Module #11 : General Graphs

Lists :-

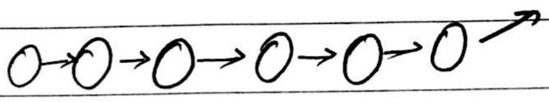
- 'head' & 'tail' (special nodes)

```
class Node {
  private Node next;
  private Object data;
}
```

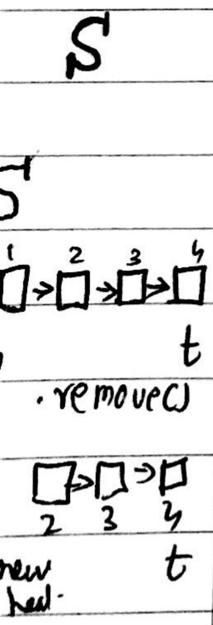
Queues :-

[Coinbillboard Queues Websocket]

- Only add at tail.
- Only remove from head



.add() .remove()



Trees :-

Special node : root

→ N edges from any node.

→ ~~Edges out of a node~~ are represented by an ArrayList in Node class.

```
class Node {
  private Object data;
  private ArrayList<Node> children;
}
```

General Graphs :-

- No special nodes.
- Edges often have associated weight
  - ex: miles b/w two cities
- Associated => Use a map

Common graph problems

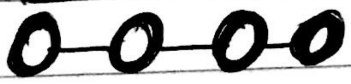
E = # of edges ; V = # of vertices

How to solve graph problems :—

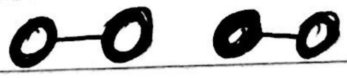
- Nodes & Edges
- Think about pics
- Draw pics.
- After these 3, write code.

Example: Is a graph connected

Connected



Disconnected



1. Create HashSet <Node>
2. Pick any node & add it to the set

Design Goals :—

graph.connect ("SFO", "LAX") - Easy method for buildin graph by specifying conn

TODO - Connect Page 47.