Recap : Trees.

Linked List :

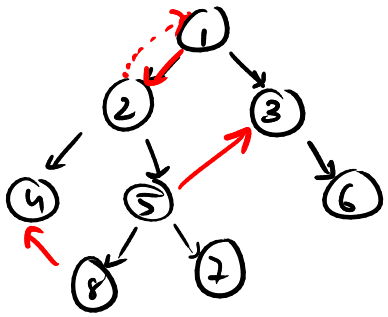
```
class Node {
    int data;
    Node next;
}
```

Linear structure.

Trees

```
class Node {
    int data
    Node left, right;
}
```

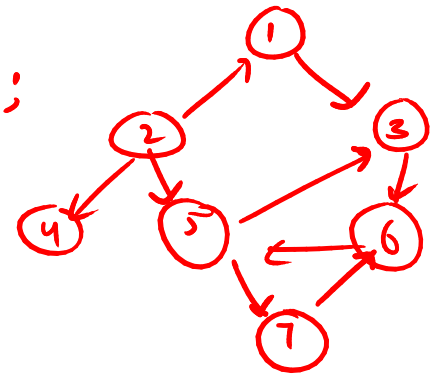
non-linear ds.



More generic

```
class Node {
    int data
    Node neighbors[];
}
```

Graph.



Graph = Nodes + Edges.

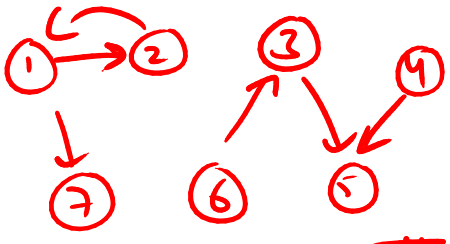
CS Math → vertices.

Terminology : nodes edges.

Eg. 'Google'

Why graphs : 'Making of a graph' ★

Graph. N nodes.

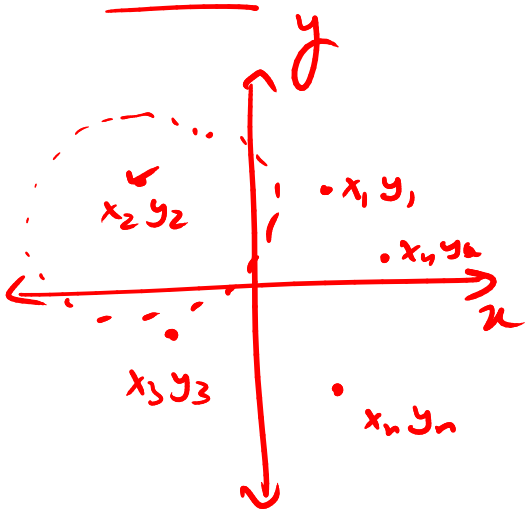


DFS → Components

N bombs r

x_i y_i r_i

chain reaction

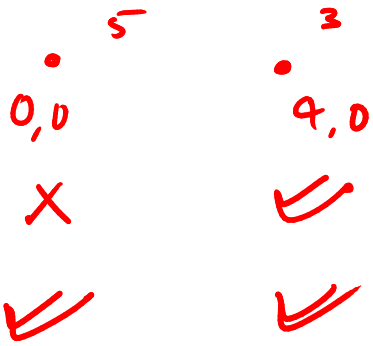


min no. of bombs that I detonate all bombs are detonated.

Graphs are everywhere !

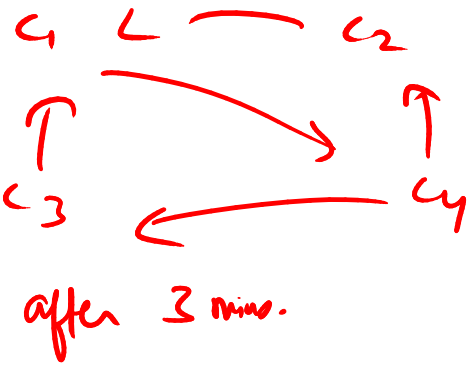
social media
transportation network
} static

dynamic graphs.



skyscanner

N cities



m-indicator (college startup)

Types of graphs. (the way we structure our theory)

graph = edge + nodes.

(int) DB primary key/index data
uuid int_id

weights
- No weights (unweighted)
- weights (weighted)

directions
- unidirectional / directed
- bidirectional / undirected

Time

Type of graphs (4)
- unweighted bidirectional
- unweighted unidirectional
- weighted bidirectional
- weighted unidirectional

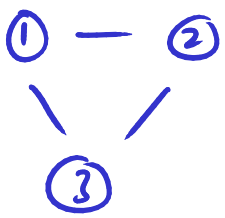
Bold claim.

5hrs
(1) BFS and DFS → unweighted. (shortest path, Connectivity) * 80%

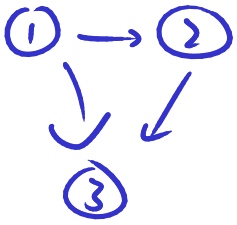
2hrs
(2) MST → weighted bidirectional. (trees and graphs) 15%. *

plus
(3) Dijkstra → BF $O(n^2)$ weighted (shortest path, Connectivity) } 5%
all source all dest^n. $O(n^3)$ Floyd Warshall

(4) Flows → weighted



undirected unweighted



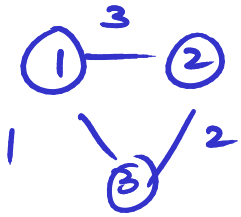
directed unweighted

How to represent ?

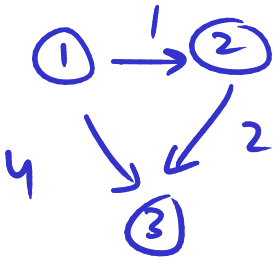
BT BST → tree

Heap → array

Graph → ?



undirected weighted



directed weighted.

Representation of Graph.

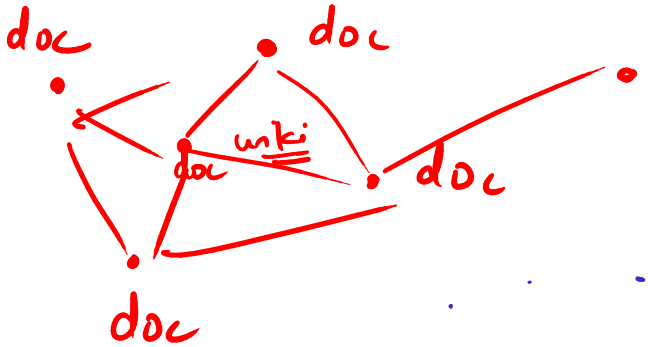
OOP:

```
class Node:
    int data
    Node[] neighbours
```

OR

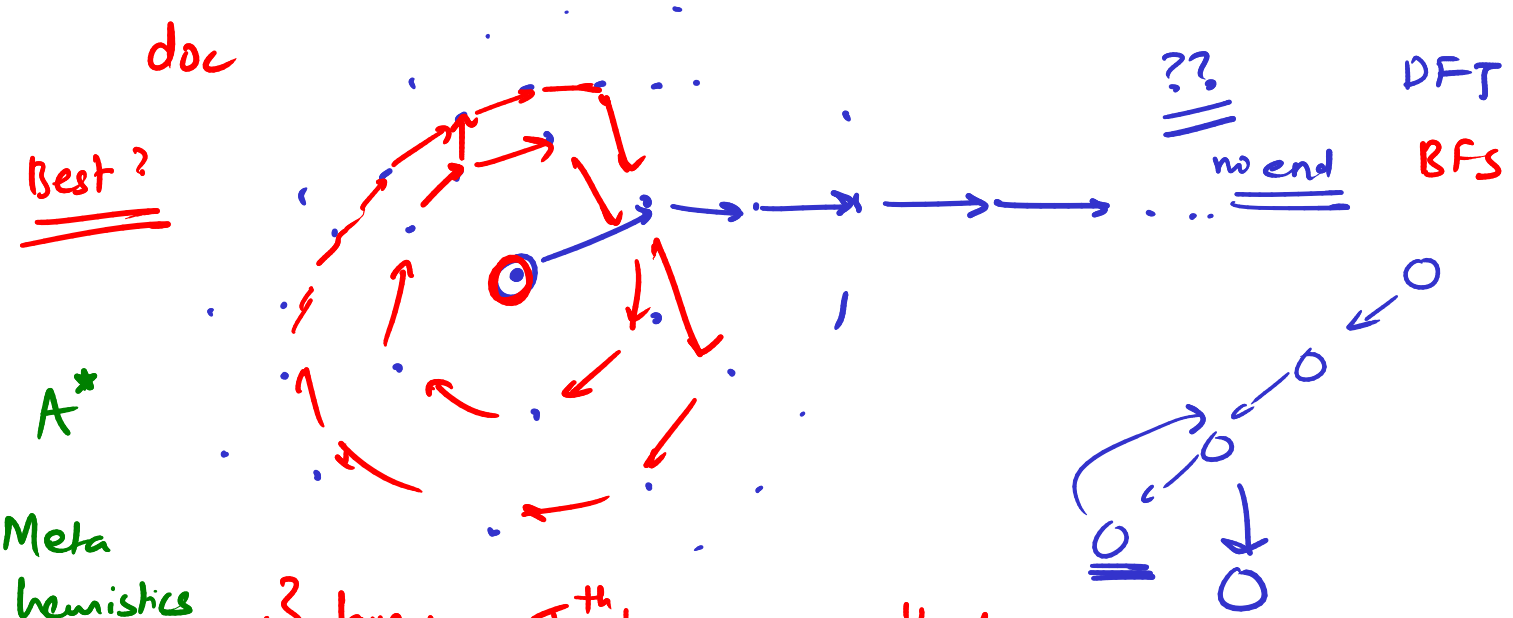
* vector<vector<int>>

Crawling on web → all data related to Maths.



infinite graph

Search ?



Best ?

A*

?? no end

DFT

BFS

Meta heuristics

3 layers

4th layer → can't do

dfs + BFS

60% BFs → deeper 40% DFs

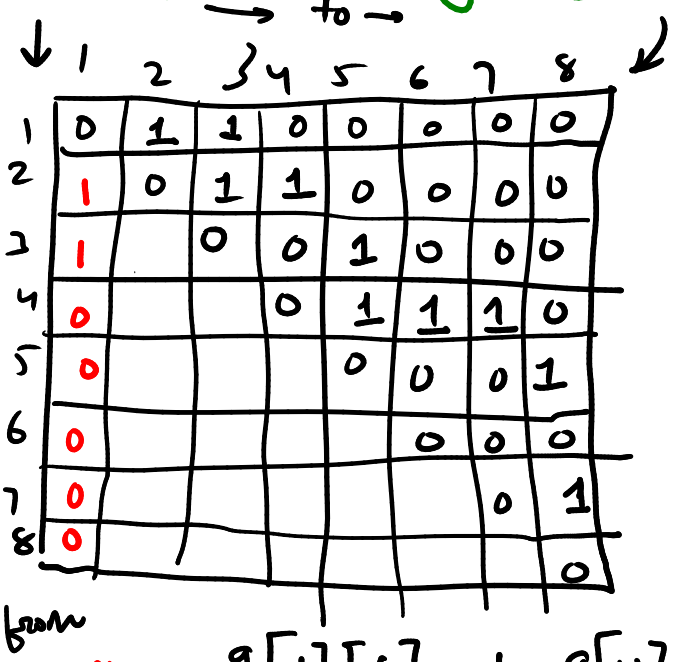
chess

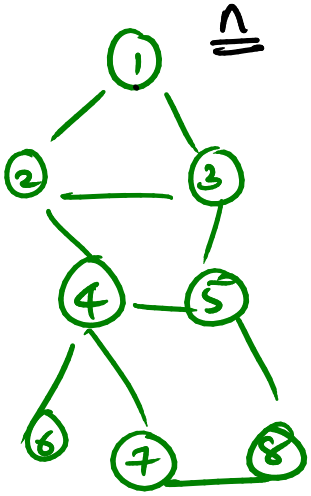
Position

middle game

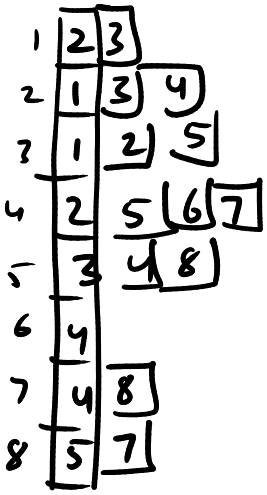
2 ways : Adjacency matrix

Adjacency list.





≡



fav of companies

O(e)

sym matrix

g[u][v] = 1 g[v][u] = 1

skip this

{ } → g[u].push_back(v)

g[v].push_back(u)

dense $O(n^2)$ space

majority → sparse

900 M insta users

not everyone follows everyone else.