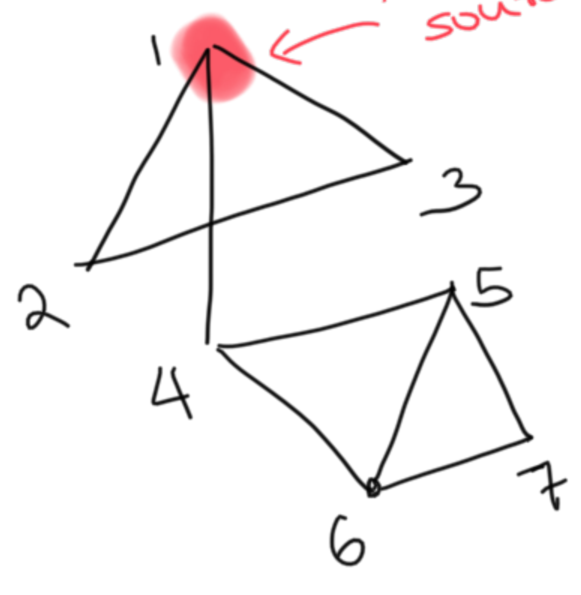
BFS

Example input

given source vertex



Graph G

"vertex i : [List of vertices that i is connected to]"

Adjacency list

1 : [2, 3, 4]
2 : [3]
3 : [2]
4 : [1, 5, 6]
5 : [4, 6, 7]
6 : [4, 5, 7]
7 : [5, 6]

Algorithm

0) Visited :

1	2	...	7
F	F		F

queue of to be explored but visited vertices : $\emptyset$

} Initially

I) → Visit source vertex.
→ Add it to queue to be explored

II) → For vertex v at head of queue, explore (v)

III) → Repeat step II till queue empty.

Exploring (v)

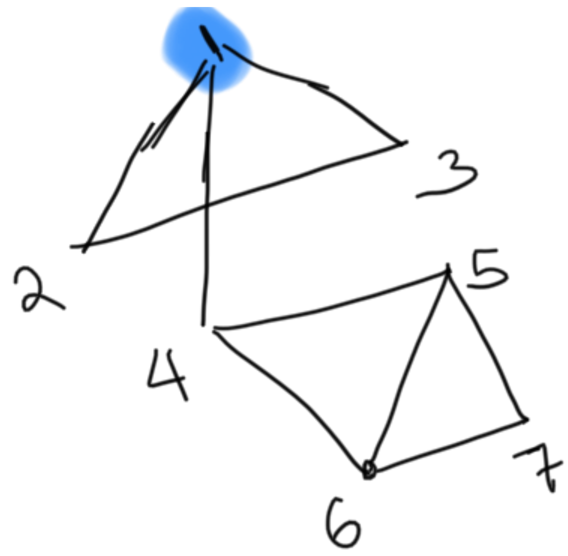
* Remove v from queue
* For each neighbour of v, if unvisited, visit it and add it to queue.

Dry run

Visited

1	2	3	4	5	6	7
F	F	F	F	F	F	F

Queue

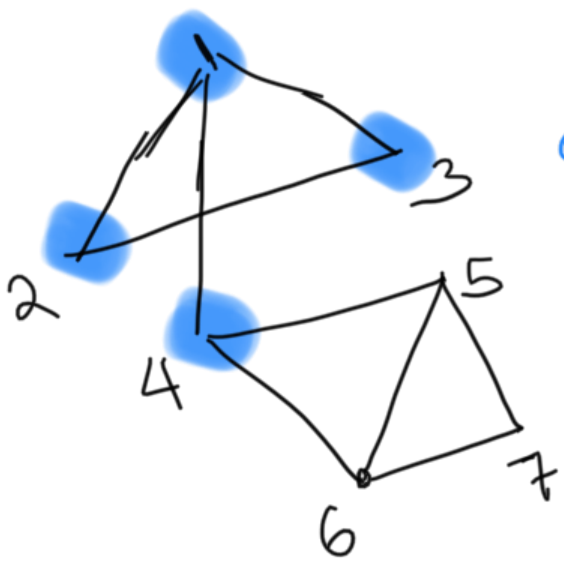


Visited

1	2	3	4	5	6	7
T	F	F	F	F	F	F

Queue

1



exploring 1

Visited

1	2	3	4	5	6	7
T	T	T	T	F	F	F

Queue

2	3	4

every vertex 2 is connected to already visited

exploring 2

Visited

1	2	3	4	5	6	7
T	T	T	T	F	F	F

Queue

3	4

every vertex 3 is connected to already visited

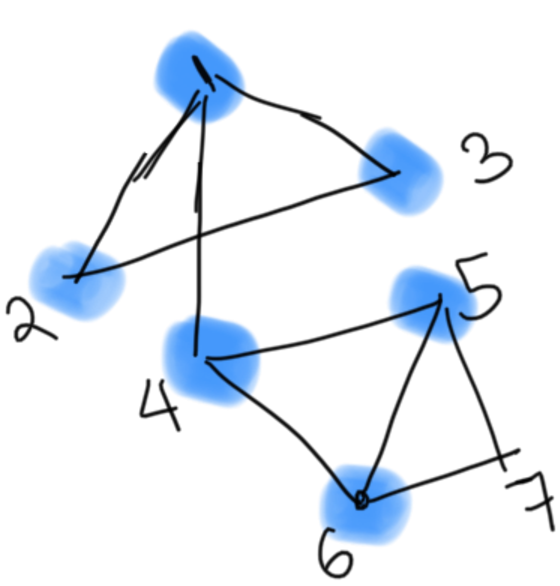
exploring 3

Visited

1	2	3	4	5	6	7
T	T	T	T	F	F	F

Queue

4



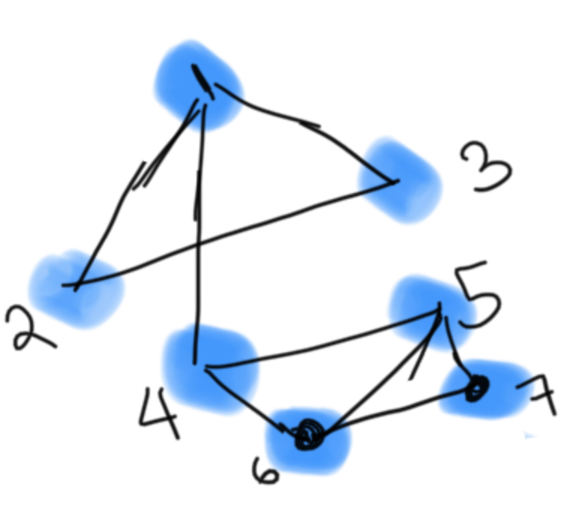
exploring 4

Visited

1	2	3	4	5	6	7
T	T	T	T	T	T	F

Queue

5	6



exploring 5

Visited

1	2	3	4	5	6	7
T	T	T	T	T	T	T

Queue

6	7

every vertex 6 is connected to already visited

exploring 6 ....

Visited

1	2	3	4	5	6	7
T	T	T	T	T	T	T

Queue

7

every vertex 7 is connected to already visited

exploring 7 ....

Visited

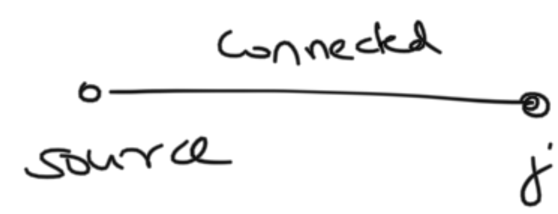
1	2	3	4	5	6	7
T	T	T	T	T	T	T

Queue $\emptyset$

Queue $\emptyset$, so stop

---

* if visited [j] = T, then



* if visited [j] = F, then

Complexity

- For each v in queue,

  → exploration takes O(degree(v)) time ↑ *

  because using adjacency list

- Each v enters queue exactly once

$$\sum_{v \in V} O(\text{degree}(v)) = O(|E|)$$

- Each v visited .... ← so $O(1) \times |V| = O(|V|)$

So over all $O(|V| + |E|)$

*: if using adjacency matrix, have to scan all vertices to see what v is connected to, so $O(|V|)$ to explore each v

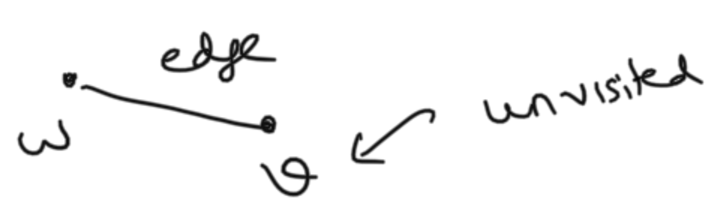
⤳ so over all $O(|V|^2 + |V|)$

$= O(|V|^2)$

Finding path from source to j

⤳ when you visit vertex v, remember the "parent" from which you marked it.

⤳ (ie) if exploring vertex (w),



So you visit v first time from w.

Set parent $(v) = w$

⤳ to find path

$j \to$ parent$(j) \to$ parent(parent(j)) $\ldots \to$ source

So have an initial "parent" array, all entries = None

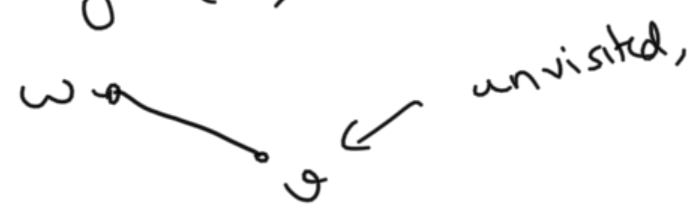
Finding "level" of vertex v (with respect to source)

* can find "shortest path from source to v"
  [each edge : weight 1]

* keep initial level array (all entries = None)

* level [source] = 0

* if exploring (w) :



you visit v from w first time,

level [v] = 1 + level [w]

So level [v] = length of shortest path from source to v