course_content

top W10: Graphs

W11: Greedy Algorithms

Data Structures and Algorithms

Intro to Graphs

Week 10

Time:
10 min

Graphs - Activity (self-work)

- Graphs are non-linear data structures that are more general than trees
  - No specific root
  - Edges can be two-way (no specified parent-tree)
  - Cycles are allowed
- What do the following mean in the context of graphs?
  - Vertex, Edges (directed and undirected), Weight, Cycle, Subgraph, Loop
  - Connected, Adjacent, Neighbours, Complete, Walk, Path, Cycle, Degree

Time:
5 min

Graph Applications

- Graphs are incredibly useful. Many things can be expressed in graphs
    - Computer Networks (computers and interactions between them)
    - Social Media Networks (people and friend relationships)
    - Google Maps (locations and paths between them)
    - World Wide Web (Pages and links between them)
- Try to think of at least 5 other applications of Graphs. Especially think about your project. Graphs/Trees are almost certainly going to be useful somewhere in your project. How? Do you have multiple of something and some form of relationship between them?

Time:
15 min

Implementation - Adjacency Lists

- One way of implementing graphs is using ‘adjacency lists’
  - Every vertex/node has a corresponding list of elements that are adjacent to it
  - For an ‘undirected graph’ (e.g. FB friends), you have to store adjacency on both ends - A is in B’s friend list, B is in A’s friend list.
  - Here, a ‘list’ actually just means any linear data structure. A hash table is often used because it is easy to check if an element is in the ‘list’ - O(1) and the value stored can be a ‘weight’ e.g. strength of friendship.
  - The set of all nodes can also be stored using any Data Structure but hash tables are good for quick access.

Time:
20 min

Implementation - Primary Operations

- There are ‘vertex’ (or ‘node’) classes and a larger ‘graph’ class.
- What do you think are the primary methods of each?

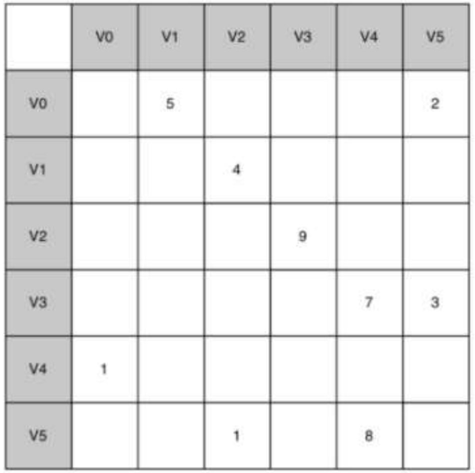
Using just your ‘primary methods’ how would you do the following? Write Pseudocode

- Compute the total number of edges in your graph
- Search if there is an edge from vertex A to vertex B
- Insert an edge from vertex A to vertex B
- Decrease the weight of all edges by 1

Time:
15 min

Implementation - Adjacency Matrix

- An alternative way of representing graphs is through a matrix
- There is one row and one column for every vertex
- The value in row i/col j is the weight for that edge
- Undirected graphs are symmetric row i/col j = col i/row j
- If there are no weights, you can just do 0/1
- The matrix itself can be a list of lists or a table
- This takes up a lot of space
- Good for getting an overview of your entire graph



	V0	V1	V2	V3	V4	V5
V0		5				2
V1			4			
V2				9		
V3					7	3
V4	1					
V5			1		8	

Figure 3: An Adjacency Matrix Representation for a Graph

Time:
10 min

Implementation - Edge List

- Another way is just to store every single edge (and it's weight) inside a big set.
- {(1,2),(1,3)} has two edges from 1 to 2 and 1 to 3.
- If there are weights, you could also represent it as a hash table with keys (1,2),(1,3) and values as your weights.
- The advantage of an edge list is that is good when the 'most important' objects are the edges themselves (e.g. knowing the total number of edges, determining if an edge exists) but not great if you want to consider a vertex and the edges it has.

Time:
15 min

Videos

- My Code School - Intro to Graphs
- Geeks for Geeks - Graph Representations

Time:
15 min

Readings

- Geeks for Geeks - Graphs DSA
- Course Textbook - Chapters 8.1-8.6
- Khan Academy - Graph Representations
- Towards Data Science - Representing Graphs
- Brilliant - Graph Representations

Time:
10 min

Quiz Time

bit.ly/DSA1920Quiz9

Deadline: Sunday 5th April (midnight)

Time:
1 min

Questions?....Office Hours!