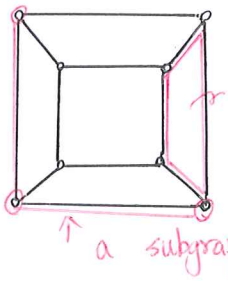
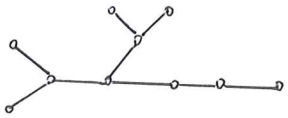
Graph theory : trees and searching algorithms



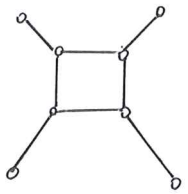
- A tree is a connected graph without cycle.

Property of a tree on n vertices :

- connected
- without cycle
- n-1 edges.



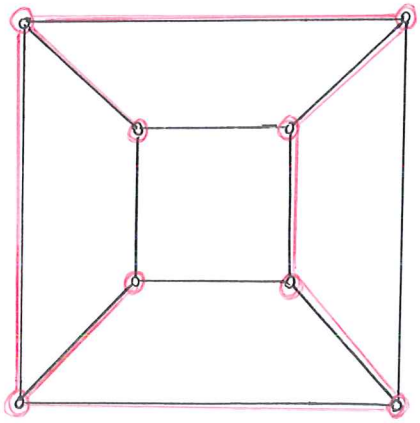
tree



not tree

- Any two properties out of the three defines a tree.
- A spanning tree of G is a subgraph of G using all vertices.
- A tree is the minimal connected structure.

in terms of number of edges



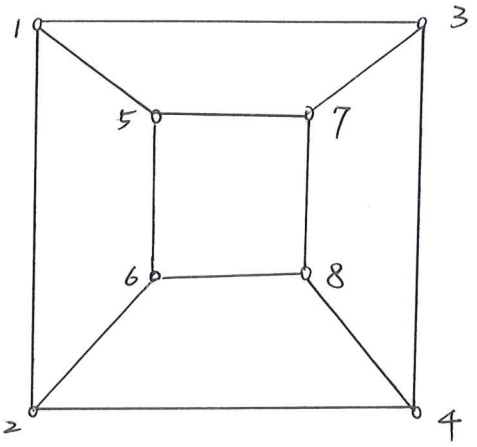
spanning tree

Breadth first search

Search each vertex of a graph such that the algorithm searches all neighbors first before move to other vertices.

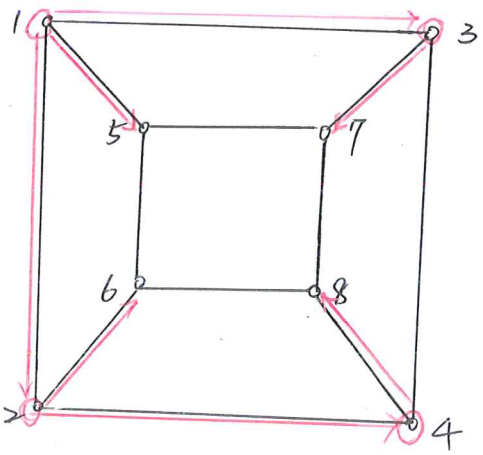
Exmp.

- Start from 1
  anchor at 1.
- Search 2, 3, 5,
  anchor at 2.
- Search 4, 6
  anchor at 3.
- Search 7,
  anchor at 5, 4
- Search 8,
  anchor at 6, 7, 8.



Applications

- Search through vertices.
- Test connected or not.
- Generate a spanning tree. (ditree)



Depth first search

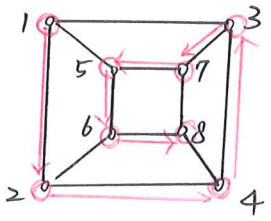
Search each vertex of a graph such that the algorithm searches as deep as possible.

Exmp

- Start from 1

→ 2 → 4 → 3

→ 7 → 5 → 6 → 8



Done!

Similar applications as BFS, same time complexity.

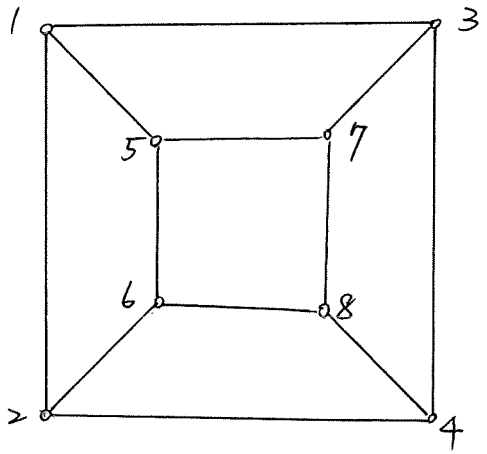
Implementing the BFS

Pseudocode:

```
Input:
    g: graph , v: vertex
Output:
    an order by BFS.
searched = [v]
waiting = [v]
while waiting not empty:
    anchor = first element in waiting
             (and remove anchor from waiting)
    new = [nbrs of anchor that is not yet searched]
    search + = new
    waiting + = new
```

Exmp (V=1)

Step	searched	waiting	anchor	new
0	[1]	[1]		
1	[1,2,3,5]	[2,3,5]	1	[2,3,5]
2	[1,2,3,5,4,6]	[3,5,4,6]	2	[4,6]
3	[1,2,3,5,4,6,7]	[5,4,6,7]	3	[7]
4	same	[4,6,7]	5	[]
5	[1,2,3,5,4,6,7,8]	[6,7,8]	4	[8]
6	done	[7,8]	6	[]
7	done	[8]	7	[]
8	done	[]	8	[]



G