LeetCamp

A project dedicated to DS&A

Agenda

1. Intro to Graphs
2. Interviewing in C++
3. Practice Problems
4. The End

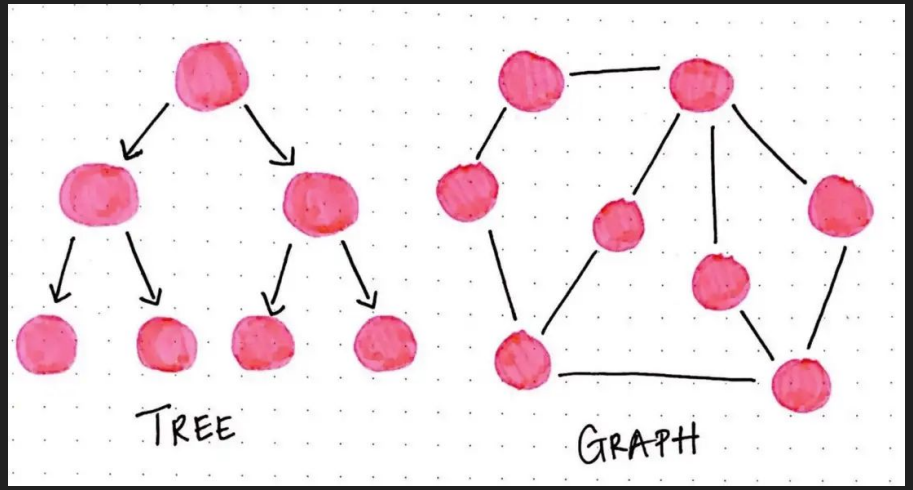
Intro to Graphs

What are Graphs?

- Graphs are non-linear data structures consisting of vertices (nodes) and edges

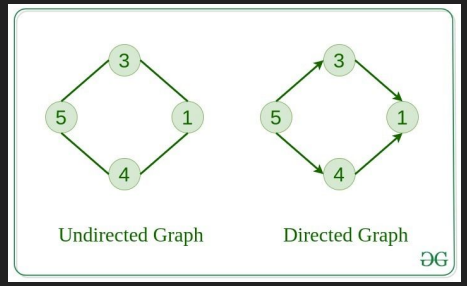
- Vertices
  - each item in a graph
- Edges
  - connections between vertices

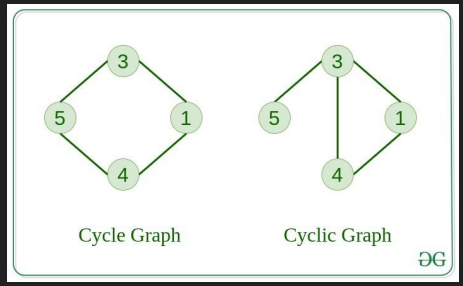
- Graphs are typically denoted as G(V, E)

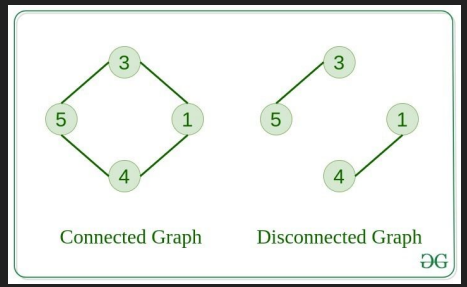


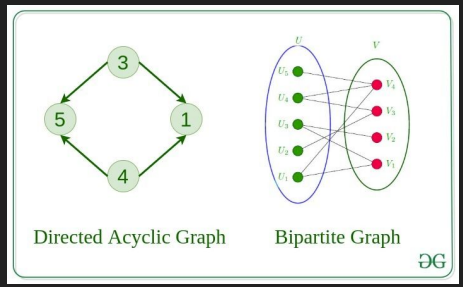
Types of Graphs

- Listed to the right are some types of graphs
- There are many other terms that can be used to describe certain graphs
- We won't go over all of these due to time, but you will have to learn these eventually!





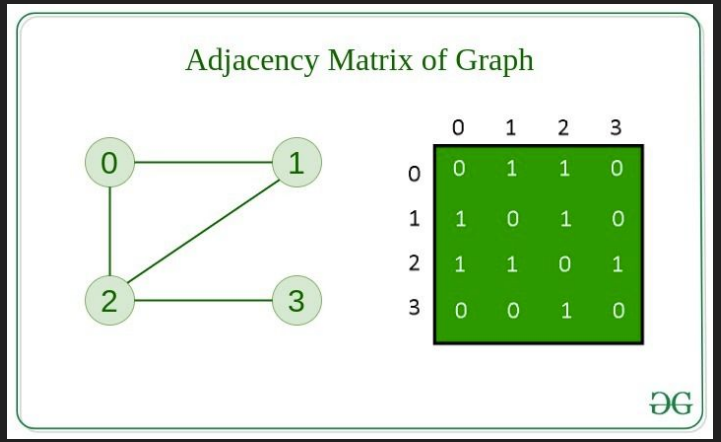


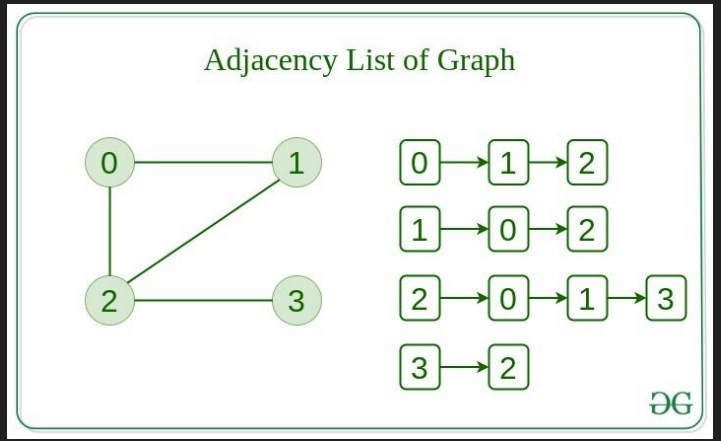


Representation of Graphs

- So how do you store graphs?
- Adjacency List
  - Represent the graph as a collection of linked lists
- Adjacency Matrix
  - Stores 1's and 0's to denote whether or not a node has a connection to another node

Action	Adjacency Matrix	Adjacency List
Adding Edge	O(1)	O(1)
Removing and edge	O(1)	O(N)
Initializing	O(N*N)	O(N)



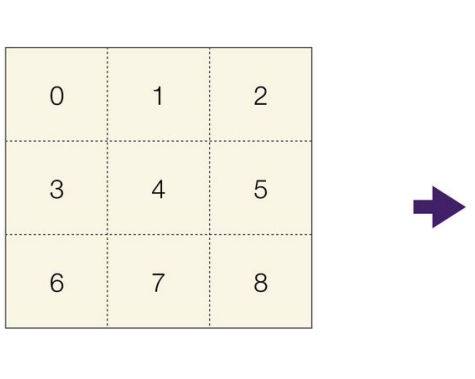


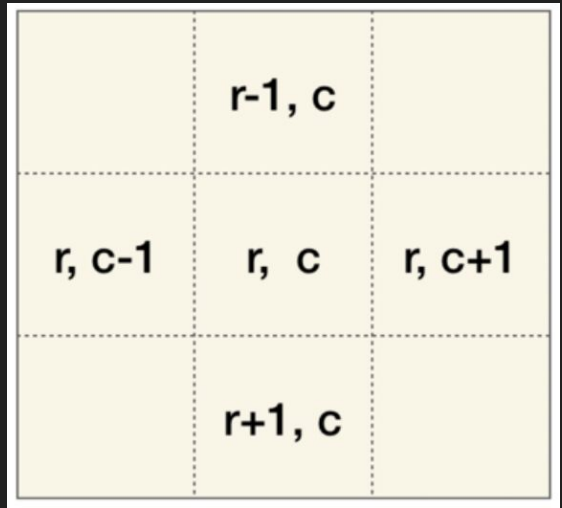
Matrix as a Graph

- You'll often times find problems where you can think of a **matrix as a graph**

- The great part about using a "get_neighbors()" function is that you can adjust for this!

- get_neighbors() just turns into something like the image on the right →



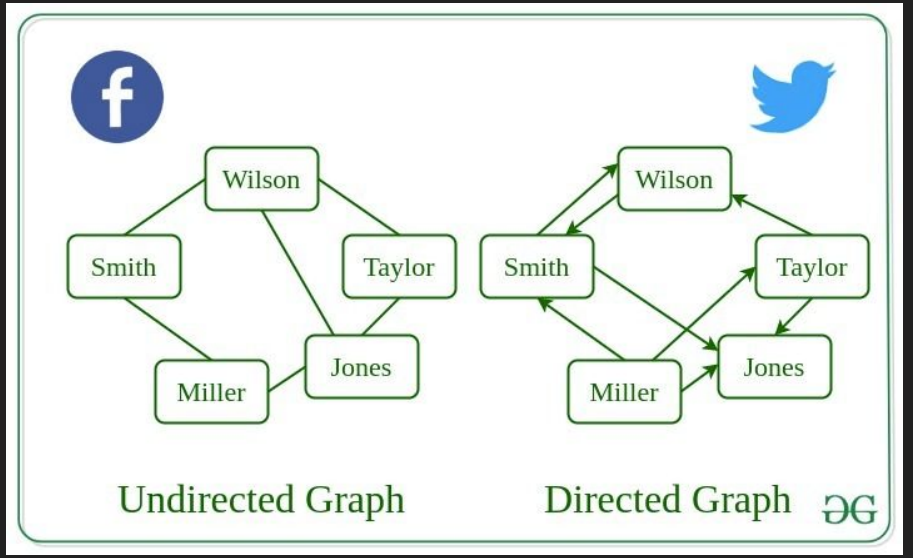


Uses for Graphs

- Social media is a great way to understand graphs!

- Facebook has “friends”, so the connection is undirected
  - Both people agree to the friendship

- Twitter has “follows”, so the connection is directed
  - Only one person follows the other, though they could follow each other



Most Important Graph Algorithms

- The absolute most important algorithms:
  - BFS
  - DFS

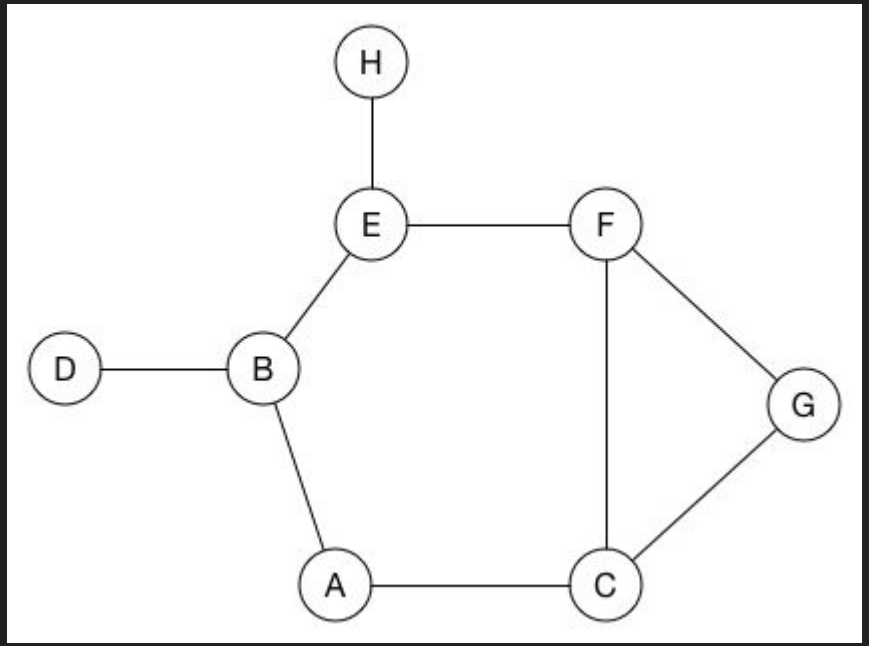
- *You already learned these in a previous LeetCamp lesson?*
  - Great ... but not quite!

- In Trees, you don't have to worry about cycles... In Graphs, you do!

- **Add memory to ensure you don't visit the same nodes twice!**

Breadth-First Search

- Use a “visited” set to remember what you have already seen
- Rather than “children”, graph nodes have “neighbors”
- Just make sure to use a queue, like the tree BFS algorithm!



BFS Boilerplate

```python
def bfs(root):
    queue = deque([root])
    visited = set([root])
    while len(queue) > 0:
        node = queue.popleft()
        for neighbor in get_neighbors(node):
            if neighbor in visited:
                continue
            queue.append(neighbor)
            visited.add(neighbor)
```

Vanilla BFS

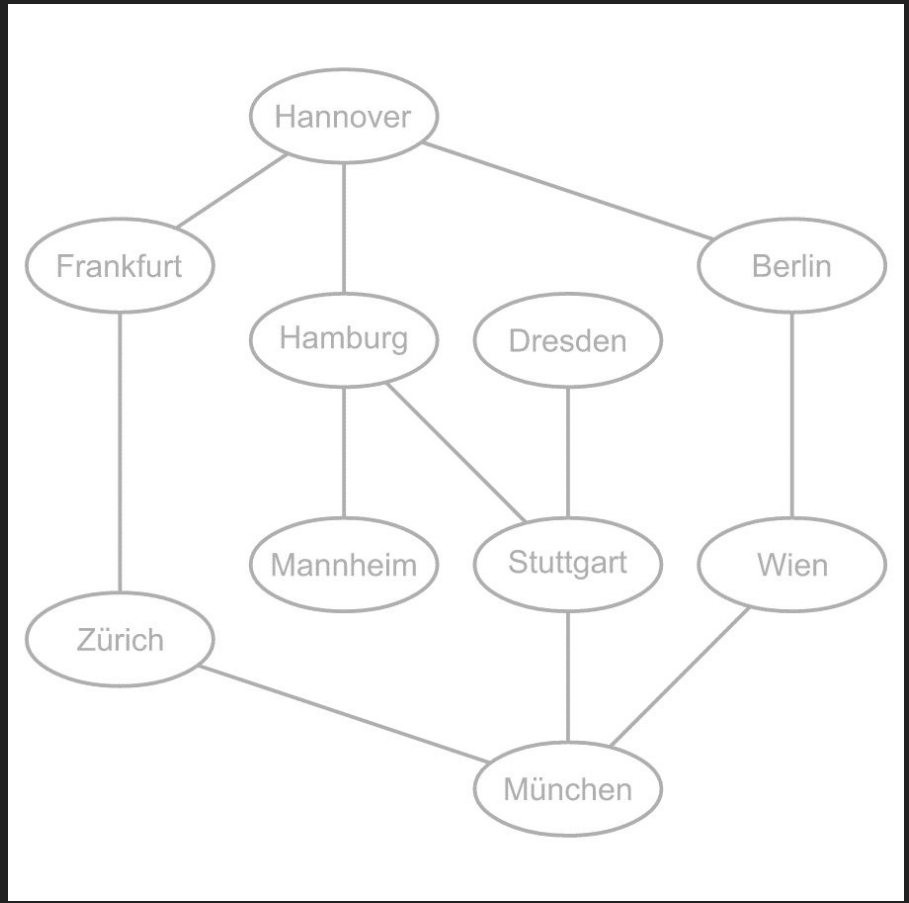
```python
def bfs_level(root):
    queue = deque([root])
    visited = set([root])
    level = 0 # add a level counter
    while len(queue) > 0:
        # the current queue length is the number of nodes in this level
        n = len(queue)
        for _ in range(n):
            node = queue.popleft()
            for neighbor in get_neighbors(node):
                if neighbor in visited:
                    continue
                queue.append(neighbor)
                visited.add(neighbor)
        # increment level
        level += 1
```

Distance-Tracking BFS

Depth-First Search

- I once again recommend implementing this recursively...

- Instead of waiting to visit neighbors, visit them immediately!



DFS Boilerplate

```python
def dfs(root, visited):
    # root is a node, visited is a set
    for neighbor in get_neighbors(root):
        if neighbor in visited:
            continue
        visited.add(neighbor)
        dfs(neighbor, visited)
```

Vanilla DFS

BFS vs DFS

- BFS
  - “Find the shortest distance between two vertices …”
  - Graphs where the size is unknown

- DFS
  - Will use less memory if the graph is *wide*
  - Finding nodes far away

Recommended Graph Problems

- 1091. Shortest Path in Binary Matrix
- 200. Number of Islands (One of my favorites)
- 1197. Minimum Knight Moves (Premium, you can google to find elsewhere)
- 773. Sliding Puzzle (quite hard, very satisfying)

- There are so many algorithms for graphs, go learn about some of them! https://en.wikipedia.org/wiki/Category:Graph_algorithms
    - Kruskal's, Prim's, Ford-Fulkerson, Bellman-Ford, PageRank, Dijkstra's, ...

Interviewing in C++

Interviewing in C++

- Why C++?

- C++ is extremely useful, and widely used across the tech industry ranging from embedded systems, IoT, networking, desktop development, and much much more!!

- The language itself still carries its usefulness even in interviews
  - Helps the interviewer understand your knowledge of memory. How its managed, and what best practices you currently know (or don't know)

- There are some tips you should keep in mind when using C++ for interviews...

Technical Interviews in C++

- You must ALWAYS keep memory in mind
  - Memory management is one of many reasons C++ is still being used today

- Understand what a class is in C++
  - what is difference between public and private?
  - What is a destructor and why do you need it?
  - What happens if you don't include a destructor?

- Pass by Value vs. Pass by Reference:
  - What is the difference, and why would one be better than another?

- Difference between Static and Dynamic Memory Allocation:
  - What are those, and why do they matter?
  - How is a stack overflow detected?

- Difference between Dynamic and Static Linking in the Compilation Process:
  - This is EXTREMELY important

Technical Interviews in C++ cont...

- Interviewing in C++ also likely means you'll be asked about the compilation process
  - I.e. What are the steps involved? What happens how does a C program run?

- NEVER EVER use casting...
  - Do not do it ever unless it's called for in your interview setting
  - It is just REALLY bad practice

- Use vectors instead of traditional C / C++ arrays

- Know what a macro is and how to use it:
  - But do not overuse them

- Multithreading:
  - This one is both a killer and a savior

- Error Handling:
  - Know what a try... catch block is

And More...

- Everything covered in the previous slides is just the tip of the iceberg
  - There is so much more to cover, but here are some other concepts you should study if you ever encounter a C / C++ interview

- These topics show up in any C++ interview (such as Tesla and Apple):
  - Bit Manipulation (VERY VERY IMPORTANT!!!!!)
  - Finite State Machines ← From Tesla Interview
  - String Manipulation ← From Apple Interview
    - Strings in C++ are mutable, meaning you can change them after declaration
    - This is great, but why?
      - C++ is very desirable when working with data you want to be mutable
  - Matrices
  - Linked Lists
  - Trees and Hashing (If you're interviewing for a Database Engineering position)

Finite State Machines in C / C++

- States are always represented as Enums:

```
typedef enum {
    STATE_ONE,
    STATE_TWO,
    STATE_THREE
} states;
```

- But that's not all, because next you must iterate through each state until you reach the end

- While iterating, you can use a switch statement for your current state
  - In the condition where your state matches, you do more coder magic stuff...

```
enum {
    S_0 = 0,
    S_1 = 1,
    …,
    S_stop = N
} state;

int errcod, firings;
state = S_0;
errcod = 0;
while( errcod == 0 && state != S_stop ) {
    firings = 0;
    switch( state ) {
        case S_0:
            if( C(0,0) ) {
                state = S_0;
                firings = firings +1;
            }

            if( C(0,1) ) {
                state = S_1;
                firings = firings +1;
            }

            if( C(0,2) ) {
                state = S_2;
                firings = firings +1;
            }

            if( C(0,N) ) {
                state = S_stop;
                firings = firings +1;
            }

            break;

        case S_1:
            if( C(1,N) ) {
                state = S_stop;
                firings = firings +1;
            }

            break;

        default:
            error_code = -1;

    }

    if( firings > 1 ) {
        errcod = firings;

    }
}

return errcod;
```

Bit Manipulation

- This will be VERY brief

- These are the bitwise operators:
  - ^, >>, <<, &, |, !
  - Can anyone explain what they represent and do?

- You will find that bit manipulation is typically used in a question regarding integers
  - Integers decay to bits, and because of that there are a lot of efficient work arounds you can do compared to traditional arithmetic operations

Bit Manipulation Example

```
int swapBits(unsigned int x, unsigned int p1, unsigned int p2, unsigned int n){
    unsigned int set1, set2, temp, result;
    set1 = (x >> p1) & ((1U << n) - 1);
    set2 = (x >> p2) & ((1U << n) - 1);
    temp = set1 ^ set2;
    temp = (temp << p1) | (temp << p2);
    result = x ^ temp;
    return result;
}
```

Practice Problems

LeetCode Problems That We Love

- There are many great LeetCode problems, but there are also terrible ones...

- We want *one last opportunity* to list out some LeetCode problems that we truly love!

- We hope you will try to get to a point where you can complete these problems with ease!

Andrew's Favorite Problems

- 744. Find Smallest Letter Greater Than Target (essential binary search)
- 236. Lowest Common Ancestor of a Binary Tree
- 141. Linked List Cycle (an elegant solution, try to get there yourself)
- 217. Contains Duplicate (try to really understand *how* it works [hashmaps])
- 200. Number of Islands (essential graph problem)
- 204. Count Primes (if you are into number theory...)

Overall Favorite Problem:

- 1197. Minimum Knight Moves

Patrick's Favorite Problems

- Pow(x, n):
  - https://leetcode.com/problems/powx-n/description/
- String Compression:
  - https://leetcode.com/problems/string-compression/description/
- Group Anagrams:
  - https://leetcode.com/problems/group-anagrams/description/
- Two Sum 3 - Data Structure Design (Premium Question):
  - https://leetcode.com/problems/two-sum-iii-data-structure-design/
- Two Sum Less Than K (Also a Premium Question):
  - https://leetcode.com/problems/two-sum-less-than-k/description/
- Add Two Numbers:
  - https://leetcode.com/problems/add-two-numbers/description/

The End ☹️

For now....

Contact Us!

- Feel free to connect with us!

- Reach out if you ever have any questions, DS&A related or not!

- LinkedIn
  - Andrew Fennell - @andrew-fennell
  - Patrick Apgar - @patrickapgar

Going Forward...

- Andrew is graduating!
  - Whoop!!!!
- Patrick will be continuing LeetCamp into next semester!
- Please join LeetCamp in the Spring for more awesome DS&A content!
- And fill out this amazing survey form please:
  - https://tx.ag/LCSurvey

Thank you for attending!