Adjacency Matrix

Checking if there exists an edge between two nodes? O(1).

Iterating through all neighbours? O(|V|)

Space? $O(|V|^2)$

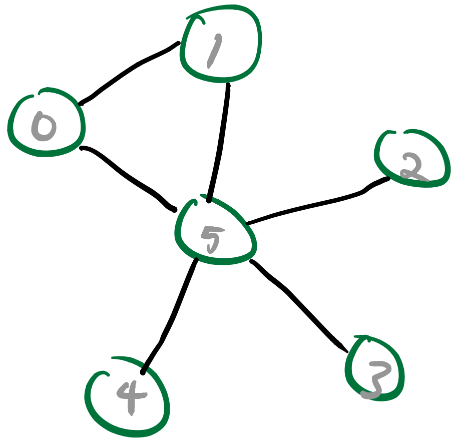
Adding an edge between vertices v and w? connections[v][w]=1

undirected!

(u connected v ⇒ v connected to u)

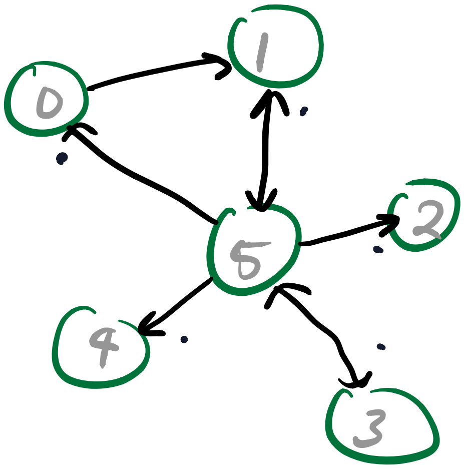
"symmetric".

	0	1	2	3	4	5
0	0	1	0	0	0	1
1	1	0	0	0	0	1
2	0	0	0	0	0	1
3	0	0	0	0	0	1
4	0	0	0	0	0	1
5	1	1	1	1	1	0



directed!

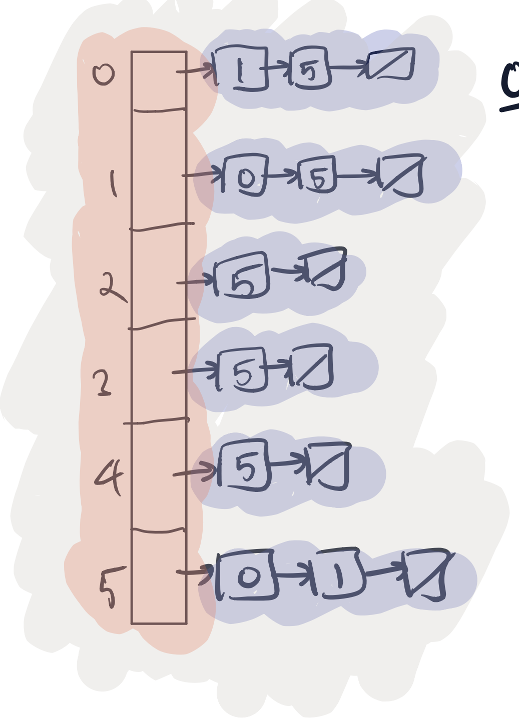
	0	1	2	3	4	5
0	0	1	0	0	0	0
1	0	0	0	0	0	1
2						
3						
4						
5	1	1	1	1	1	0



```
struct GraphRep {
    int nV.
    Node *connections;
```

Adjacency List

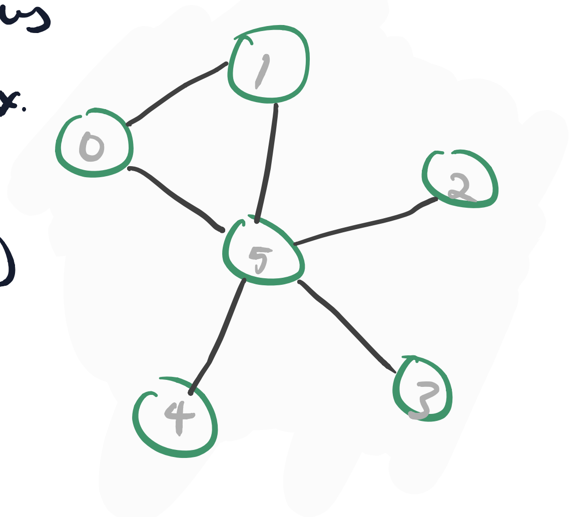
}



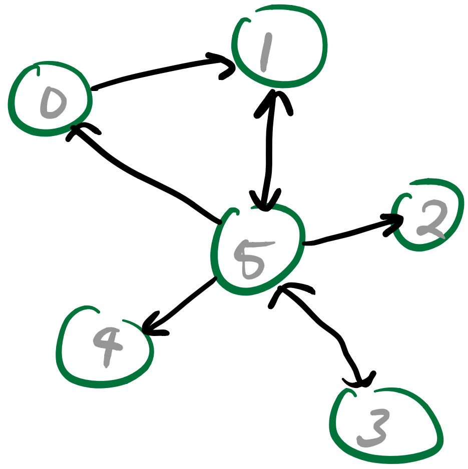
Only store neighbours for a given vertex.

Space?

$O(|V| + |E|)$



Left as an exercise to the reader :)



Questions.

(0) Time complexity to check if two vertices are neighbours? Adding an edge?

(1) When is it appropriate to use an adj. matrix? similarly for adj list.

Extension : There is another representation of graphs that is useful in Functional Programming. Introduced in Erwig (2001) it's coined an inductive graph. Inductive because it's defined in terms of itself.

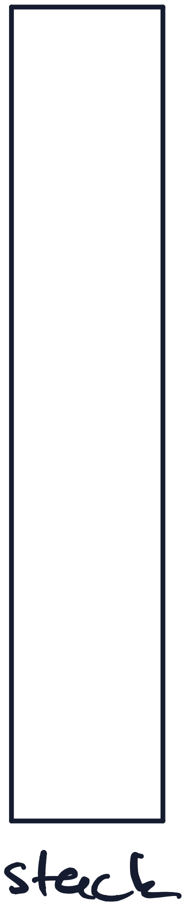
Try and design an inductive graph.

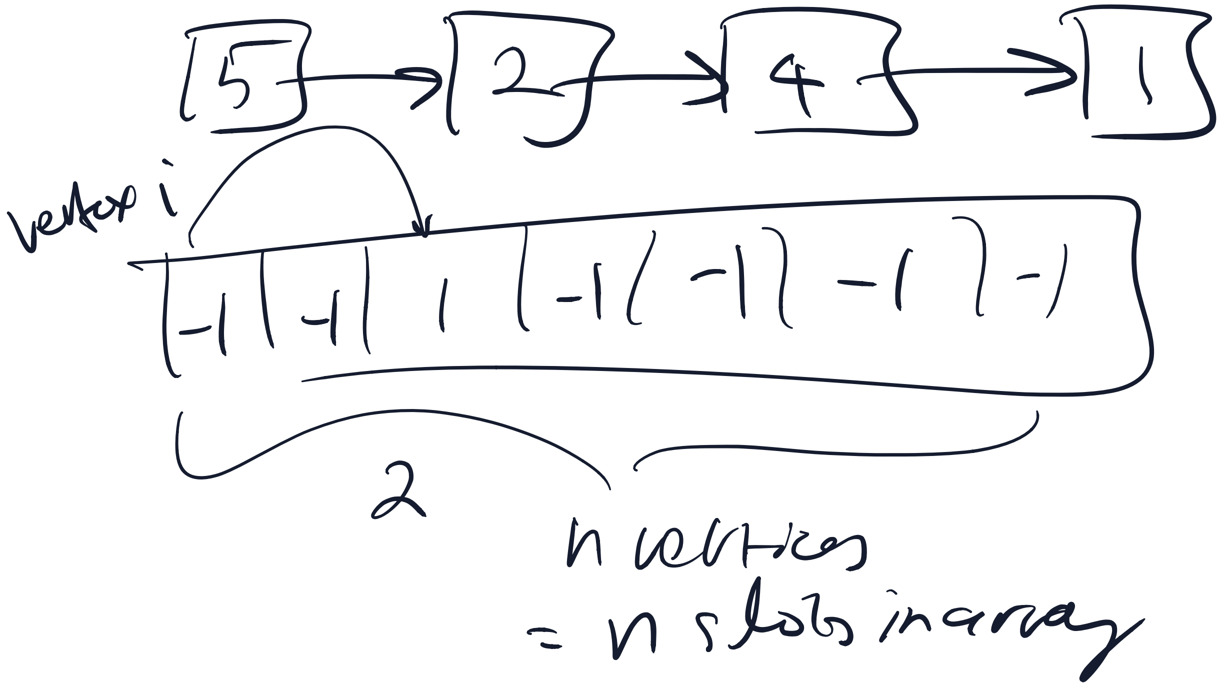
C representations.

```
struct GraphRep {

    int  nV
    int  **connections

}
```

```
struct GraphRep {

    int nV
    Node connections []

}

typedef struct _node {

    int vertex
    struct _node * next

}  *Node.
```

TRAVERSING GRAPHS.





$$\begin{array}{c|cccccc} & & & & v & & \\ & 0 & 1 & 2 & 3 & 4 & 5 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 & 0 \\ 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 & 0 & 0 & 0 \end{array}$$

$O(|V|^2)$