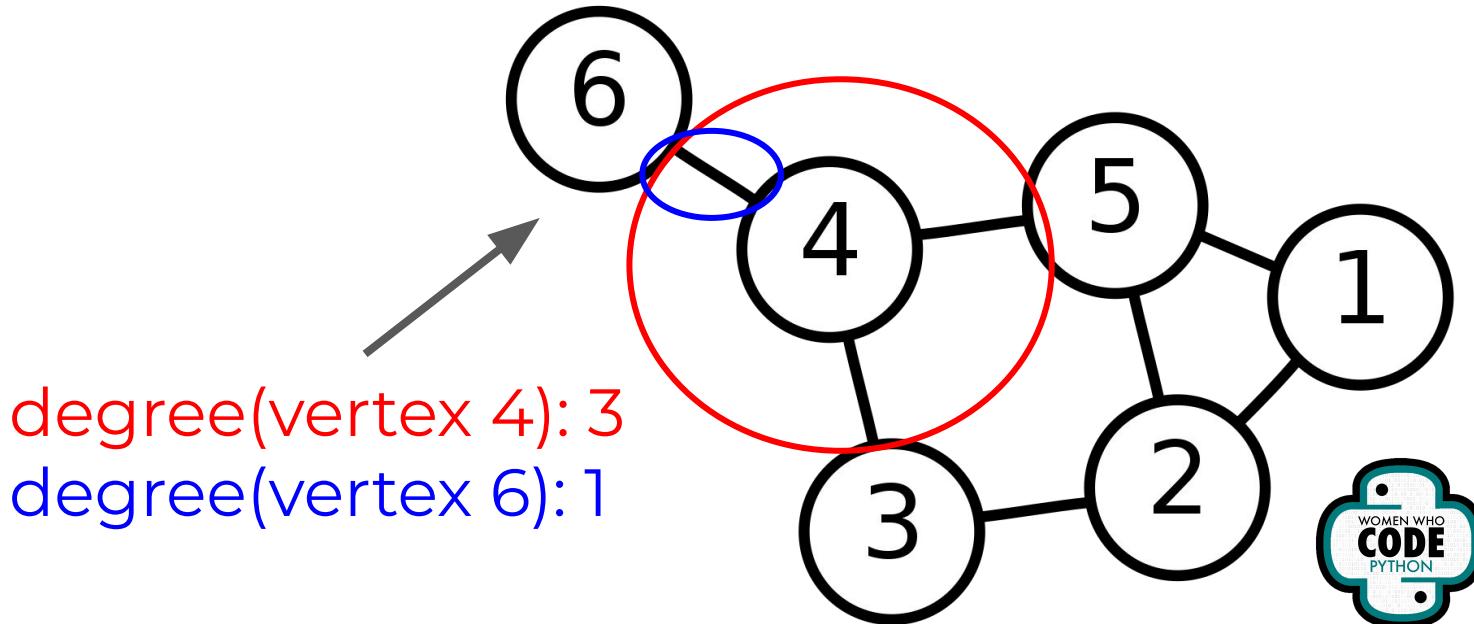
- Number of Vertices/Edges
- $|V| \rightarrow$ # of Vertices
- $|E| \rightarrow$ # of Edges



Degree - Graphs

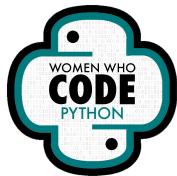
- Degree of a vertex:

of edges connected to a vertex



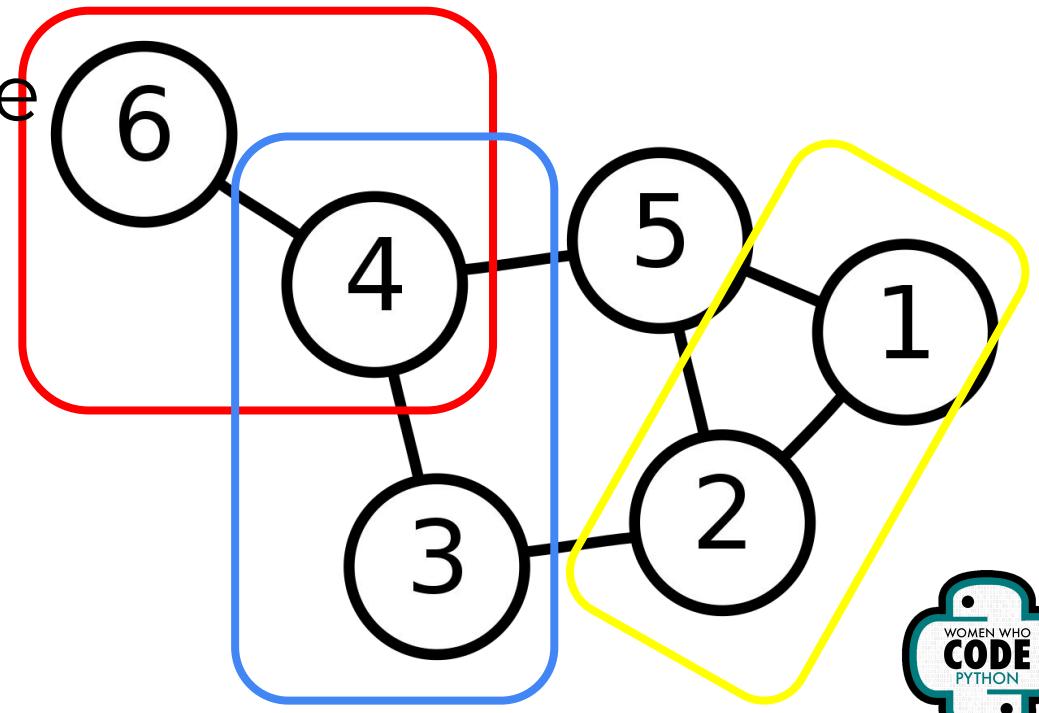
Additional Concepts - Graphs

- Cardinality
- Degree
- Adjacent Vertices
- Path
- Cycle



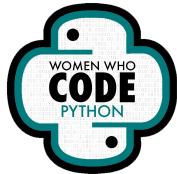
Adjacent Vertices - Graphs

- A pair of vertices that have an edge between them



Additional Concepts - Graphs

- Cardinality
- Degree
- Adjacent Vertices
- Path
- Cycle

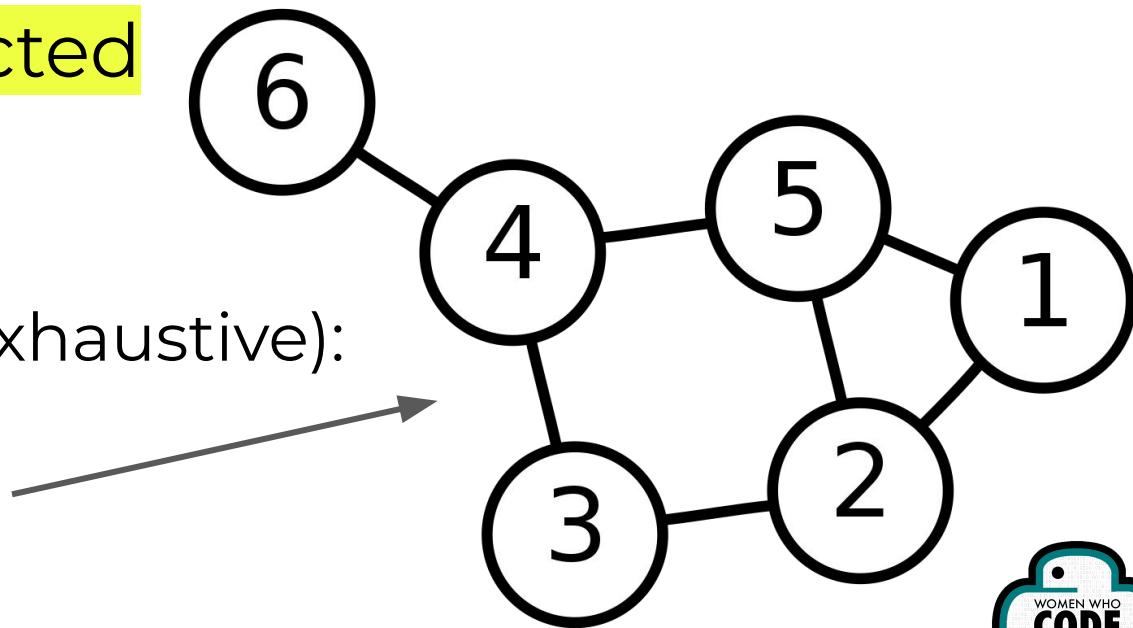


Path - Graphs

- Sequence of vertices
that are connected
by edges

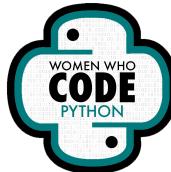
Paths (not exhaustive):

- 6, 4, 5, 2, 1
- 3, 4, 6
- 3, 2, 5, 4



Additional Concepts - Graphs

- Cardinality
- Degree
- Adjacent Vertices
- Path
- Cycle

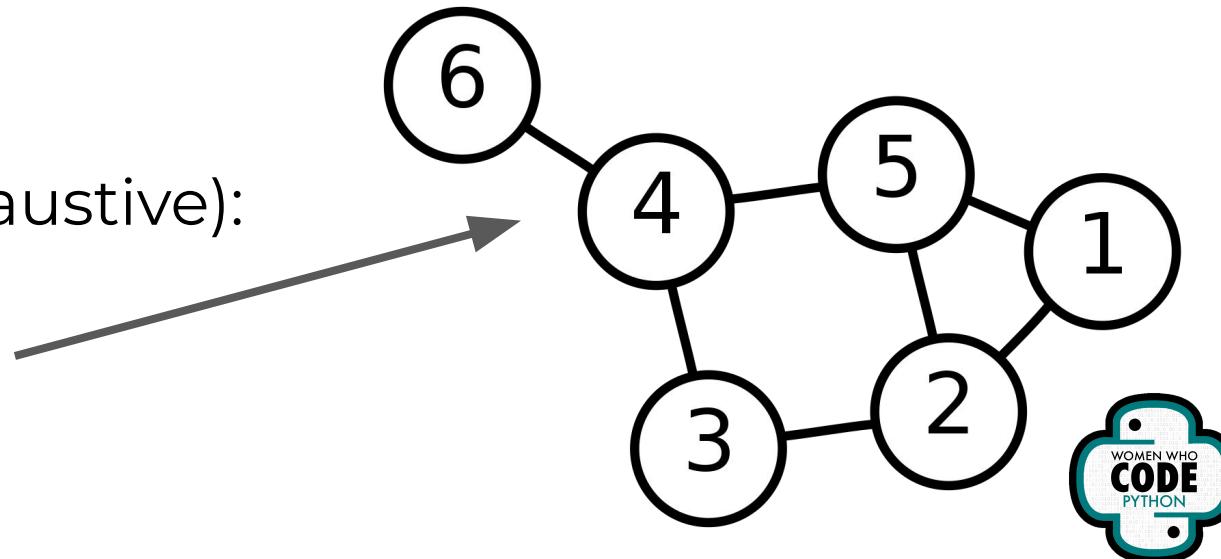


Cycle - Graphs

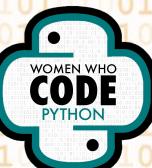
- Path that starts and ends at the same vertex
 - Acyclic graph: No cycle
 - & Directed: DAG (Directed Acyclic Graph)

Cycle (not exhaustive):

- 4, 3, 2, 5, 4
- 1, 5, 2, 1
- 3, 2, 5, 4, 3

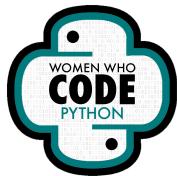


Implementation



Implementation - Graphs

- Adjacency List
- Adjacency Matrix



Adjacency List - Graphs

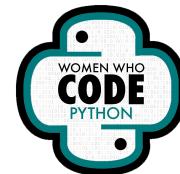
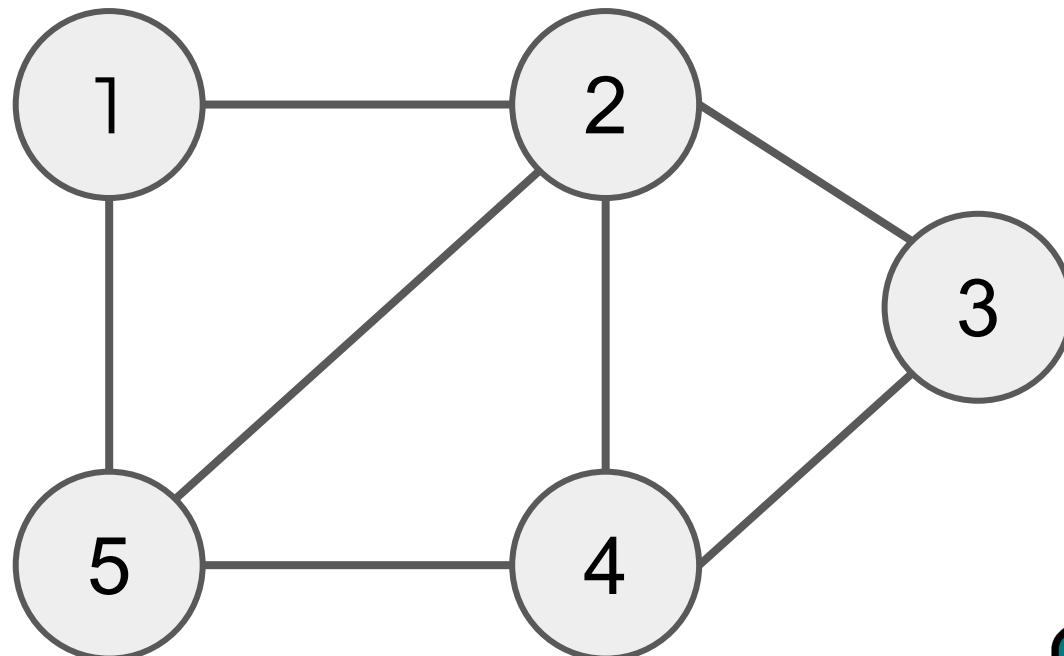
1 → 2 → 5

2 → 1 → 3 → 4 → 5

3 → 2 → 4

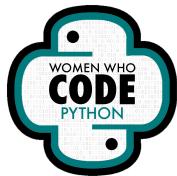
4 → 2 → 3 → 5

5 → 1 → 2 → 4



Implementation - Graphs

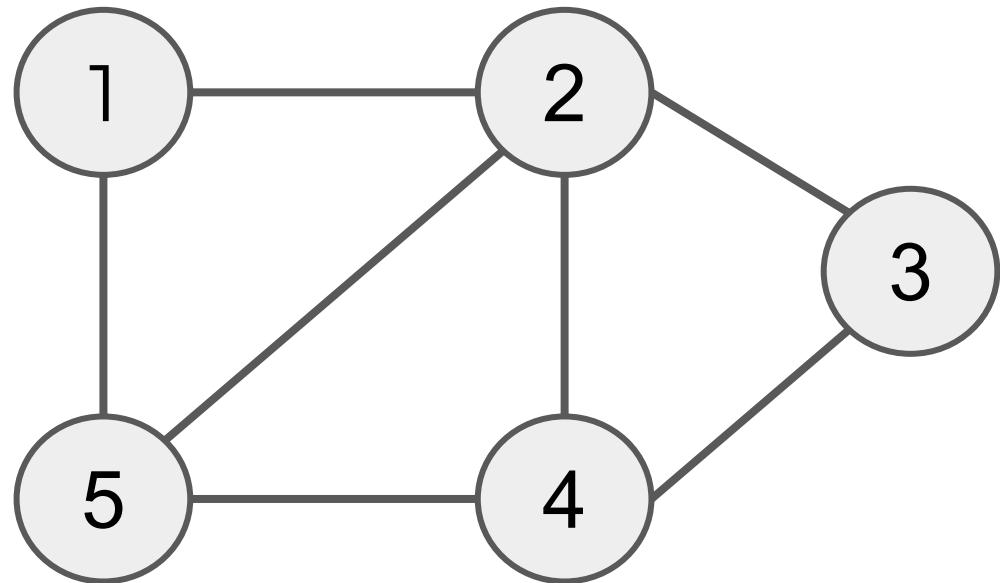
- Adjacency List
- Adjacency Matrix



Adjacency Matrix - Graphs

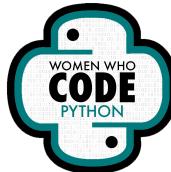
- 2D matrix

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0



Implementation - Graphs

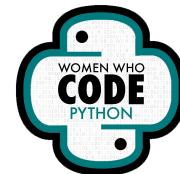
- Which implementation to use?
 - # of vertices doesn't matter
 - # of edges do!
 - Sparse graph $\rightarrow |E| < |V|^2$
 - Use adjacency list
 - Dense graph $\rightarrow |E| \text{ close to } |V|^2$
 - Doesn't matter



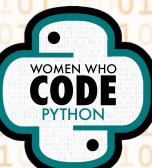
Implementation - Graphs

- Sparse graph
 - Most cells empty
- Adjacency Matrix
 - Learn quickly whether an edge exists between two vertices

	v0	v1	v2	v3	v4	v5
v0		5				2
v1			4			
v2				9		
v3					7	3
v4	1					
v5			1		8	

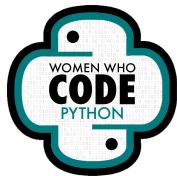


Traversals



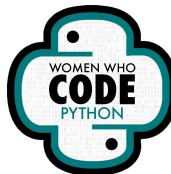
Traversals - Graphs

- Breadth-First Search
- Depth-First Search

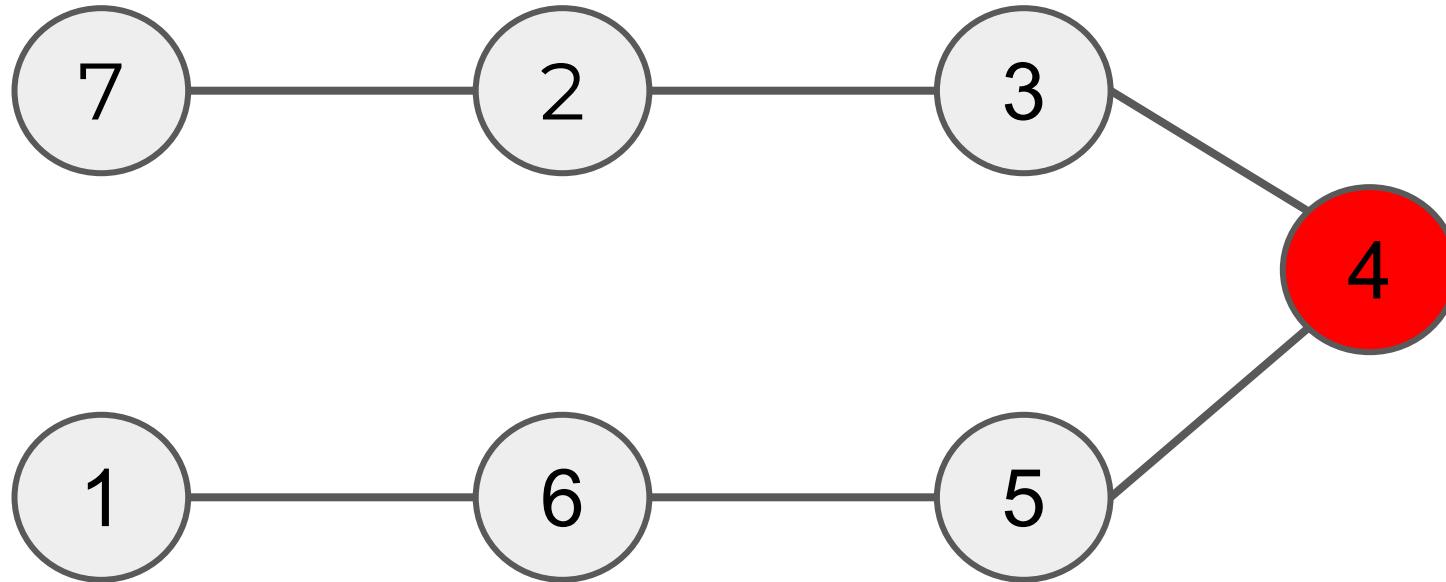


Breadth-First Search - Graphs

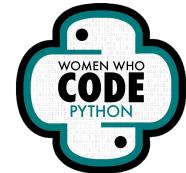
1. Visit all vertices at first level
2. Equivalently:
 - a. Visit all vertices that are one step away
 - b. then vertices that are 2 steps away
 - c. then vertices that are 3 steps away.....
 - d. Until all vertices have been visited



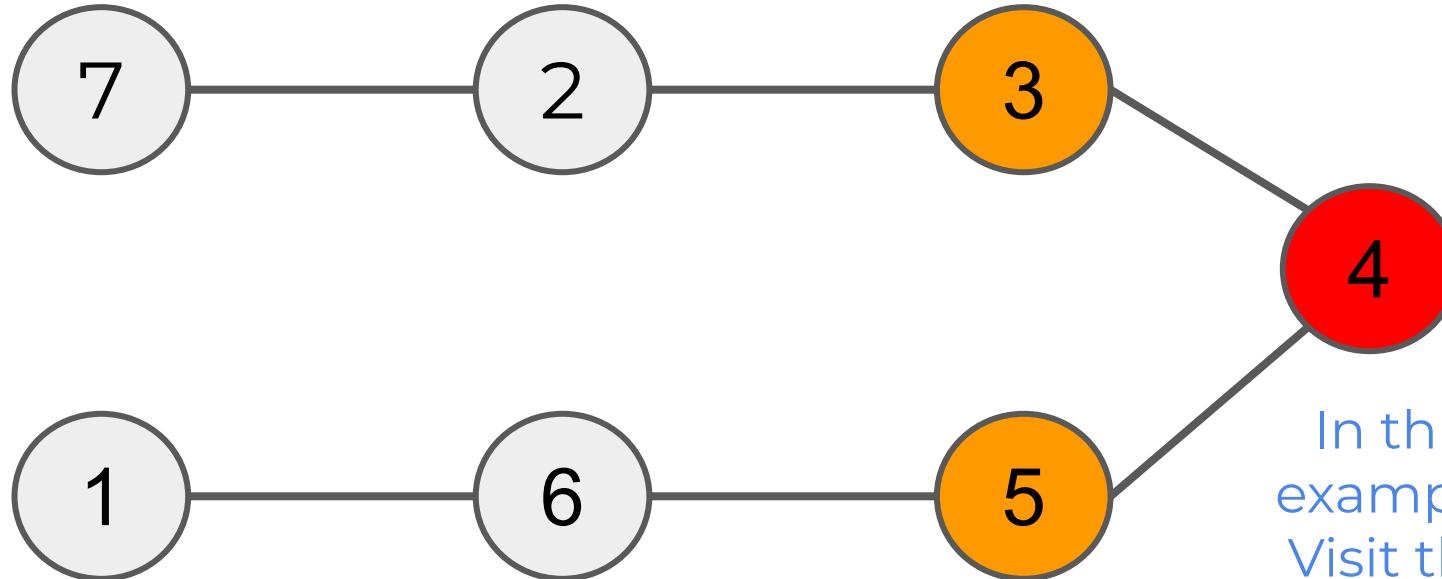
Breadth-First Search - Graphs



Visit Order: 4



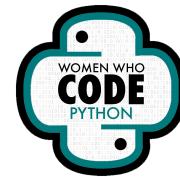
Breadth-First Search - Graphs



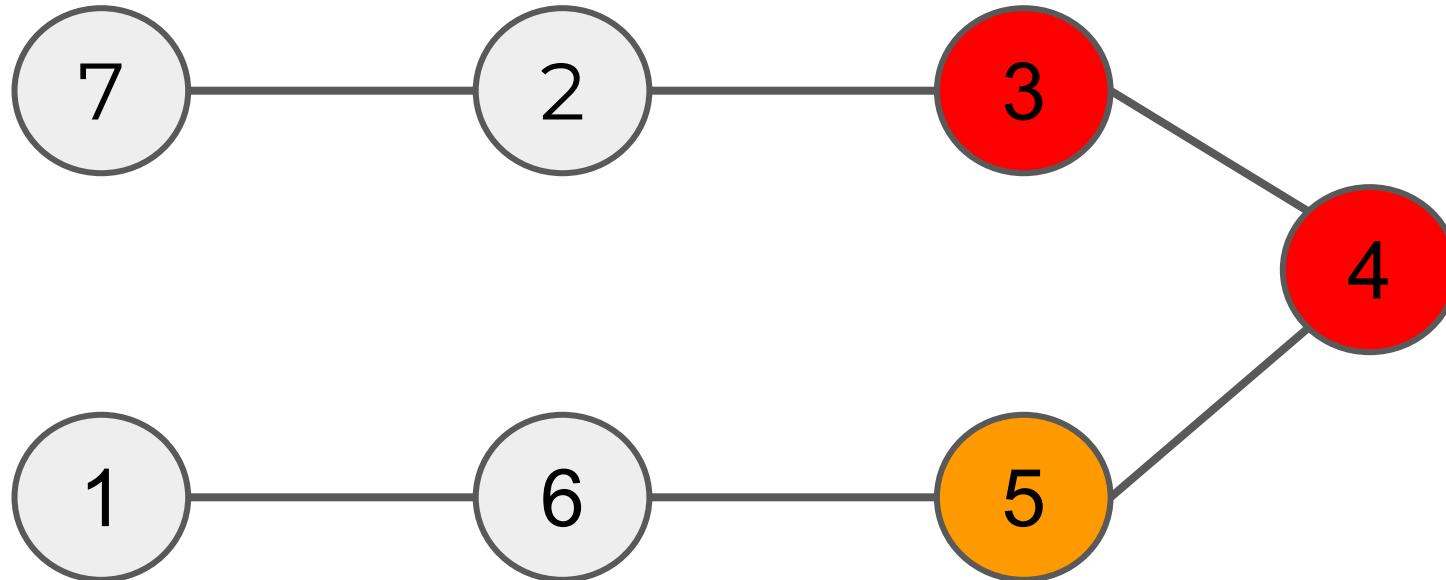
Visit Order: 4

Next level: 3, 5

In this example:
Visit the smaller ones first!

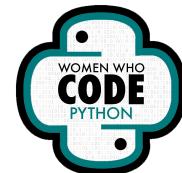


Breadth-First Search - Graphs

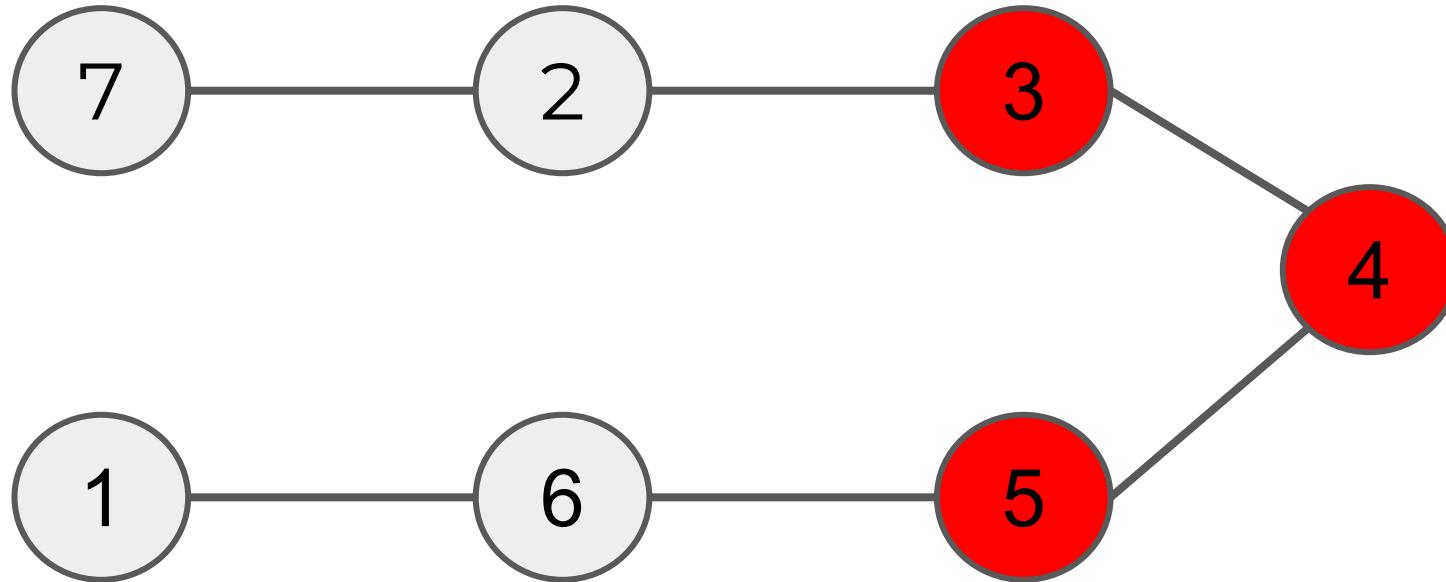


Visit Order: 4, 3

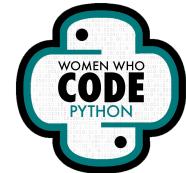
In holding: 5



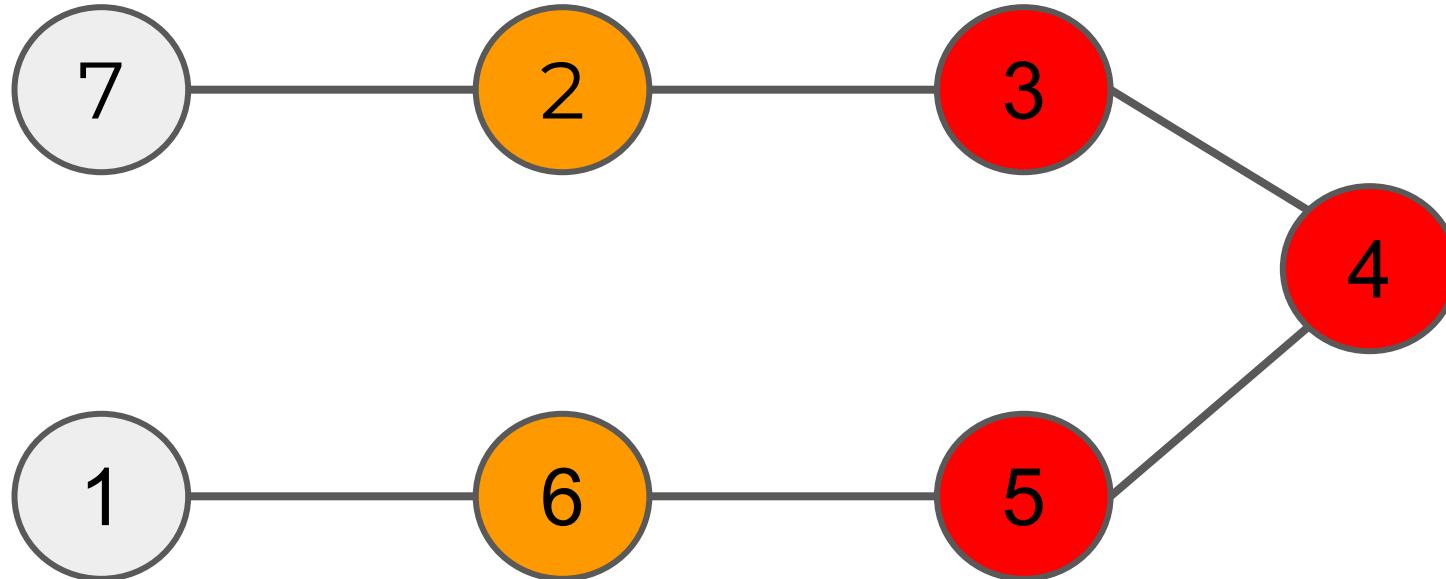
Breadth-First Search - Graphs



Visit Order: 4, 3, 5

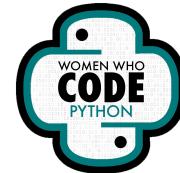


Breadth-First Search - Graphs

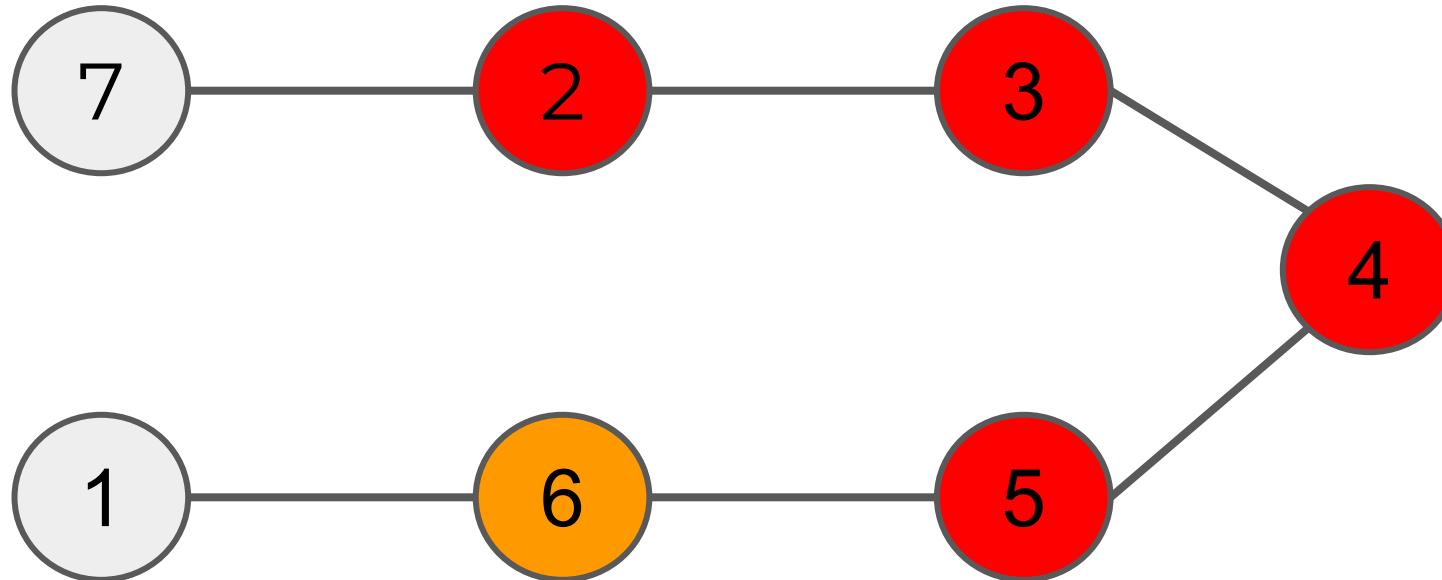


Visit Order: 4, 3, 5

Next level: 2, 6

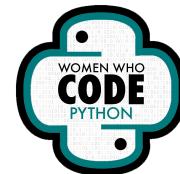


Breadth-First Search - Graphs

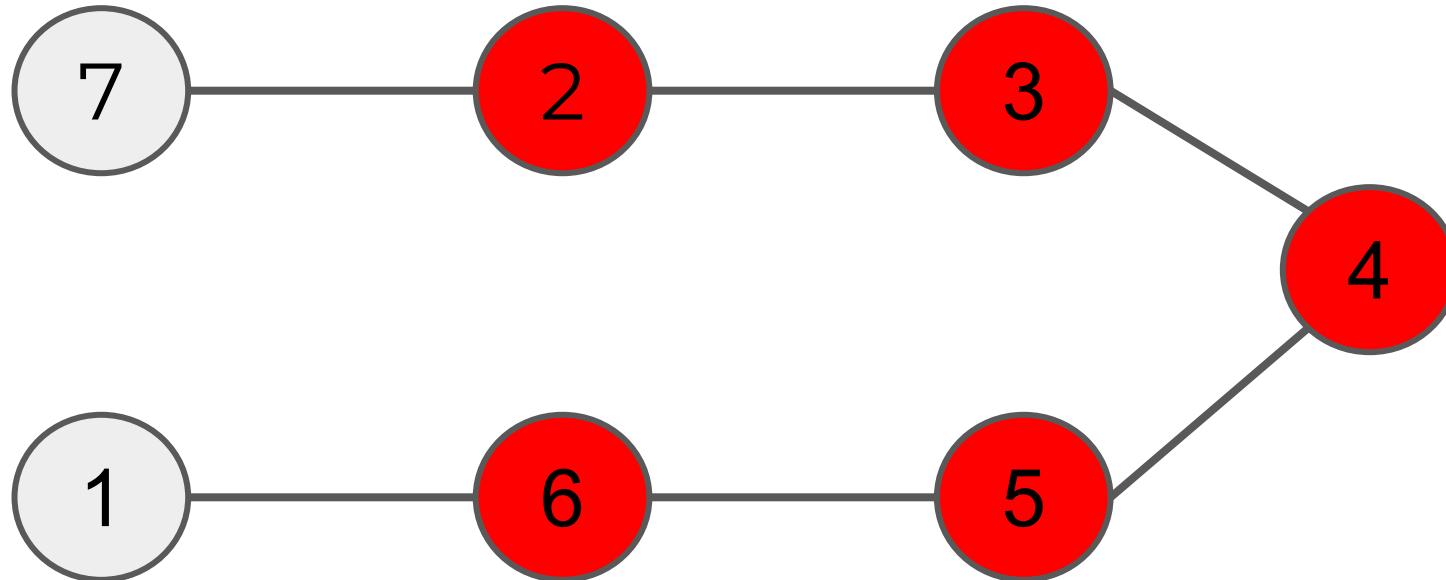


Visit Order: 4, 3, 5, 2

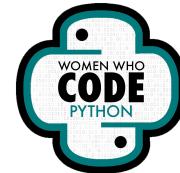
In holding: 6



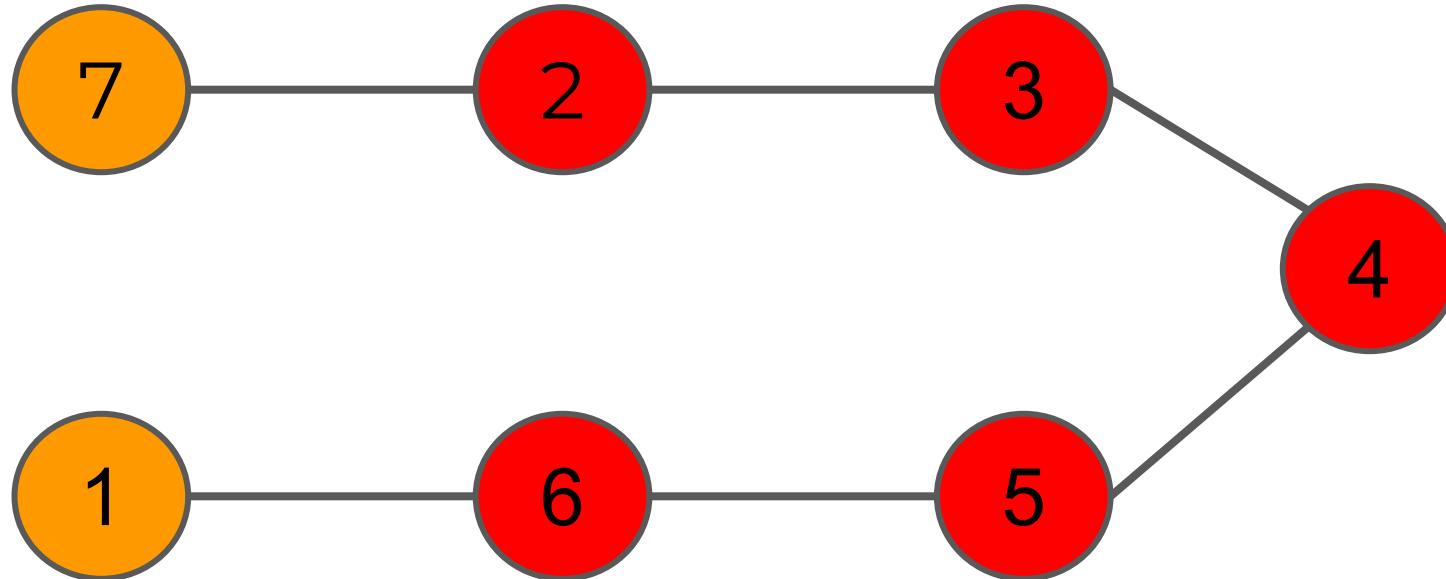
Breadth-First Search - Graphs



Visit Order: 4, 3, 5, 2, 6

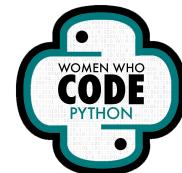


Breadth-First Search - Graphs

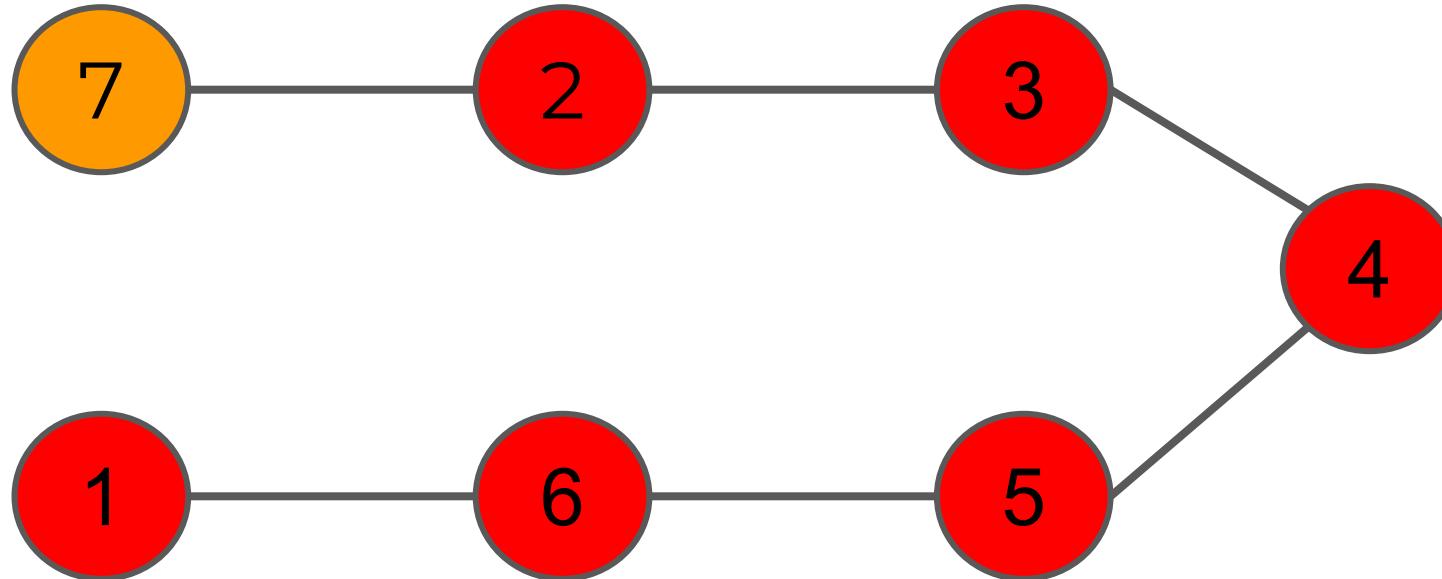


Visit Order: 4, 3, 5, 2, 6

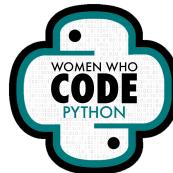
Next level: 7, 1



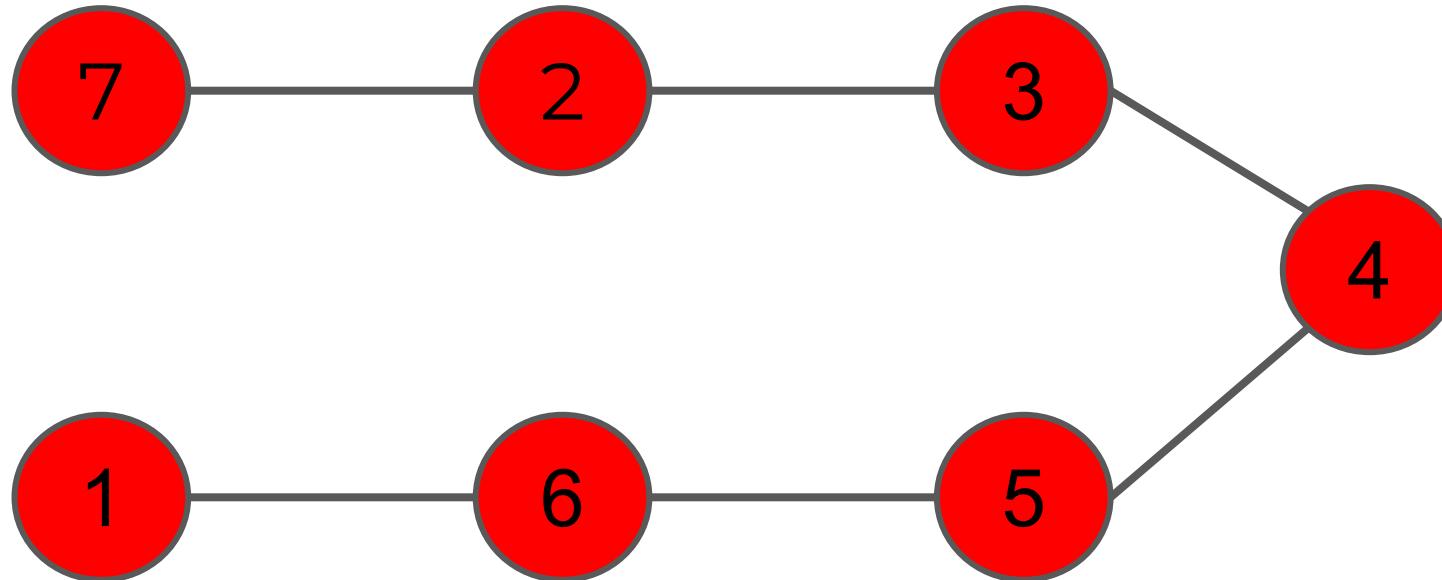
Breadth-First Search - Graphs



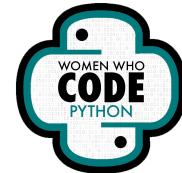
Visit Order: 4, 3, 5, 2, 6, 1 In holding: 7



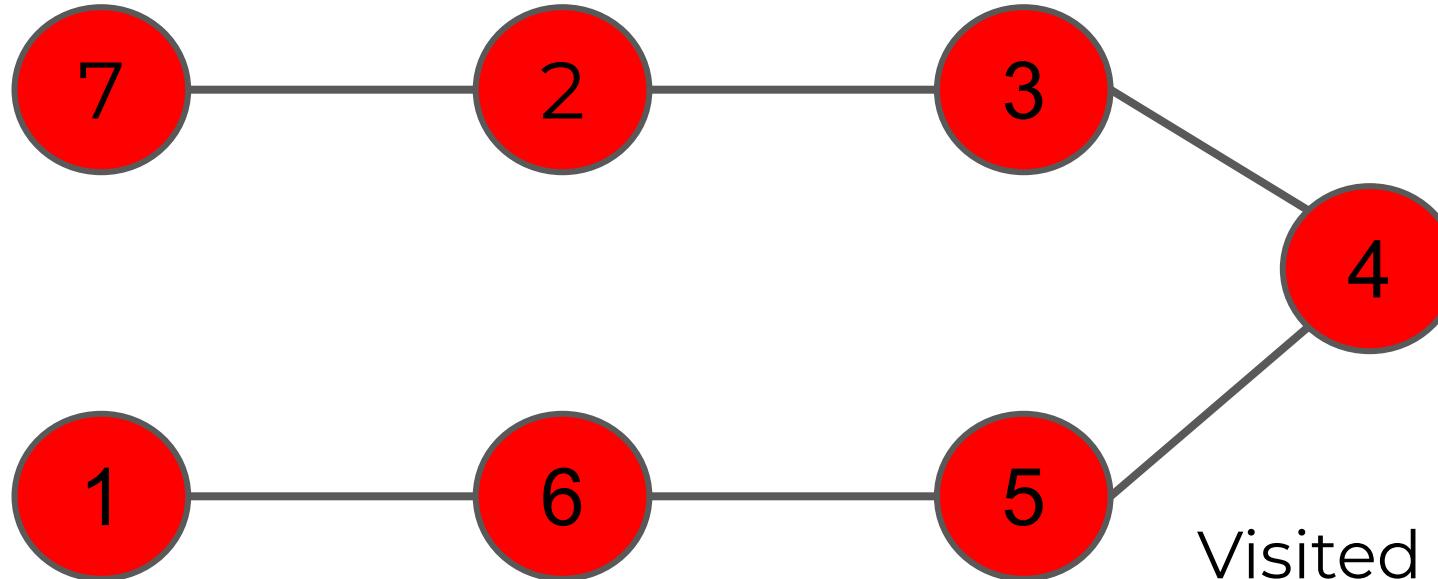
Breadth-First Search - Graphs



Visit Order: 4, 3, 5, 2, 6, 1, 7

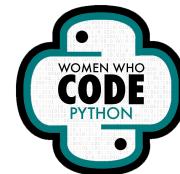


Breadth-First Search - Graphs



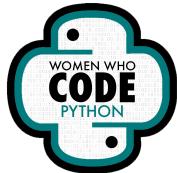
Visit Order: 4, 3, 5, 2, 6, 1, 7

Visited all
vertices
→ Done

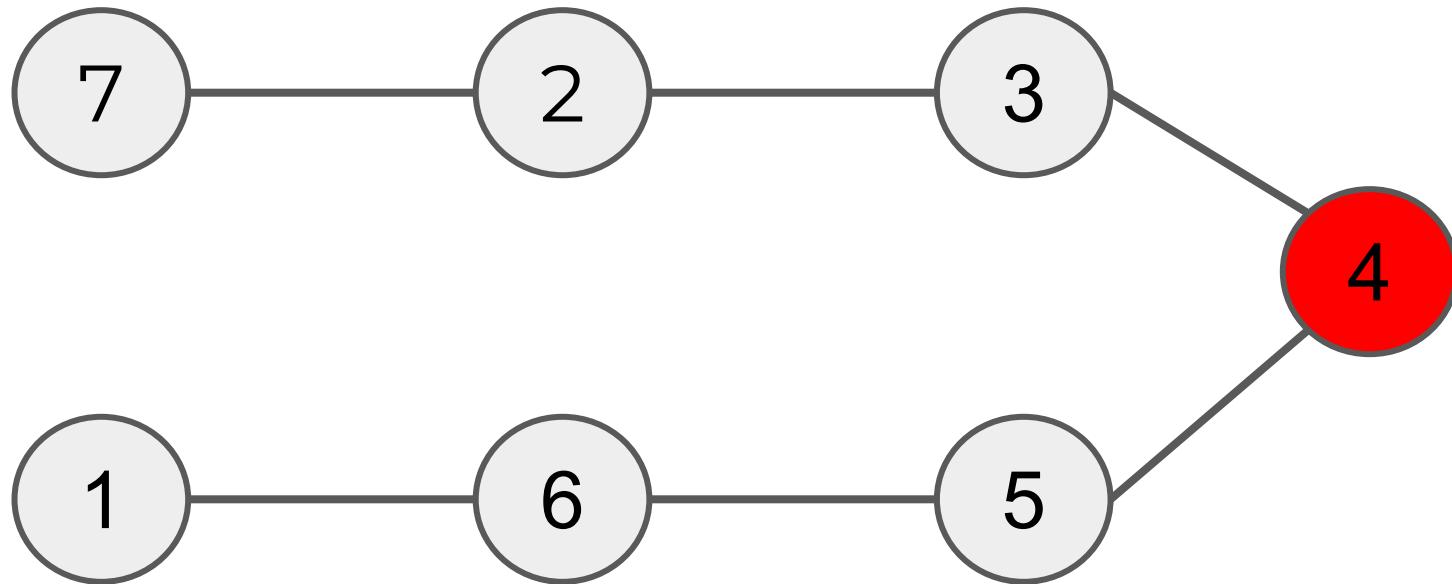


Depth-First Search - Graphs

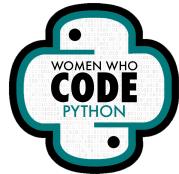
1. Go as deep as possible
 - Until there's no adjacent vertex
2. Backtrack and check for other vertices
3. Repeat until visiting all vertices



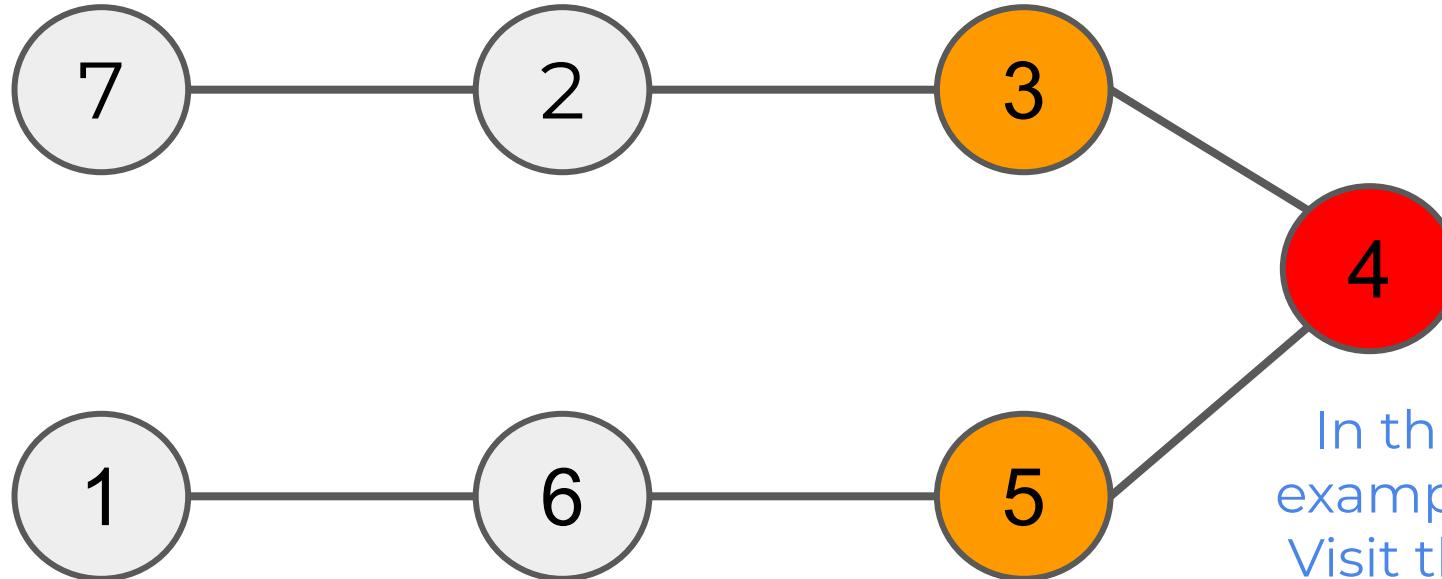
Depth-First Search - Graphs



Visit Order: 4



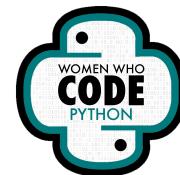
Depth-First Search - Graphs



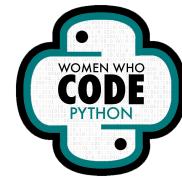
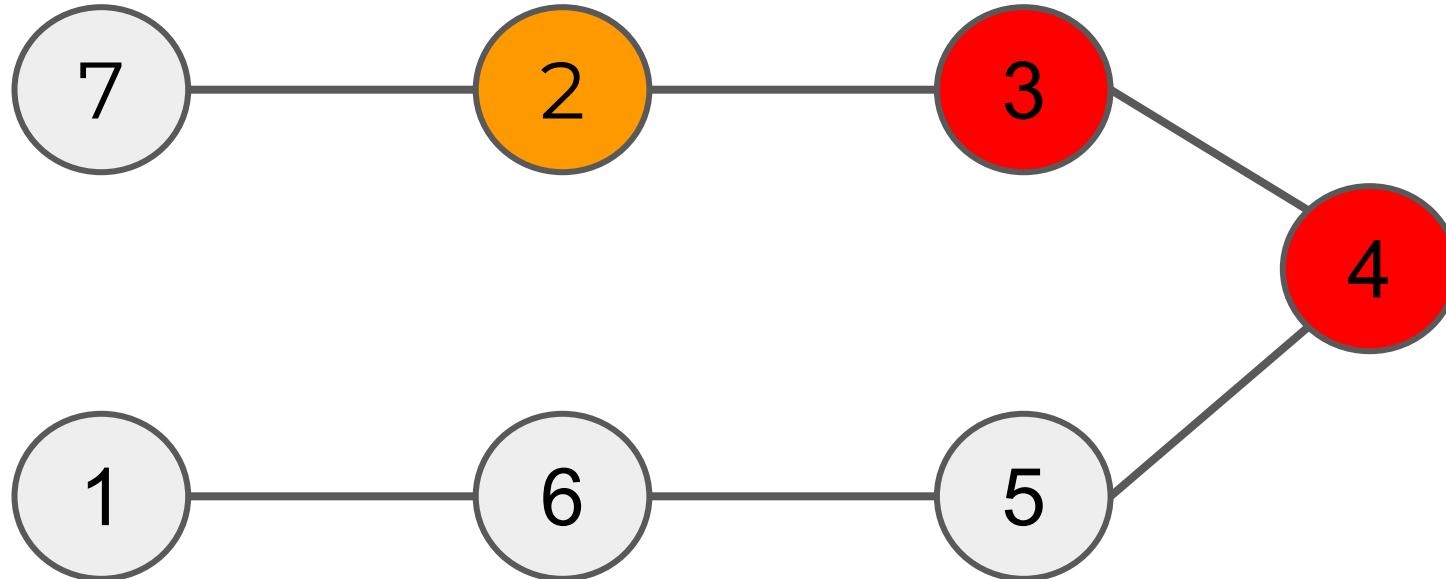
Visit Order: 4

Next level: 3, 5

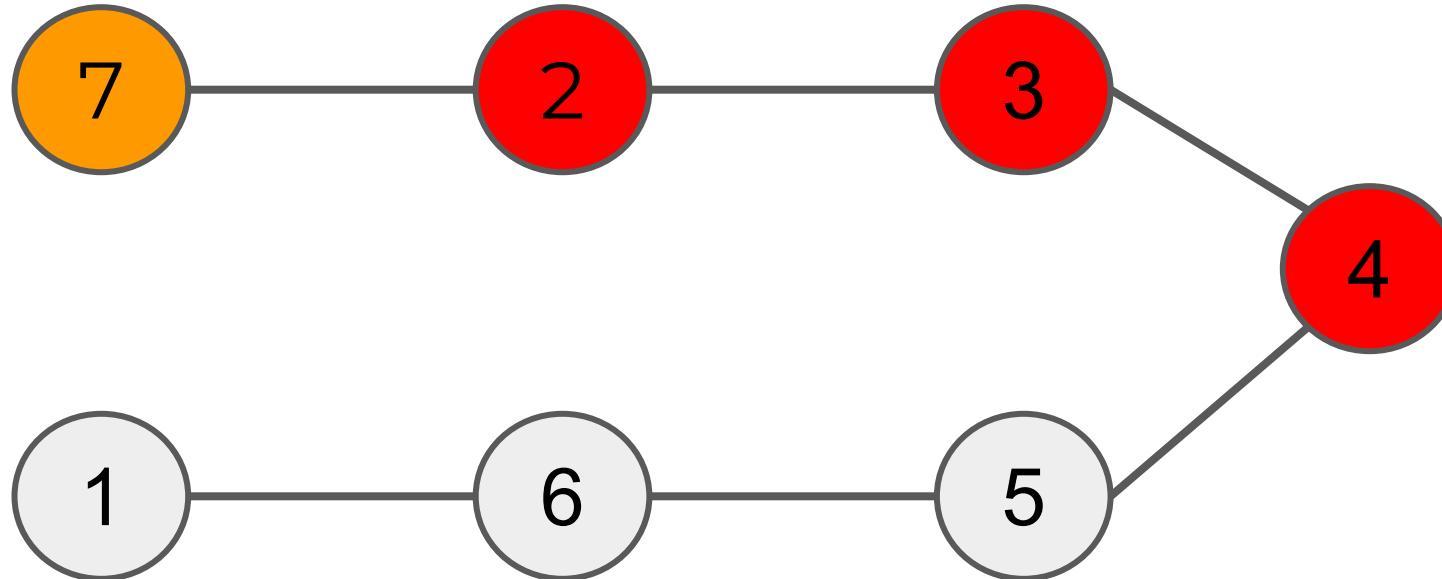
In this
example:
Visit the
smaller
ones first!



Depth-First Search - Graphs

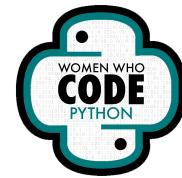


Depth-First Search - Graphs

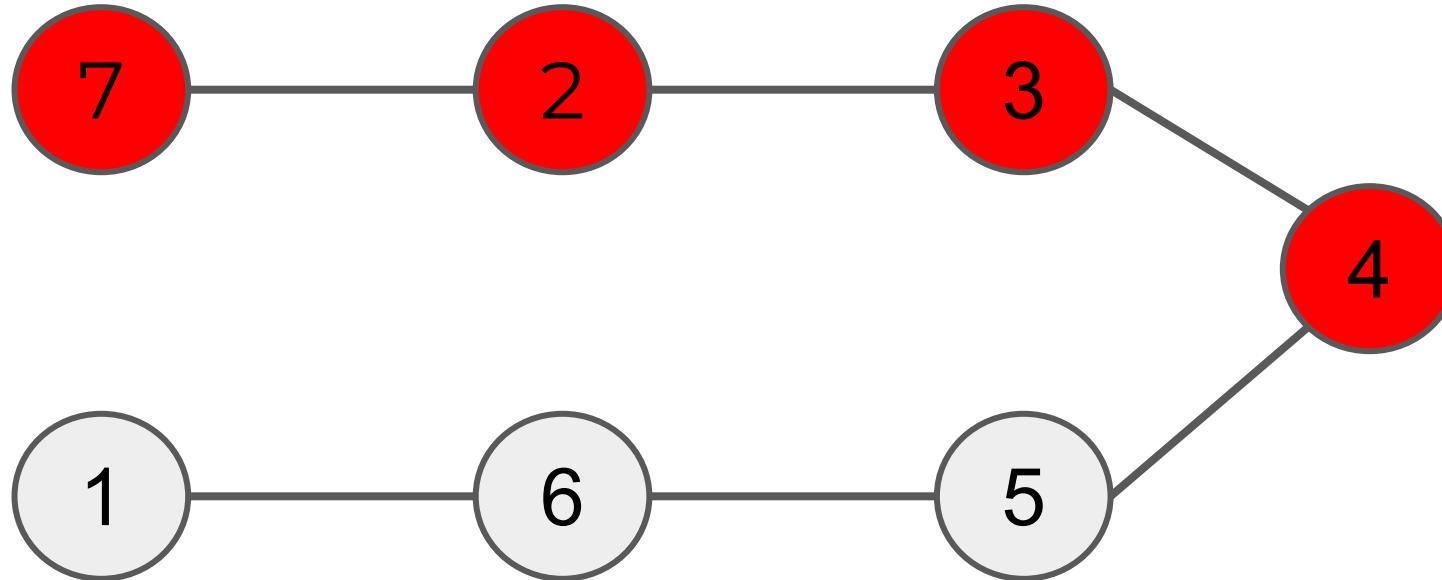


Visit Order: 4, 3, 2

Next level: 7

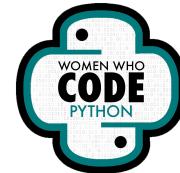


Depth-First Search - Graphs

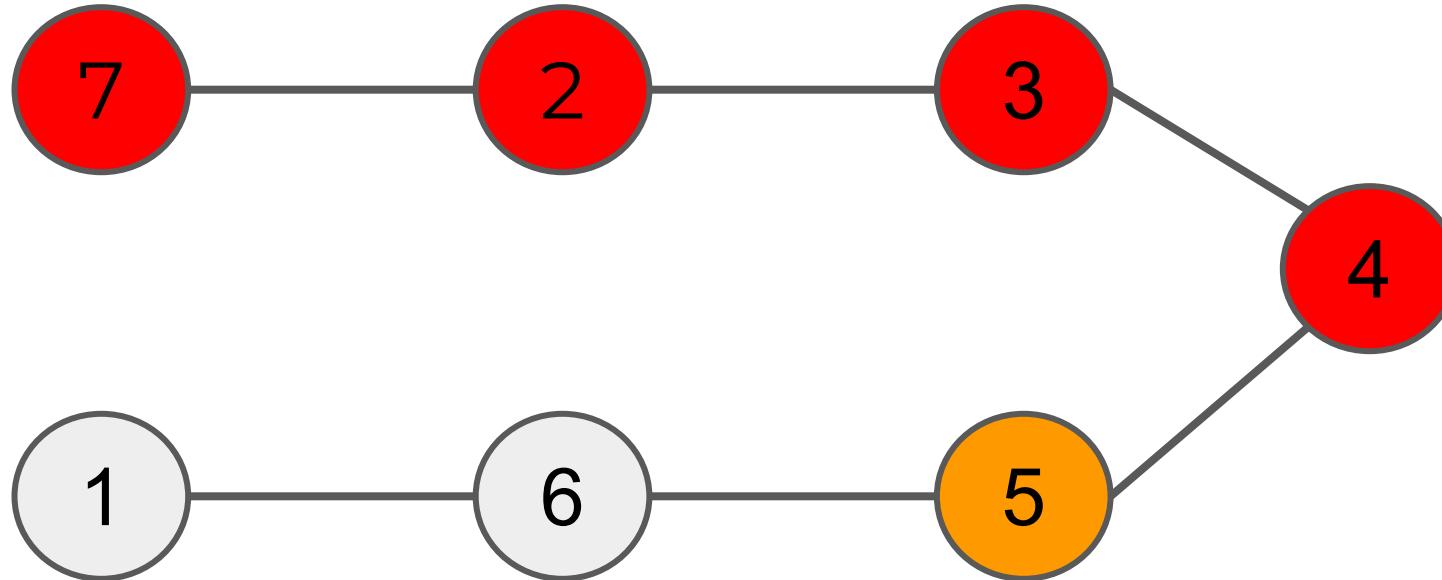


Visit Order: 4, 3, 2, 7

Next level: None
→ Backtrack

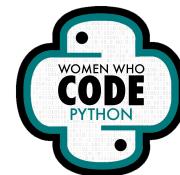


Depth-First Search - Graphs

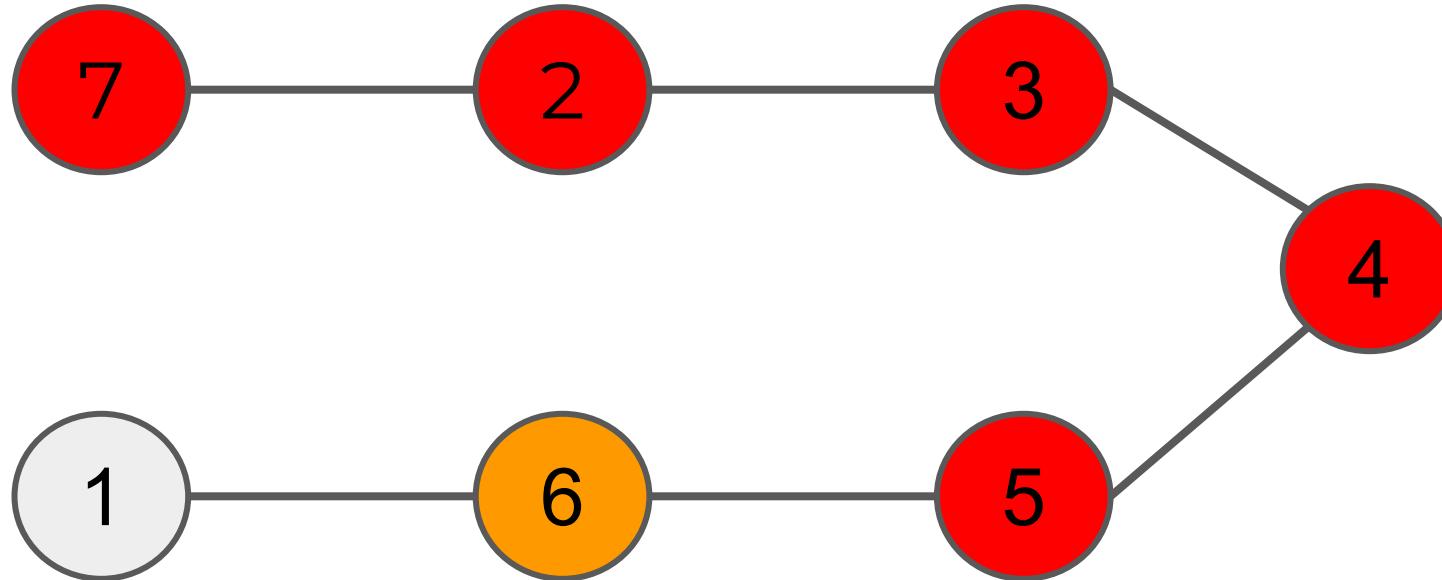


Visit Order: 4, 3, 2, 7

Next: 5

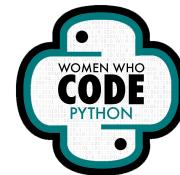


Depth-First Search - Graphs

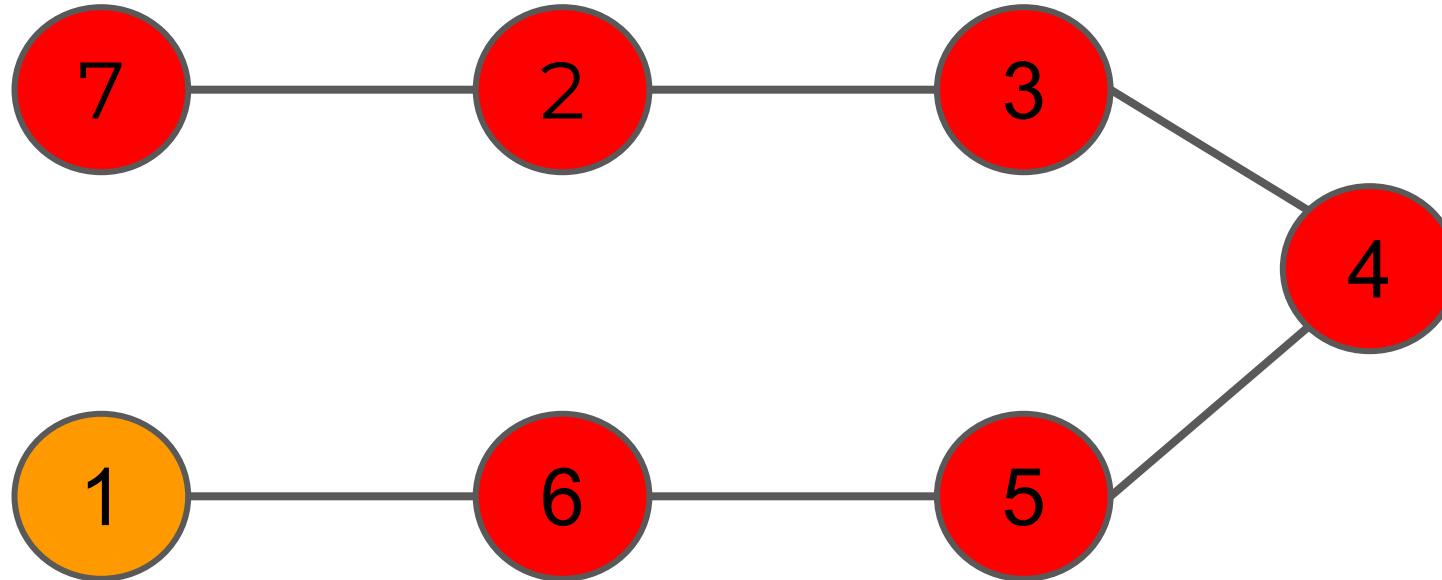


Visit Order: 4, 3, 2, 7, 5

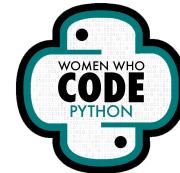
Next level: 6



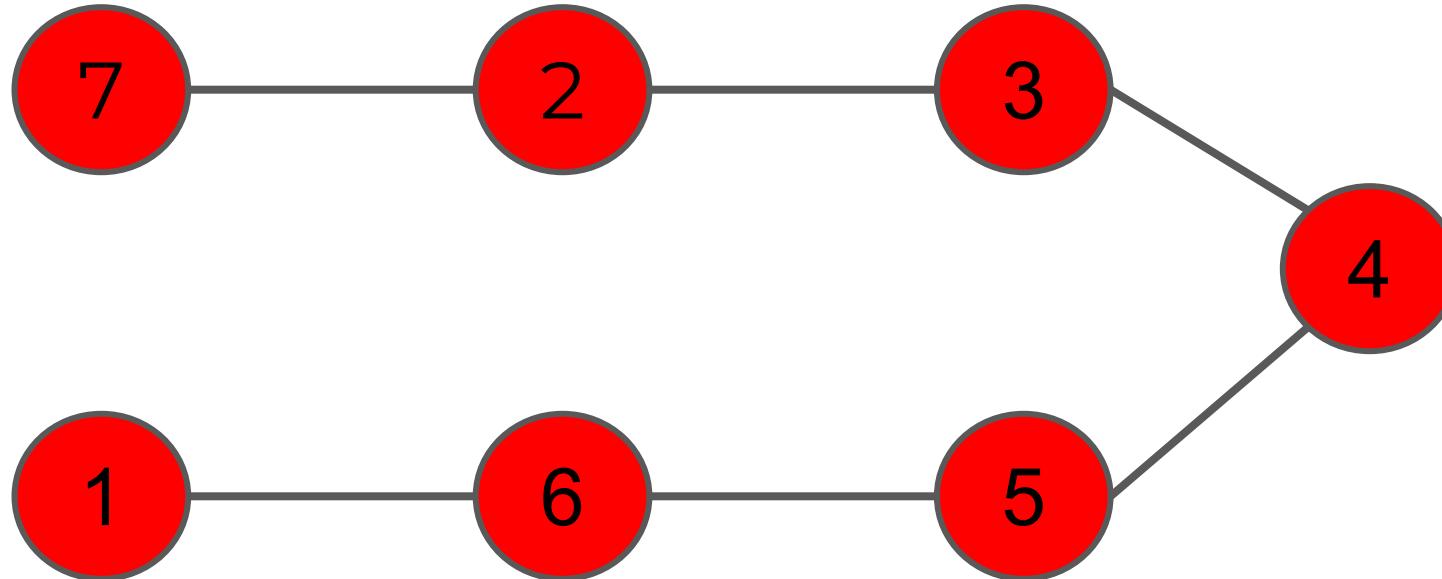
Depth-First Search - Graphs



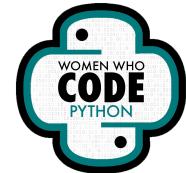
Visit Order: 4, 3, 2, 7, 5, 6 Next level: 1



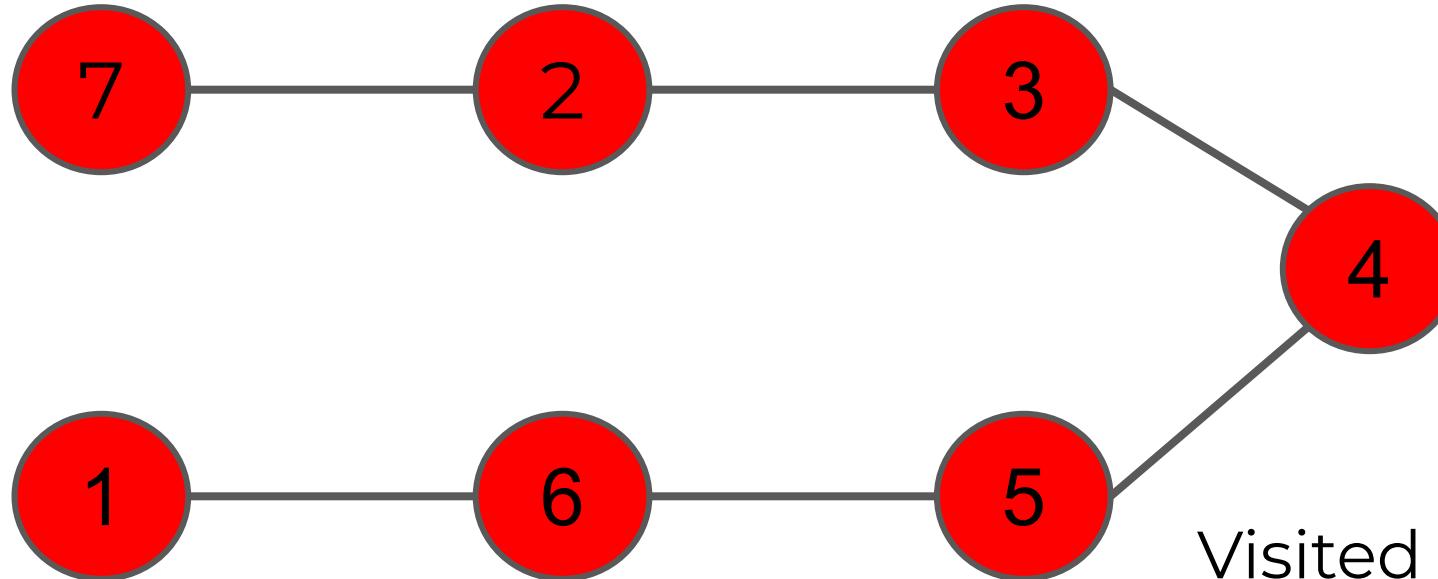
Depth-First Search - Graphs



Visit Order: 4, 3, 2, 7, 5, 6, 1

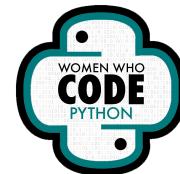


Depth-First Search - Graphs

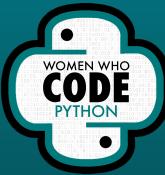


Visit Order: 4, 3, 2, 7, 5, 6, 1

Visited all
vertices
→ Done

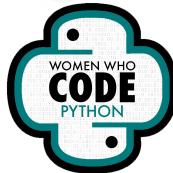


Q&A Time!



Time for Live Coding!

- [https://colab.research.google.com/drive/1fFhbQcQguUPeB0GPxwK9s8X6MUme43NY?](https://colab.research.google.com/drive/1fFhbQcQguUPeB0GPxwK9s8X6MUme43NY?usp=sharing)
usp=sharing

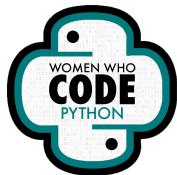
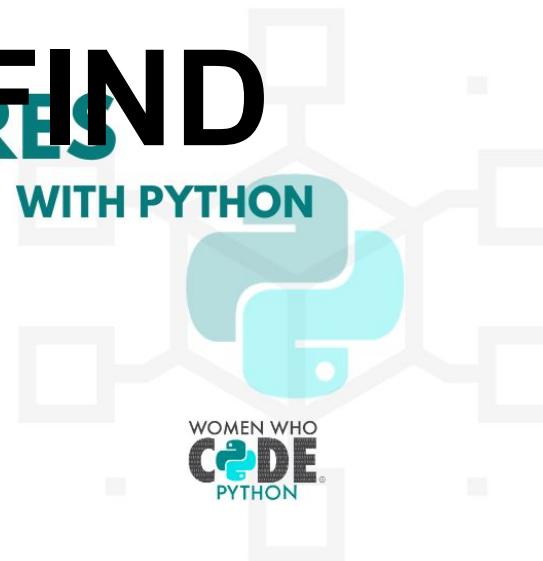


Next Session!

WHERE DO I FIND THE IMAGE?

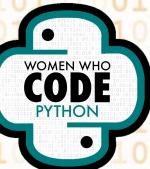
INTRO TO
DATA
STRUCTURES
ACE THE
TECHNICAL
INTERVIEW

THU. JUNE 17TH
@ 8:00PM EDT



Questions?

Join our Slack channel:
#intro-data-structures-stdy-grp



Thank You!

