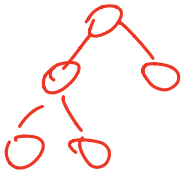
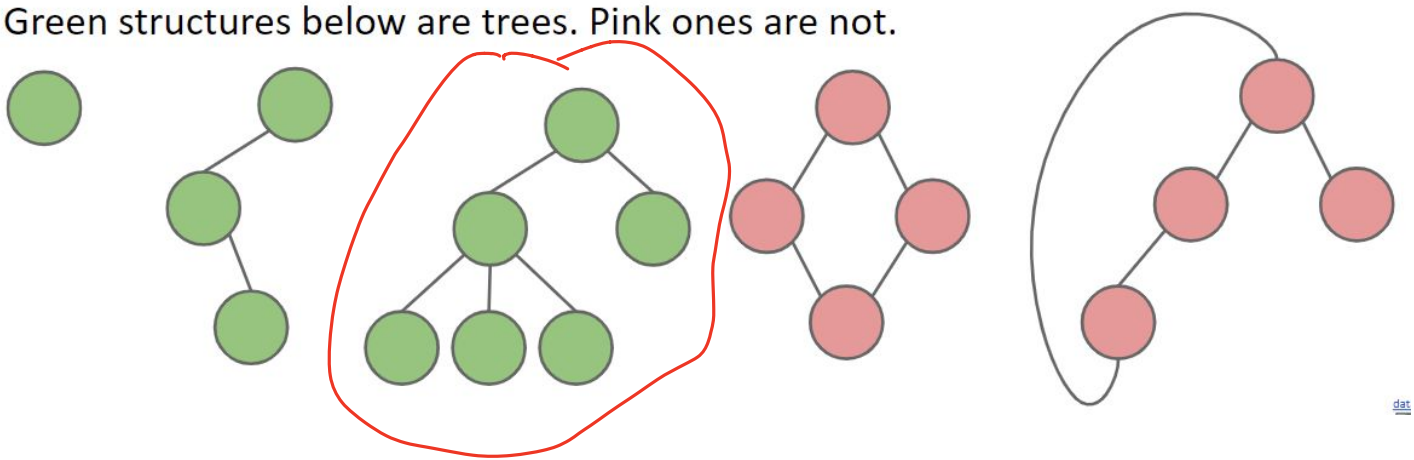
Trees

Hug's Slides: Traversals, MSTs

What?



- We've seen many examples of trees already! - BST, B-Trees, RB-Trees
- Basically an unconnected graph
  - Can be used more generally than a BST, storing more than left and right nodes



Types of Traversals

- See Visuals Here
- DFS
- ~~Level-Order (BFS)~~

* Usually more than L/R branches, break ties alphanumerically

* Recursive calls

1. Process Node
2. visit left branch
3. visit right branch

DBACFEG (Pre)

1. visit left branch
2. Process Node
3. visit Right branch

ABCDEFG (In)

1. visit left
2. visit branch
3. Process Node

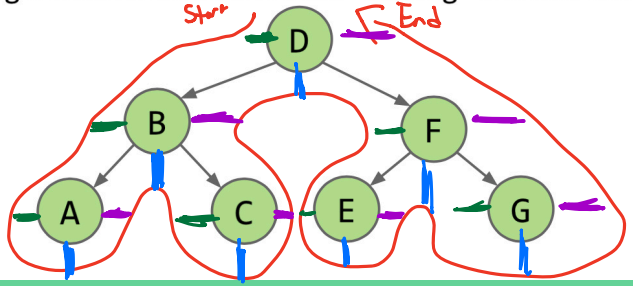
ACBEGFD (Post)

Depth First Traversals

- 3 types: Preorder, Inorder, Postorder
- Basic (rough) idea: Traverse "deep nodes" (e.g. A) before shallow ones (e.g. F).
- Note: Traversing a node is different than "visiting" a node. See next slide.

Quick Tricks

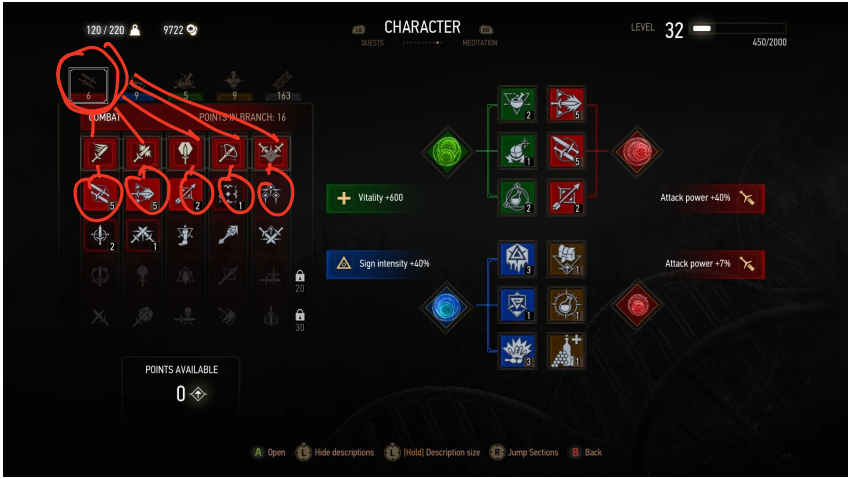
- Pre (Left)
- In (Bottom)
- Post (Right)



datastructur.es

Why?

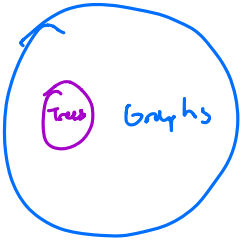
- Represents hierarchical relationships (ex. Folders in your directory!)
- Video Games (skill trees)
- Specific type of a more important idea - Graphs



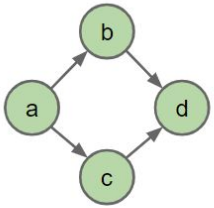
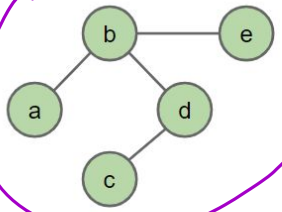
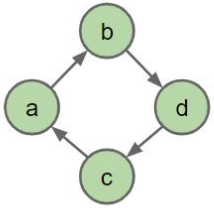
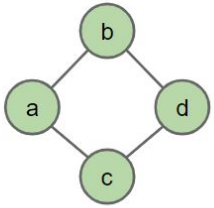
Graphs

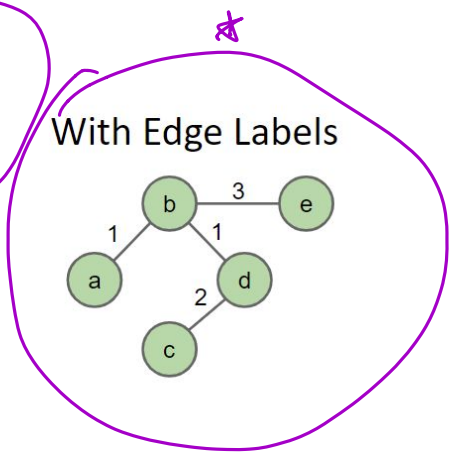
Hug's Slides: Traversals, Implementation, Shortest Paths

What



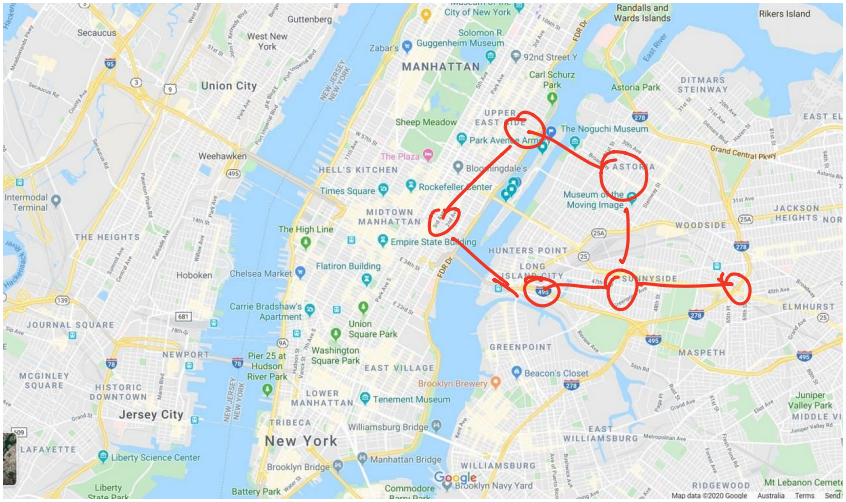
Graph Types

	Directed	Undirected
Acyclic:		
Cyclic:		



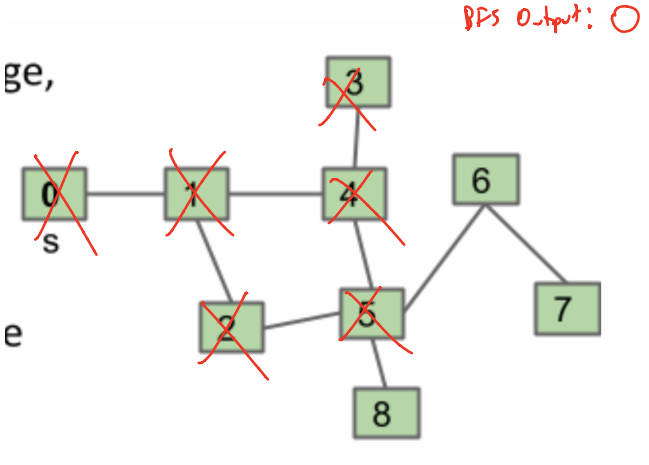
Why?

- Represents so many problems!
- Google Maps, Video Games, Islands, etc.



Queues, Stacks, BFS/DFS problems/implementations

BFS Output: 0 1 2 4 5 3 6 8 7



BFS — Queue (FIFO): [~~4~~, ~~5~~, ~~3~~, 6, 8, 7]

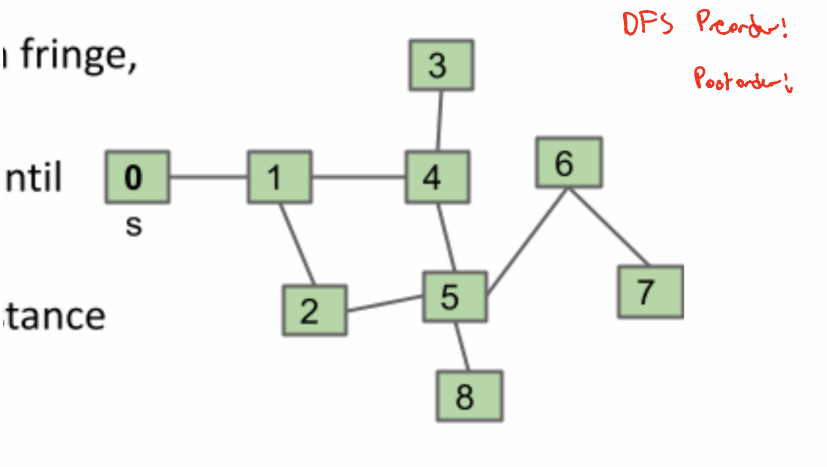
1. Add neighbors to Queue (break ties by alphanumeric order)
2. Process Node (.poll(), print( )

visited = { 0, 1, ... }

Python Ex: [_, ... , _]
.popleft() .pop()
Queue Stack

DFS Preorder:

Postorder:



DFS – Stack (LIFO): [ ]

Postorder ✗ 1. Add unvisited neighbors (deeply) until reach visited node

2. Process Node (.pop(), print( ))

visited = { 0, 1, ... }