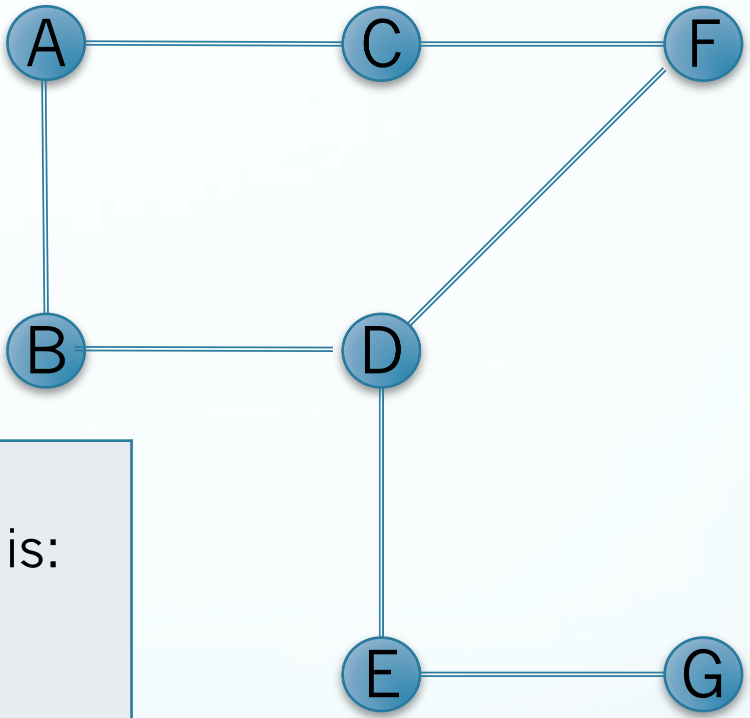
COMP20007 Workshop Week 5

0	**Preparation:** - (optional) open `wokshop6.pdf` (from `LMS`)
1	**Topic 1:** DFS and BFS: Problems 3, 5
2	**Topic 2:** Dijkstra's and Prim's Algorithms; Problem 7 ***Group Work:*** Problems 8, 4, 6, 7 (preferably in that order)
LAB	**Q&A on Practice MST** **Assignment 1 Q&A**, and -make sure that you can scp, ssh, use dimefox to test/submit -make sure you understand how to start the programming part -do Assignment 1 or review for MST & ask questions

DFS & BFS

Problem 4: Depth First Search

- List the order of the nodes visited by the a) DFS and b) BFS algorithms



YOUR ANSWER:

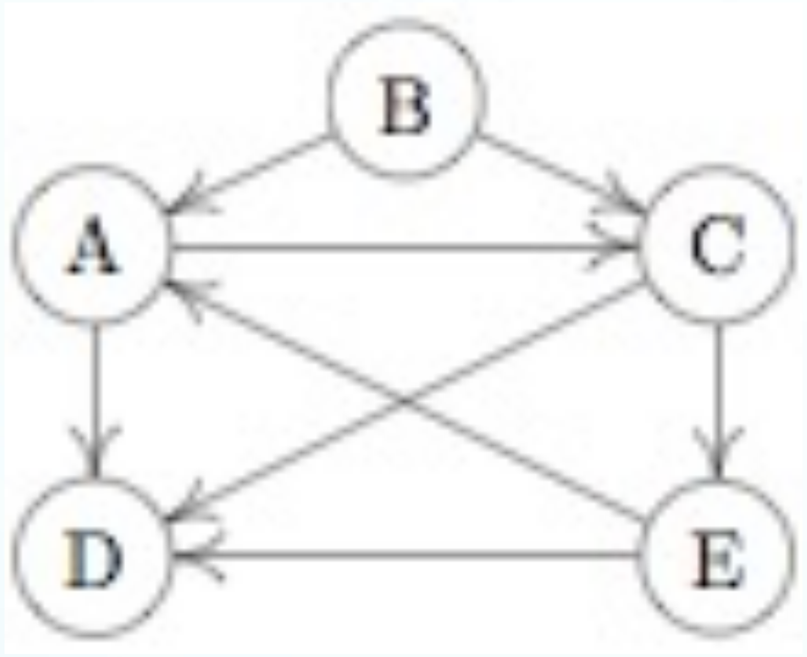
a) The order of the nodes visited by DFS is:

A

b) The order of the nodes visited by BFS is:

A

Problem 5: Tree, Back, Forward and Cross Edges



A DFS of a di-graph can be represented as a collection of trees. Each edge of the graph can then be classified as a *tree edge*, a *back edge*, a *forward edge*, or a *cross edge*. A tree edge is an edge to a previously un-visited node, a back edge is an edge from a node to an ancestor, a forward edge is an edge to a non-child descendent and a cross edge is an edge to a node in a different sub-tree (i.e., neither a descendent nor an ancestor)

Draw a DFS tree based on the following graph, and classify its edges into these categories.

In an undirected graph, you wont find any forward edges or cross edges. Why is this true? You might like to consider the graph above, with each of its edges replaced by undirected edges.

A gift from Anh: If you are a bit bored or tired, skip this exercise, and instead use pages 23-35 (of this file) to entertain yourselves ;-). Remember to do it or to have a look at the solution later!

Problem 6: Finding Cycles

a) Explain how one can also use BFS to see whether an undirected graph is cyclic.
b) Which of the two traversals, DFS and BFS, will be able to find cycles faster? (If there is no clear winner, give an example for proof).
Note: skip part b) if it takes you more than 2 minutes, you can do it later!

YOUR BRIEF ANSWER:

BFS algorithm from lecture

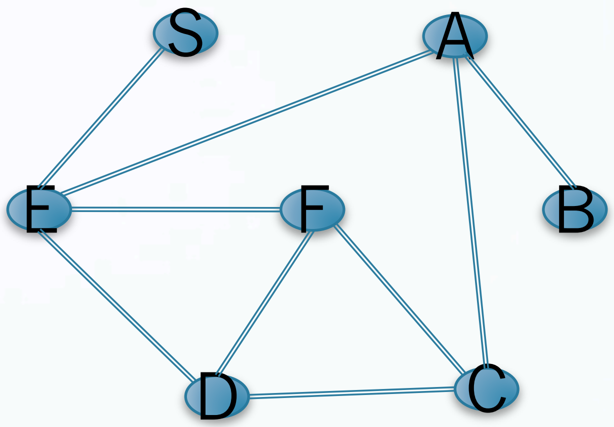
```
function BFS(⟨V, E⟩)
    mark each node in V with 0
    count ← 0, init(queue)
    for each v in V do
        if v is marked 0 then
            BFSEXPLORE(v)

function BFSEXPLORE(v)
    count ← count + 1
    mark v with count
    inject(queue, v)                        ▷ queue
    while queue is non-empty do
        u ← eject(queue)
        for each edge (u, w) adjacent to u do
            if w is marked with 0 then
                count ← count + 1
                mark w with count
                inject(queue, w)
```

Problem 7: 2-Colourability

Design an algorithm to check whether an undirected graph is 2-colourable, that is, whether its nodes can be coloured with just 2 colours in such a way that no edge connects two nodes of the same colour.

To get a feel for the problem, try to 2-colour the following graph (start from **S**).

Do you expect we could extend such an algorithm to check if a graph is 3-Colourable, or in general: k-Colourable?



YOUR BRIEF ANSWER:

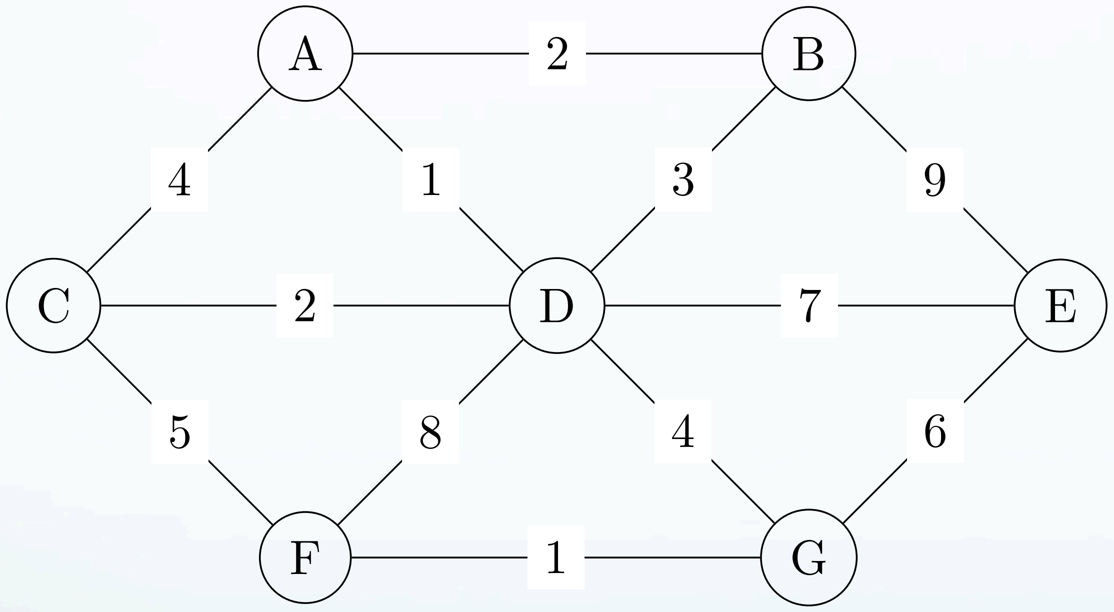
a) try to 2-colour the above graph, starting from **S**, using 2 colours 1 and 2
   hint: what is colour for S? how do we continue?

b) So, how to solve the 2-colourability? The pseudocode? What's the time complexity?

c) Do you expect we could extend such an algorithm to check if a graph is 3-Colourable, or in general: k-Colourable?

Prim's & Dijkstra's algorithm

- Purpose? How?



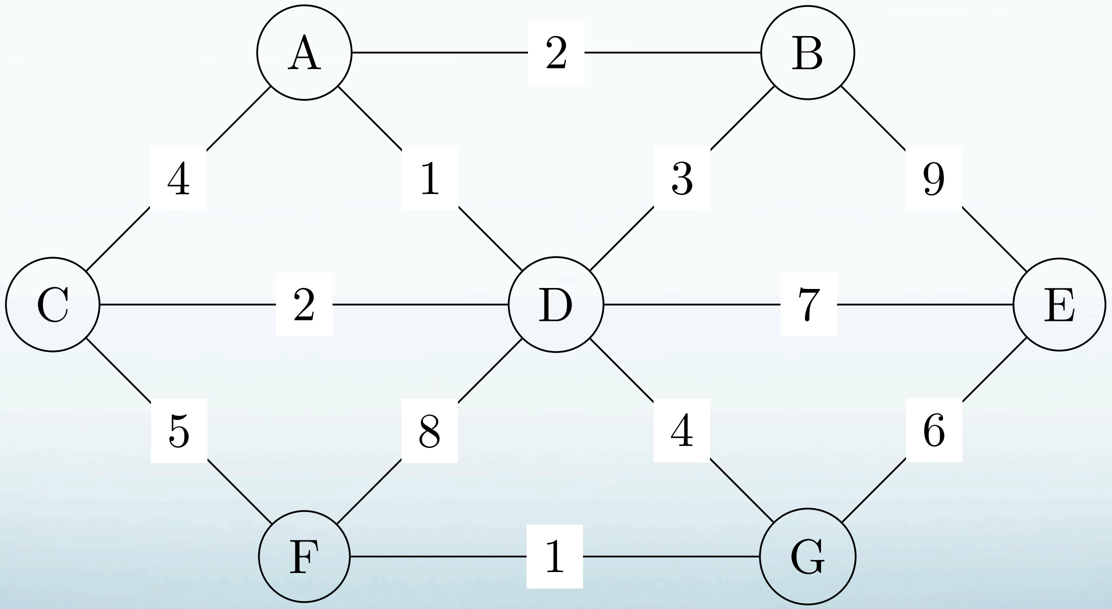
Dijkstra’s and Prim’s are similar?

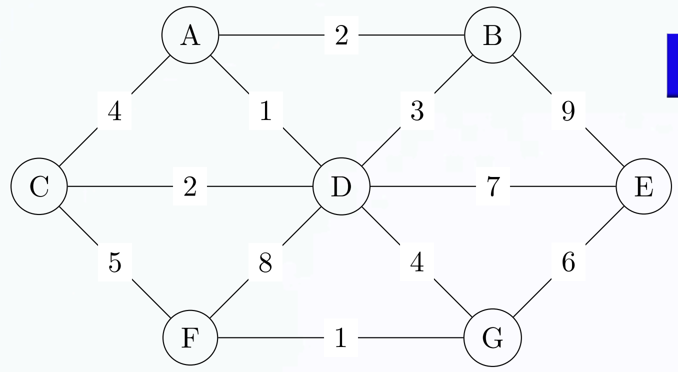
Dijkstra(G,S)	Prim(G)
SSSP (that involves all nodes of a *connected* graph)	MST (that involves all nodes of a *connected* graph)
for each v ∈V do	for each v ∈V do
dist[v]:= ∞	dist[v]:= ∞
prev[v]:= nil	prev[v]:= nil
	pick initial s
dist[S]= 0	dist[S]:=0
PQ= create_priority_queue(V)	PQ= create_priority_queue(V)
while (PQ is not empty) do	while (PQ is not empty) do
u := eject node with minimal	u := eject node with minimal
dist[u] from PQ	dist[u] from PQ
for each neighbour v of u do	for each neighbour v of u do
if **dist[u]+w(u,v)** < dist[v]then	if **w(u,v)** < dist[v] then
dist[v]:= **dist[u]+w(u,v)**	dist[v]:= **w(u,v)**
prev[v]:= u	prev[v]:= u

Problem 9: SSSP with Dijkstra’s Algorithm (DA)

Dijkstra’s algorithm computes the shortest path to each node in a graph from a single starting node (the ‘source’). Trace Dijkstra’s algorithm on the following graph, with node E as the source

Repeat the algorithm with node A as the source. How long is the shortest path from E to A? How about A to F





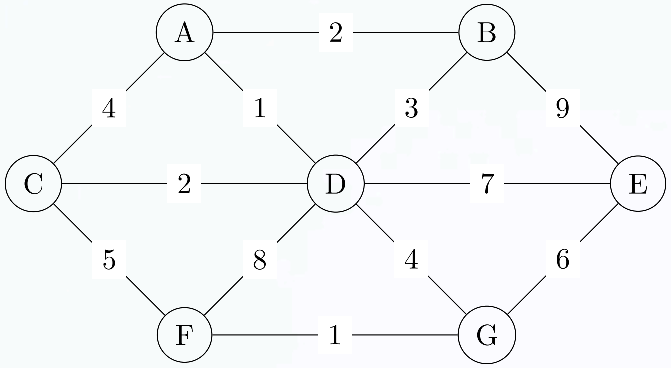
Dijkstra's Algorithm from E

dist[E]

prev[E]

step	node ejecte d	A	B	C	D	E	F	G
0		∞/nil	∞/nil	∞/nil	∞/nil	**0**/nil	∞/nil	∞/nil
	E	∞/nil	**9,E**	∞/nil	**7,E**		∞/nil	**6,E**
	G	∞/nil	**9,E**	∞/nil	**7,E**		**7,G**	
	D	**8,D**	**9,E**	**9,D**			**7,G**	
	F	**8,D**	**9,E**	**9,D**				
	A		**9,E**	**9,D**				
	B							

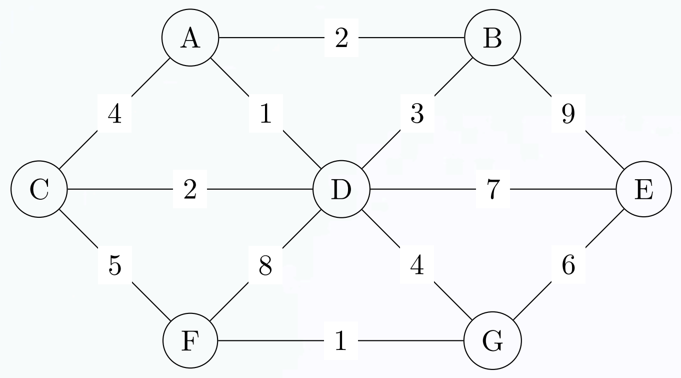
DA is a BFS. DA is used to find shortest path from a **source** to *all other*



DA from E

step	node visited	A	B	C	D	E	F	G
0		∞/nil	∞/nil	∞/nil	∞/nil	**0**/nil	∞/nil	∞/nil
1	E		**9,E**		**7,E**			**6,E**

At step 1: choose the node with minimal dist ***amongst the unvisited nodes***. This is E. Then, consider to update dist[v] for v= {B, D, G}. **And, mark E as visited ???**.

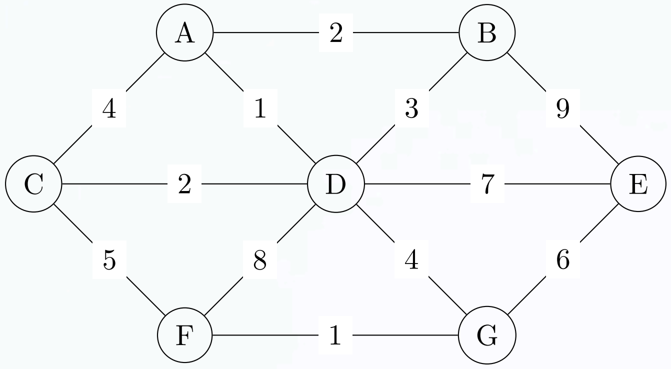


DA from E

How long, *and what is*, the shortest path from E to A?

A 🡐 D 🡐 E: path= E D A , cost= 8

step	node ejected	A	B	C	D	E	F	G
0		∞/nil	∞/nil	∞/nil	∞/nil	**0**/nil	∞/nil	∞/nil
1	E	~	9/E	~	7/E		~	**6**/E
2	G(6)		**9/E**		**7/E**		**7,G**	
3	D (7)	**8,D**	**9/E**	9,D			**7,G**	
4	F(7)	**8,D**	**9/E**	9.D				
5	A(8)		**9/E**	9/D				
6	B(9)			9/D				
7	D							



DA from A

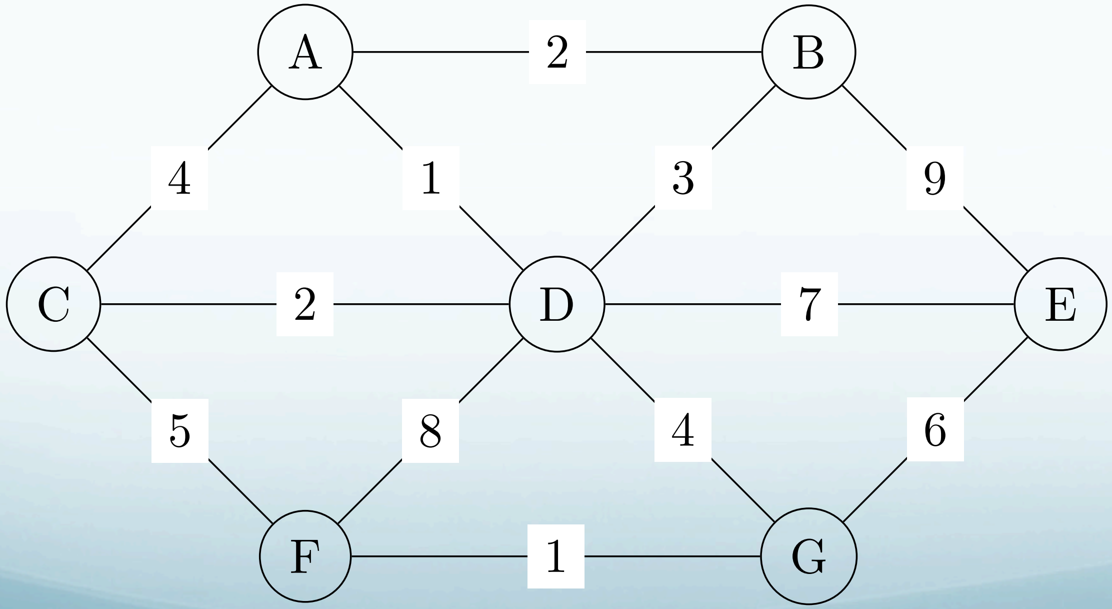
How long, *and what is*, the shortest path from E to A?
How about from A to F?

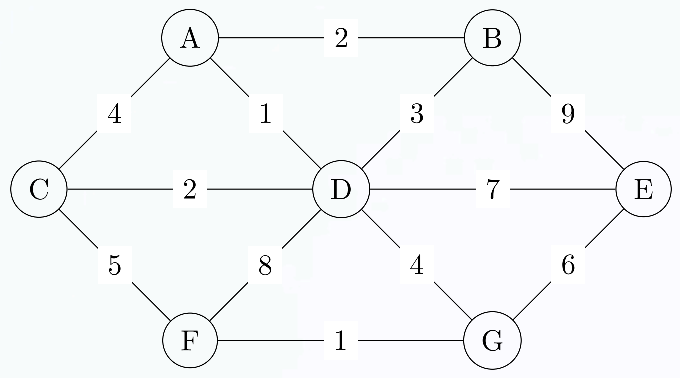
step	node visited	A	B	C	D	E	F	G
0		0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
1								
2								
3								
4								
5								
6								
7								

Problem 8: Minimum Spanning Tree with Prim's Algorithm

Prim's algorithm finds a minimum spanning tree for a weighted graph. Discuss what is meant by the terms 'tree', 'spanning tree', and 'minimum spanning tree'.

Run Prim's algorithm on the graph below, using A as the starting node. What is the resulting minimum spanning tree for this graph? What is the cost of this minimum spanning tree?





Prim's Alg from A

What's the resulting MST?
What's the cost of that MST?

step	node ejected	A	B	C	D	E	F	G
0		0/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil	∞/nil
1	A		2,A	4,A	1,A	-	-	-
2	D		**2,A**	2,D		**7,D**	**8,D**	**4,D**
3	B			2,D		**7,D**	**8,D**	**4,D**
4	C					**7,D**	**5,C**	**4,D**
5	G					**6,G**	**1,G**	
6	F					**6G**		
7	G							

Food for our brain

- You’re organizing your birthday party and inviting n friends. From these friends you know that there are some pairs of people who hates each other. You want to know if it’s possible to divide guests into 2 different tables so that in each table there is no such hating pair? Design an algorithm for that.
- You have an unweighted undirected graph. Design an algorithm to find the shortest part (the part with least number of edges) between 2 vertices. What if the graph is weighted?

assignment 1

- Do it early! Submit early, resubmit if needed!
- Read & participate in discussion forum!
- Make sure that you follow well the specification.
- Check your writing part carefully..
- Test your program carefully: remember to test on dimefox

- Make sure you don't have memory leak:
  - check that every execution of `malloc` matches with an execution of `free`, and
  - [optional] use tools like `valgrind` to test for memory leak.

github.com/anhvir/c207/Lab_Notes.pdf
(how to work with dimefox)

LAB:

- work on not-yet-done this week's workshop problems,

- prepare for MST: **see sample MST test from LMS**

- work on ass1

Lab: do assmt1 or your choice