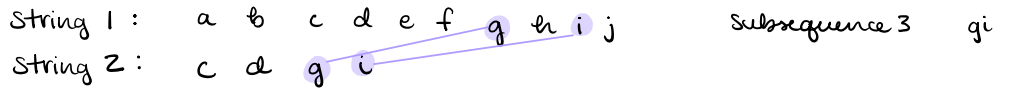
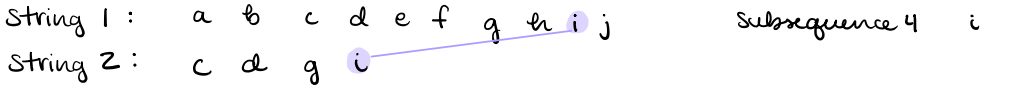
Longest Common Subsequence (LCS)

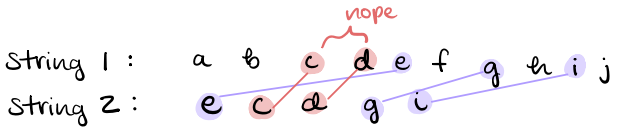
String 1 : a b c f g h i j
String 2 : c d g i

Some characters common between string 1 and string 2.
which set of chars already in string 2 **IN SEQUENCE** are also in string 1?
↳ select longest common subsequence





What if we added e?
limitation: **NO INTERSECTION**, cannot go across ✗



SO egi LCS

PRACTICE:

String 1 : a b d a c e nope
String 2 : b a b c e

String 1 : a b d a c e ✓
String 2 : b a b c e

OR

String 1 : a b d a c e ✓
String 2 : b a b c e

LCS = 4 but there are different subsequences

Tabulation Method

String 1 : l o n g e s t
String 2 : s t o n e

	0	1	2	3	4	5	6
A	l	o	n	g	e	s	t
B	s	t	o	n	e		
	0	1	2	3	4		

		l	o	n	g	e	s	t
	0	0	0	0	0	0	0	0
s	1	0	0	0	0	0	1	1
t	2	0	0	0	0	0	1	2
o	3	0	0	1	1	1	1	2
n	4	0	0	1	2	2	2	2
e	5	0	0	1	2	2	3	3

match = max of diagonal + 1
otherwise = max of diagonal
↳ previous column and row

		l	o	n	g	e	s	t
	0	0	0	0	0	0	0	0
s	1	0	0	0	0	0	1	1
t	2	0	0	0	0	0	1	2
o	3	0	0	1	1	1	1	2
n	4	0	0	1	2	2	2	2
e	5	0	0	1	2	2	3	3

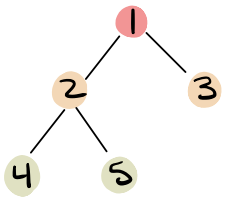
← if same # to left
↖ otherwise
● = included in LCS

LCS = one

Breadth First Search

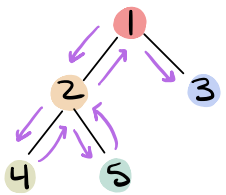
BFS & DFS: graph traversal search
- discover the vertex (visit)
- explore the vertex
- select any vertex as source

BFS:



Queue
1, 2, 3, 4, 5

DFS:



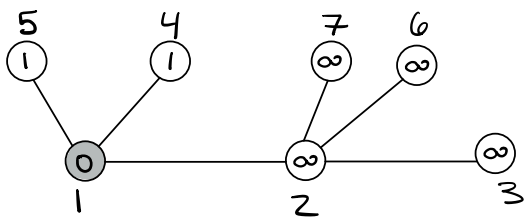
Stack
1, 2, 4, 5, 3

Color [v]
- white (vertex has not been discovered)
- Grey (vertex has been discovered & is being processed)
- Black (has been completely processed)

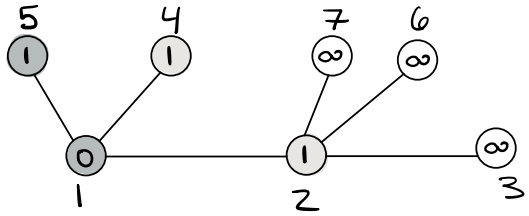
```
BFS(V, E, S)        ← vertex, edge, source
  for each vertex v in V - {S}   ← exclude Source
      do
        color[v] ← white   ← initially all vertices are white
          d[v] ← ∞         ← d[v]: distance of vertex v from Source
        π[v] ← NULL        ← π[v]: parent of vertex v
      color[S] = Grey
          d[S] ← 0         ← distance of source from source = 0
        π[S] ← NULL
          Q ← { }
      ENQUEUE(Q, S)        ← Queue Source now
      while Q is not empty
            do v ← DEQUEUE(Q)
            for each u adjacent to v
            do if color[u] ← White   ← all white adjacent nodes to S
                then color[u] ← Grey   ← change color to Grey
                 d[u] ← d[v] + 1       ← increase d[v] by one
               π[u] ← v
               ENQUEUE(Q, u)   ← add the now Grey dude to queue
               color[v] = Black   ← after dequeue = black
```

1. all white adjacent nodes to S change color to grey
2. increase d[v] by one
3. add now grey nodes to the queue
4. change color of dequeued nodes to black

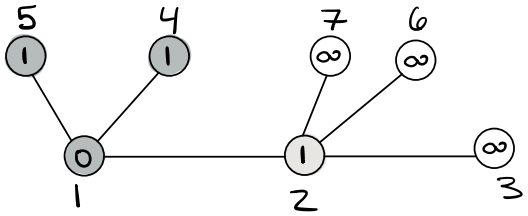
1, 5, 4, 2, 7, 6, 3



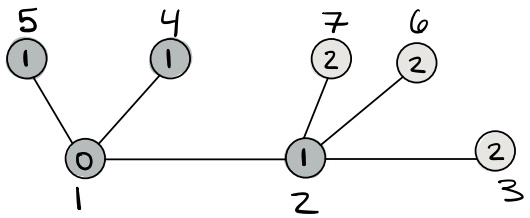
~~1~~ 5 4



~~5~~ 4



~~4~~



~~2~~ 7 6 3