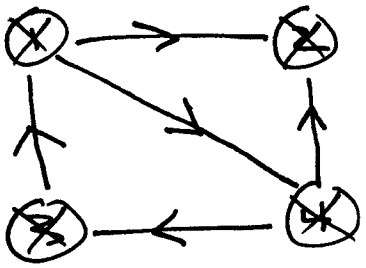
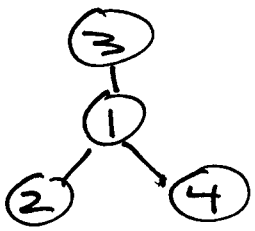
CSE 101 1-19-21

- Pa2 : ext. 2 days
- Pa3 : Posted tonight.

Ex.  run BFS with $s = 3$

	adj	color	dist	Parent
1	2 4	black	1	3
2		black	2	1
3	1	black	0	nil
4	2 3	black	2	1

Q: ~~3~~ X ~~2~~ ~~4~~

Tree 

dist

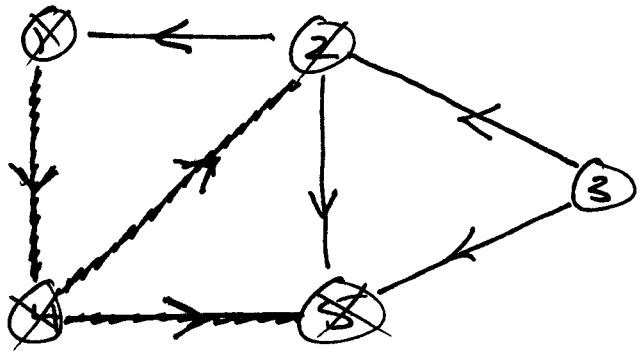
0

1

2

Ex.

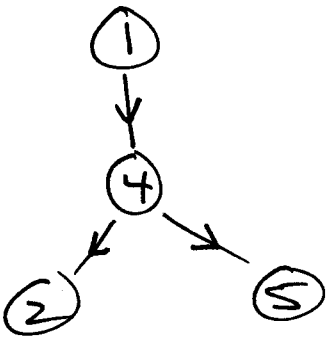
s = 1



	adj	color	dist	Parent
1	4	b	0	nil
2	1 5	b	2	4
3	2 5	w	∞	nil
4	2 5	b	1	1
5		b	2	4

Q: 1 4 2 5

Tree



dist

0

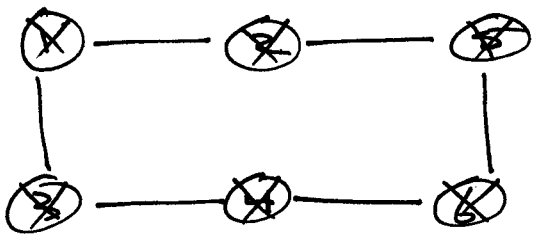
1

2

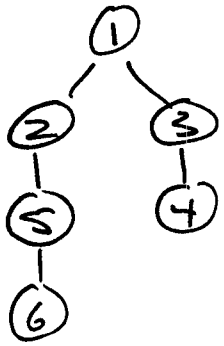
PrintPath(G, s, x) prints a shortest s-x path in G.

Precondition: BFS(G, s) must be run first.

Ex.



Tree

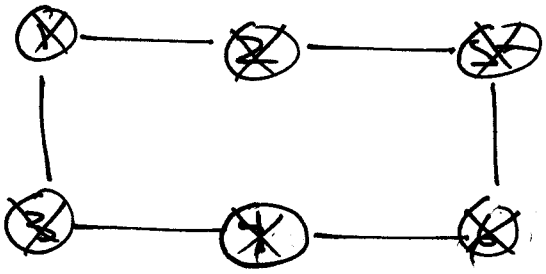


s = 1 Q: 1 2 3 5 4 6

Shortest 1-6 Path:

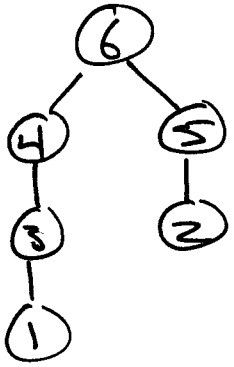
1, 2, 5, 6

Ex.



s = 6 Q: 6 4 5 3 2 1

Tree



shortest 6-1 Path:

6, 4, 3, 1

Exercise(s)

- Run BFS on many examples.
- defn The eccentricity of $x \in V(G)$, denoted $e(x)$ is the max. poss. distance from x.

$$e(x) = \max_{y \in V} d(x, y)$$

use Graph ADT ops to write a client fcn. that computes $e(x)$, for any x.

- defn The radius of G is the minimum eccentricity, denoted $rad(G)$. The central vertices in G are those for which

$$e(x) = rad(G).$$

using Graph ADT, write a client fcn that computes (and returns) $rad(G)$, and appends to L (2nd arg) the central vertices.

note

$$rad(G) = \min_{x \in V} e(x)$$

$$= \min_{x \in V} \left(\max_{y \in V} \delta(x, y) \right)$$

- defn The diameter of G is the maximum eccentricity, denoted $\text{diam}(G)$.

$$\text{diam}(G) = \max_{x \in V} e(x)$$

$$= \max_{x \in V} \left(\max_{y \in V} f(x,y) \right)$$

The peripheral vertices are those $x \in V$ for which

$$e(x) = \text{diam}(G).$$

Write a client fcn. that finds $\text{diam}(G)$, and the set of peripheral vertices.

Recall: a graph H is a subgraph of a graph G iff

$$V(H) \subseteq V(G)$$

and $E(H) \subseteq E(G)$

Recall (G an undirected graph) G is called connected iff for all $x, y \in V(G)$: x is reachable from y (and y from x).

Defn

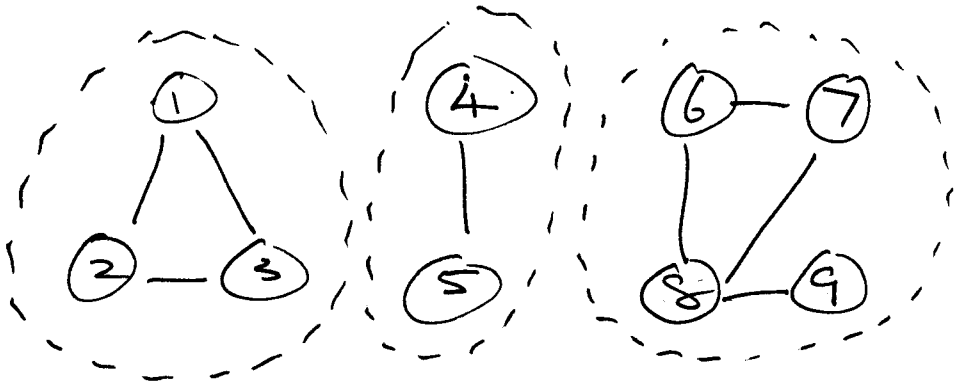
A connected component in G is a subgraph H s.t.

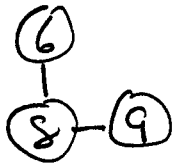
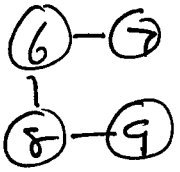
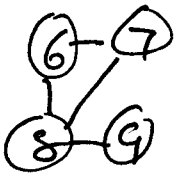
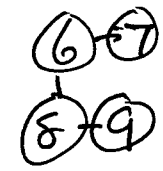
(1) H is connected.

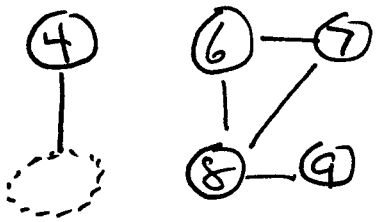
and

(2) H is maximal w.r.t. (1)

Ex.



- $(\{6,8,9\},\{68,89\})$  conn. not maximal
- $(\{6,8,9,7\},\{67,68,89\})$  conn. not maximal
- $(\{6,8,9,7\},\{67,68,89,78\})$  conn. maximal.
- $(\{6,8,9,7,4\},\{67,68,89,78\})$  not conn.
- $(\{6,8,9,4,7\},\{67,68,89,78,45\})$



not a graph!

Exercise

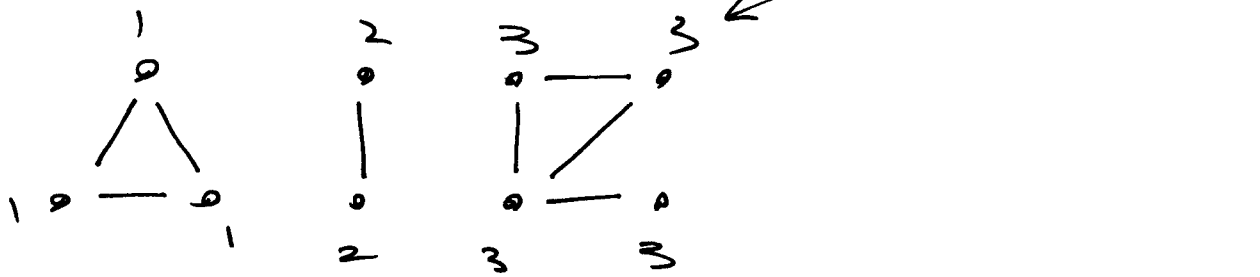
Add one new vertex attribute called $cc[x]$ for $x \in V(G)$.

Now write a client fcn for Graph ADT that assigns value $cc[x]$ each $x \in V(G)$, such that

$$cc[x] = cc[y]$$

iff x, y lie in same Conn. Comp.

Ex.



Depth First Search (DFS)

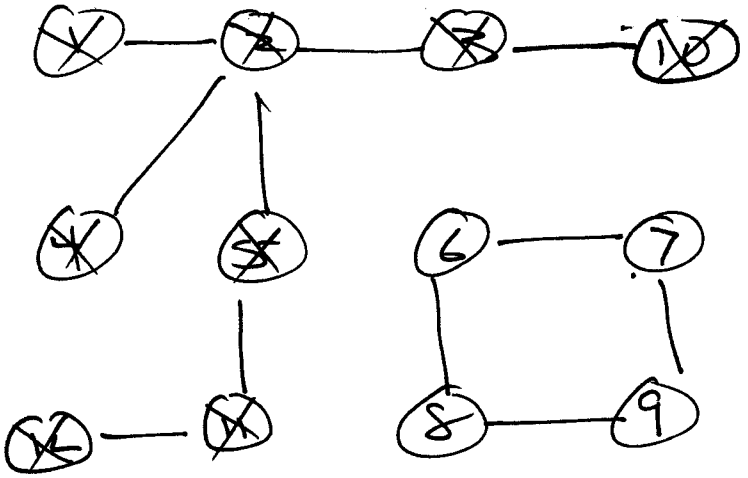
Vertex attributes

- color[x] : white, gray, black
- P[x] : Parent or Predecessor
- d[x] : discover time
- f[x] : finish time

uses a recursive subroutine called Visit(x)

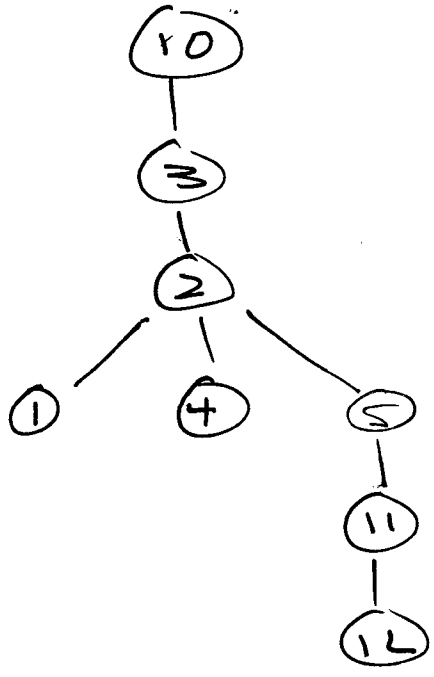
uses variable time (in DFS), which is static over all recursive calls to Visit( ).

Ex



Q: 10 3 2 1 4 5 11 12

Tree



dist

6

1

2

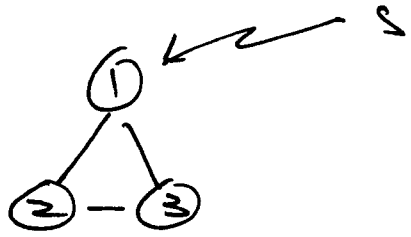
3

4

5

Graph ADT

client view



inside view