course_content

top

W10: Graphs

W11: Greedy Algorithms

Data Structures and Algorithms

Graph Algorithms

Week 10

Time:
10 min

Graph Algorithms - Activity

- Once we have a graph, we want to do things with graphs.
- Try to think about/imagine how you would do the following graph algorithms.
  - Find the node with the largest value at the node
  - Find if there is a node with a specific value
  - Given two nodes, find a path between them
- This should be hard. If it’s too much, try it with a tree first.
- What is the issue when we introduce the possibility of ‘cycles’?
- In general searching is very important in graphs!

Time:
5 min

Breadth First Search

- Graph G and starting vertex s
- Finds all vertices 1 away from s, 2 away from s, 3 away from s.....
- Puts all vertices 1 away from s into a queue e.g. a,b,c,d,e
- Then runs through queue again finding vertices 1 away from each of a,b,c,d,e putting those into the queue and then repeats
- If we reach a vertex previously visited, don't re-explore (e.g. check stored set)
- Through this, we can store the max, look for a specific value, find path to vertex t

Look at it here, visualise it here, read about it here

Time:
5 min

Breadth First Search - Applications

- Web Crawlers (in Search Engines) - to index popularity of websites, crawler needs to search through all possible paths
- Torrenting (P2P sharing) - Checks peers for files already downloaded
- Basic shortest path algorithms (e.g. Google Maps, Flights)
- Social Networks (E.g. In LinkedIn, determine if connection is 1st/2nd/3rd+ degree)
- Spell check - looks for similar words with slight changes (related to this)
- Read more here

Time:
5 min

Breadth First Search - Complexity

- 2 different inputs here - V is the number of vertices, E is the number of edges.
- **Adjacency List** - O(V+E) as we have to visit every vertex exactly once and for each vertex we have to get all of its neighbours each of which is through an edge.
- **Adjacency Matrix** - $O(V^2)$ since for each vertex, we have to go through the array/list for that vertex which is length V.
- **Edge List** - An edge list would be very poor here as you wouldn't have a nice way of getting which edges are adjacent to any given vertex.

E can be $O(V^2)$ if there are many edges (since each each connects 2 vertices so at most "n choose 2" edges but for most graphs E is much smaller than V^2.

Time:
5 min

Depth First Search

- Graph G and starting vertex s
- Picks a neighbour of s and adds it to a stack. E.g. a
- Picks an unvisited neighbour of a and adds it to a stack. E.g. b
- Visited neighbours are checked through a set (hash map) - O(1) for each check
- Eventually we reach a vertex where all neighbours have already been visited.
- Remove this from the stack and again check for an unvisited neighbour of the top
- Keep going until the entire stack is empty (it will grow again)

Look at it here, visualise it here, read about it here

Time:
5 min

Depth First Search - Applications

- Testing if a network is ‘connected’. (E.g. will message reach everyone in network)
- Shortest Path Algorithm Alternative (Flights, Google maps etc) but this generally performs poorly compared to other algorithms
- Detecting cycles (e.g. in a schedule, a cycle would make it impossible to complete)
- Topological Sorting - Creates an ordering of vertices (in a directed graph) where vertices that come before lead to vertices that come after. More here.
- A variety of interview questions

Time:
5 min

Depth First Search - Complexity

- **Adjacency List** - O(V+E) as we again have to visit every vertex exactly once and for each vertex we have to access/check each of its neighbours until it is complete
- **Adjacency Matrix** - $O(V^2)$ since again for each vertex, we have to go through the array/list for that vertex which is length V to get the adjacent vertices.
- **Edge List** - An edge list would be very poor here as you wouldn't have a nice way of getting which edges are adjacent to any given vertex.

Can't do parallel computing as we shouldn't take next item in stack without finish top

Time:
10 min

Graph Algorithms

- There are many other graph algorithms. You can explore some of them:
  - Dijkstra's Algorithm is the standard ‘shortest path’ algorithm that uses a priority queue (built using a heap)
  - Prim's Spanning Tree Algorithm finds the tree from a central root to every other vertex that minimises the time taken to ‘broadcast a message out’
  - Ford-Fulkerson Algorithm for sending the maximal ‘flow’ from one node through to another restricted by the weights on the edges.
- A lot of graph algorithms are incredibly useful for computer science. They are explored more in the 4th-year elective ‘Graph Theory’.

Time:
1 min

Questions?....Office Hours!