CS2040 Lab 9
Graph Traversal

Take Home Assignment 3 – Ladice

- Key observation: once an item is put in a drawer (any drawer), it is never discarded
- If a series of swaps between drawers allows a new item to be added in, then it is considered a valid addition
- Therefore, knowing exactly which drawer the item is in is not necessary; it suffices to know the set of drawers that the item *could* be in, as well as whether this set of drawers has the capacity to fit an additional item
- Implies the need to keep track of how many items are in a set of drawers

Take Home Assignment 3 – Ladice

- Example: Initially we have 4 empty drawers numbered 1 to 4
- 2 items are inserted, with Ai and Bi of (1, 2) and (3, 4) respectively
- We then attempt to insert a new item with Ai, Bi = (1, 3).
- 3 possibilities exist (assuming we do not know exactly which drawer is occupied, but we know (1, 2) and (3, 4) have 1 item each):
  - 1. Drawer 1 is empty. We put the new item into drawer 1.
  - 2. Drawer 1 is occupied but 3 is not. We put the new item into drawer 3.
  - 3. Both drawers 1 and 3 are occupied. The item currently in drawer 1 will shift to drawer 2, allowing the new item to be put into drawer 1.
- In all cases, the new item can be added. Therefore, keeping track of the exact location of the item is not necessary.

Take Home Assignment 3 – Ladice

- Start off with a UFDS representing all drawers. Initially, they are all disjoint, have a size of 1, and hold 0 items
- When we attempt to add a new item which will fit in drawers A or B, check if either of the sets A and B belong to (note: they may be in the same set) can support an additional item (ie. item count < size)
  - If so, union A and B if they are not in the same set already, and sum up the size and item count of both sets together
  - Increment the item count of this set by 1
- If neither A nor B can support a new item, then this item is discarded

Take Home Assignment 3 – GCPC

- Write a custom AVL class
- Each node in the AVL needs to store (minimally) the number of problems a team has solved, and the penalty a team has
- There can be multiple teams with the same number of problems solved/penalty. To address this:
  - Include the team number in each node as well, making each node distinct; or
  - Store a frequency count in each node, so multiple teams with the same number of problems solved/penalty will be represented by a single node

Take Home Assignment 3 – GCPC

- When a team solves a problem, we need to know how many solved problems/penalty the team has already, before we can find it in the AVL tree
  - Use an array (or HashMap, but array is preferable here) to map team number -> (problems solved, penalty)
- To find the rank of team 1, implement a rank(x) method (similar to tutorial 7), where x is the information for team 1 (ie. problems solved & penalty)
  - Requires updating a size attribute as well

Take Home Assignment 3 – GCPC

- AVL deletion can be tricky
- Instead of actually deleting the node, can lazily delete the node (eg. mark the node as "not in use")
  - "Not in use" nodes should not contribute to the size attribute
- By doing this, the number of nodes in the AVL tree can reach (n + m) as opposed to just (n) for an AVL tree with actual deletion, but will still run fast enough for this problem

Lab 9 – Graph Traversal

- Two different forms of graph traversal are covered: Breadth First Search (BFS) and Depth First Search (DFS)
  - BFS tends to employ the use of a queue, while DFS uses an implicit stack via recursion
- Pseudocode for both can be found in the lecture slides
- Important note for BFS: mark a vertex as visited when you add it to the queue, as opposed to when you remove the vertex from the queue. Otherwise, you may end up enqueueing the same vertex multiple times, which will result in a slow program

One Day Assignment 8 – Islands

- Count the minimum number of islands present in the graph
- ‘L’ cells – definitely land
- ‘W’ cells – definitely water
- ‘C’ cells – could be either land or water

One Day Assignment 8 – Examples

- CCCCC
- CCCCC
- CCCCC
- CCCCC

- Assume all the clouds are water, answer is 0
- LW
- CC
- WL
- Assume the 2 clouds are land, answer is 1

One Day Assignment 8 – Notes

- In these 2 examples all the clouds are water or all the clouds are land, however clouds need not be assigned graph-wide to all water or all clouds (ie. there can be both clouds that are water, and clouds that are land in the same graph) eg:
- LCCL
- WWWW
- CCCC
- While the graph is not given in the form of any of the DSes we have taught (Adj Matrix, Adj List, Edge List), it may be easier to leave the graph as it is, and modify your algorithm to account for the grid structure, instead of explicitly converting it to a graph DS you have learnt