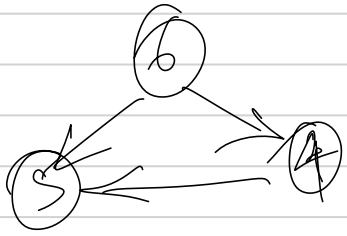
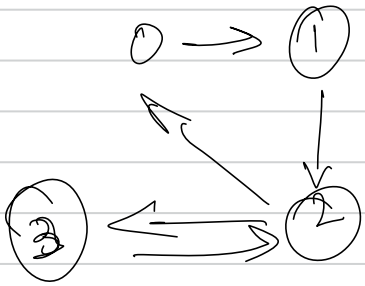
Graphs

1. Graphs can be directional or undirectional
2. multiple isolated subgraphs
3. An acyclic graph (or) cyclic





Adjacency list

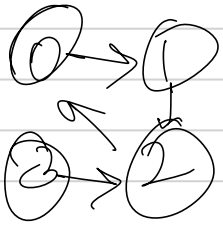
Every vertex stores a list of adjacent vertices.

An array of list can be used to represent a graph.

eg: 0:1 → 1:2 → 2:0,3 → 3:2 → 4:6, 5 ; 4,6,5

Adjacency Matrix:

N × N matrix to denote graph.



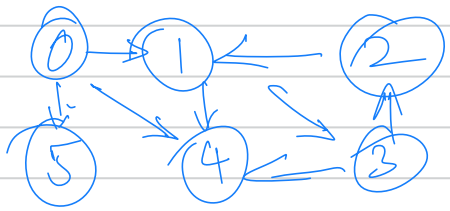
	0	1	2	3
0	0	1	0	0
1	0	0	1	0
2	1	0	0	0
3	0	0	1	0

Graph Search

Depth first

Start at Root

Explore each branch before going on to next branch

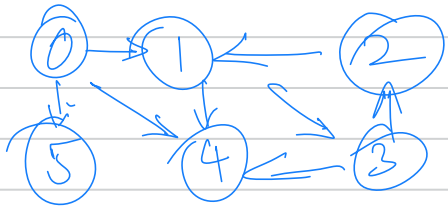


0 → 1 → 3 → 2 → 4 → 5

Breadth first

Start at root

explore each neighbor before going on to children



0 → 1 → 4 → 5 → 3 → 2

Bit Manipulation

+ → Addition

− → Subtraction

* → multiplication

^ → EXOR

~ → NOT

& = AND

<< = Shift by

Signed Bit numbers

To find, −3

3 = 0 11

flip of 3 = 1 00

add 1 = 0 01

(1) for sign → 1101

1101

Get Bit: get i^{th} bit

$(num \& (1 << i)) != 0$

Set Bit: change i^{th} bit to 1

$num | (i << 1)$

Clear Bit:

clear i^{th} bit

int mask = ~(1 << i)

num & mask

leave last i bits

int mask = (1 << i) - 1

num & Mask

Seq of 1's followed by i zeros.

int mask = (-1 << (i+1))

num & mask

Update bit: change i^{th} bit to value v

int value = bitIs1 ? 1 : 0;

int mask = ~(1 << i);

(num & mask) | (value << i)