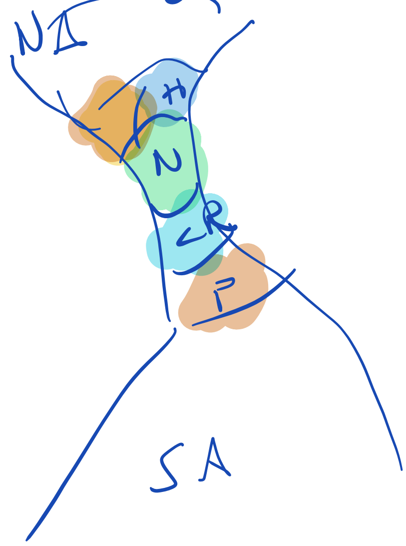
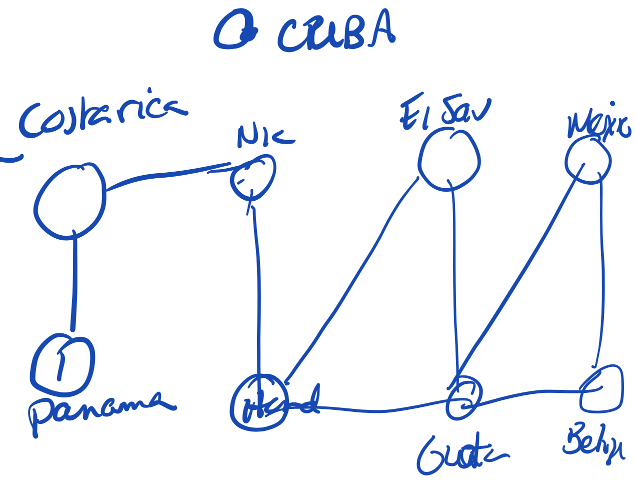
Hello again !!!

A cartographers problem





Color a graph s.t. neighboring countries have diff. colors.

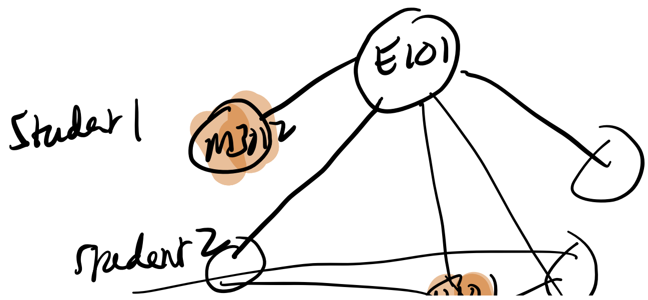
Exam scheduling

As few time slots as possible
Each student has to have exams at diff times

F 10am: Math 3032 CS 2050 H101

Nodes: exams
Edges: Some student is taking both classes
Color: Time slots

Coloring, again!



Graphs

$G = (V, E)$ where V = set of vertices

E : edges of (v, u)

V: students in class

E: pairs who know each other

Undirected graph: $(v,u) \in E \rightarrow (u,v) \in E$

symmetric relationship

Data Science:

- Social networks: Pairs who know each other
  "6 degrees of separation"
  HUGE
- Biological networks: Colonies of ants
  Cells in body
- Physical system: Vertices are particles interacting

Efficiency (running time) really matters!

How are graphs stored on a computer?

1. Adjacency matrix

$$\begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ 1 & 0 & 1 & 1 & 0 \end{array}$$

$$\begin{array}{c} 2 \\ 3 \\ 4 \end{array}\begin{bmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

① . . . . ④

PRO
check if an edge exists in $O(1)$ time

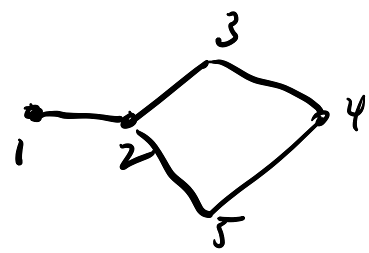
$|V| = n$ (n vertices)
$|E| = m$ (m edges)

Simple graph : no self loops $A(v,v) = 0$
no multiple edges
$A(v,u) \in \{0,1\}$

CON
always uses $O(|V|^2)$ space

Symmetric matrix if G is undirected
$A(v,u) = A(u,v) \quad \forall\, v,u \in V$

② Adjacency list



1 → 2
2 → 1 → 3 → 5
3 → 2 → 4
4 → 3 → 5
5 → 2 → 4

PRO
uses $|E|$ space (sparse graph) (few edges)

CON Can take $O(|V|)$ time to check for an edge.

PRO Easily go through each neighbor of a vertex

$n = |V|$, $m = |E|$ Simple graph

$0 < m \leq \binom{n}{2} = O(n^2)$ if $m = \binom{n}{2}$

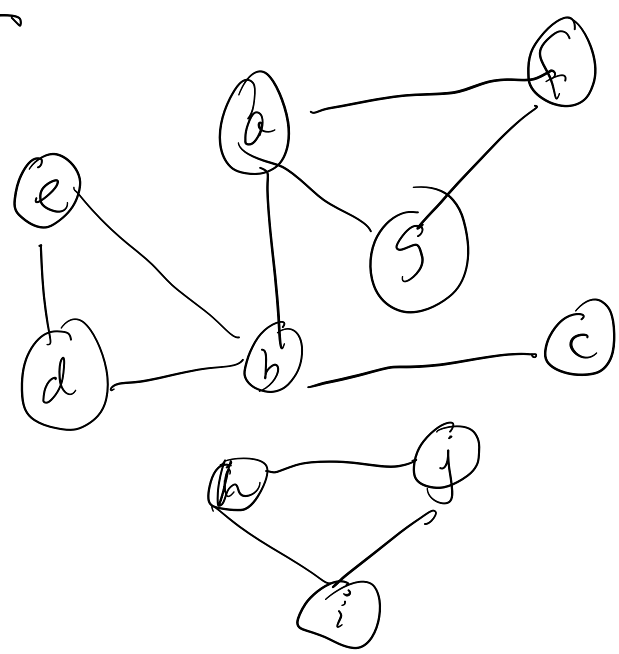
complete graph

Simple connected graph

$n-1 \leq m \leq \binom{n}{2}$

DFS

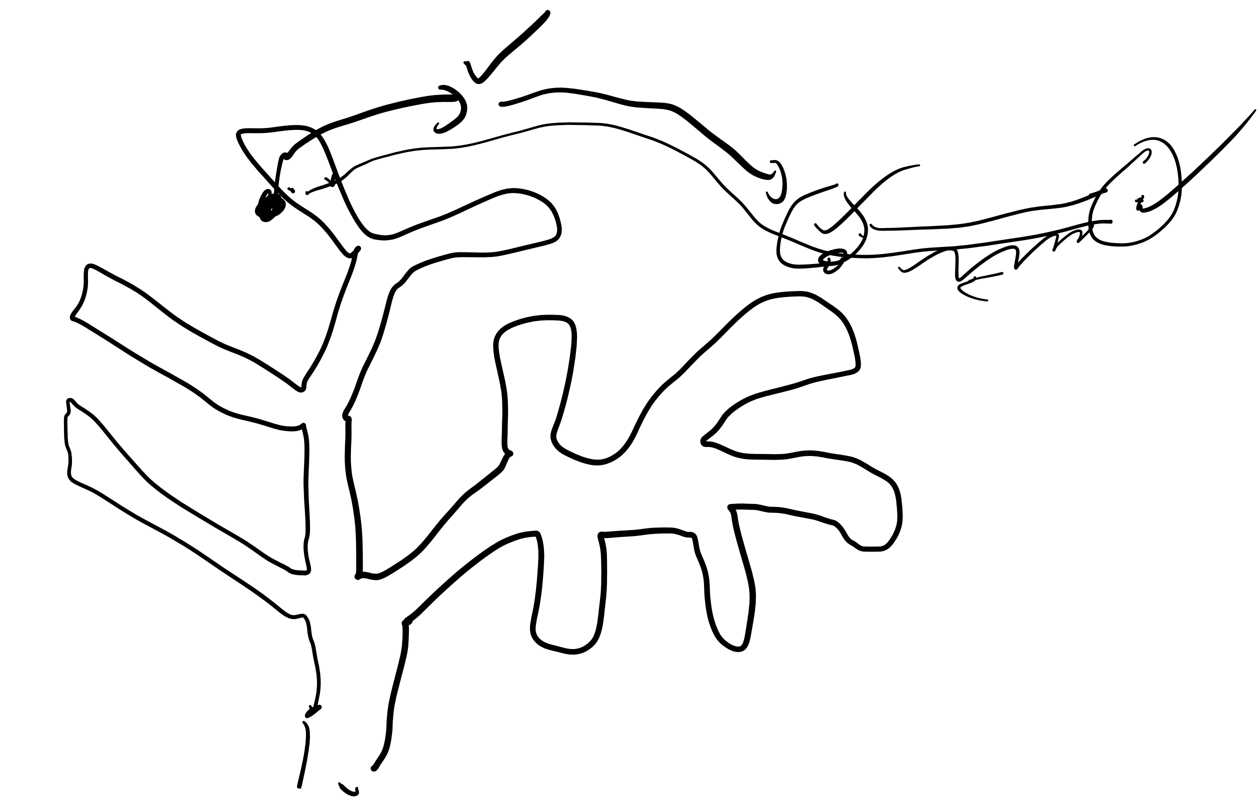
$G = (V, E)$:



What parts of the graph are

reachable from a particular vertex?

Classic solution: Navigating a maze



An exploration procedure

Explore (G, v)

input: Graph $G = (V, E)$; node $v \in V$

output: visited[u] is set to true $\forall$ u reachable from v

[~~initialize: visited[u] = false $\forall u \in V$~~]

visited[v] = true

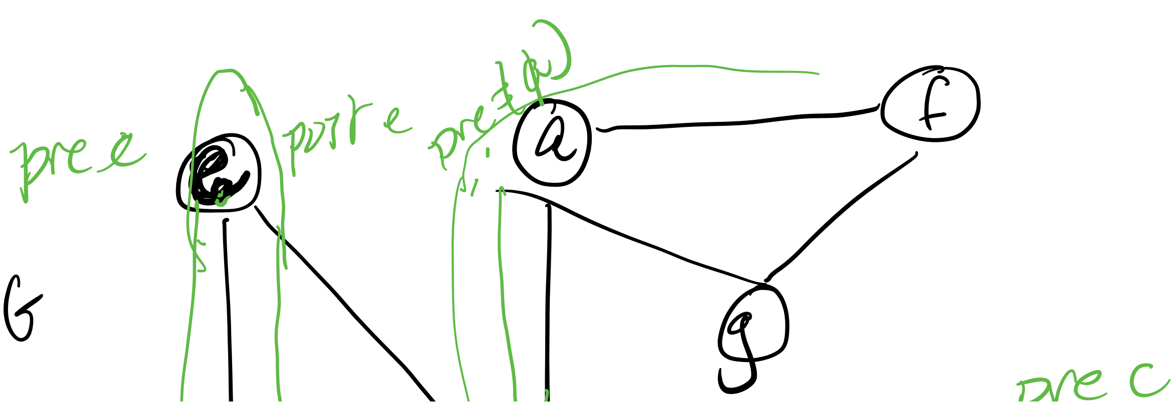
previsit(v)

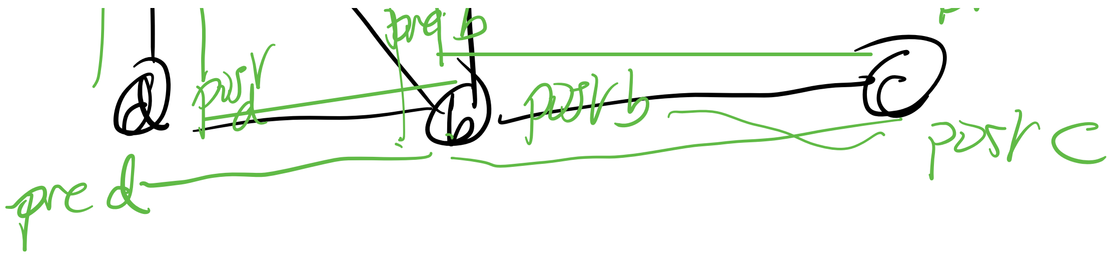
for each edge $(v, u) \in E$:

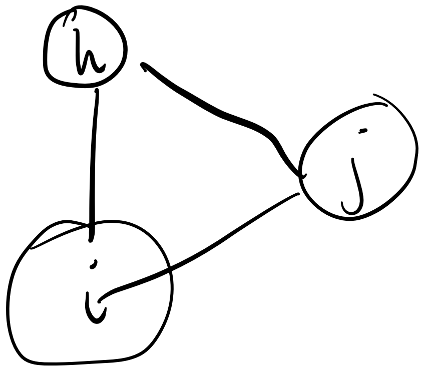
    if not visited[u]:

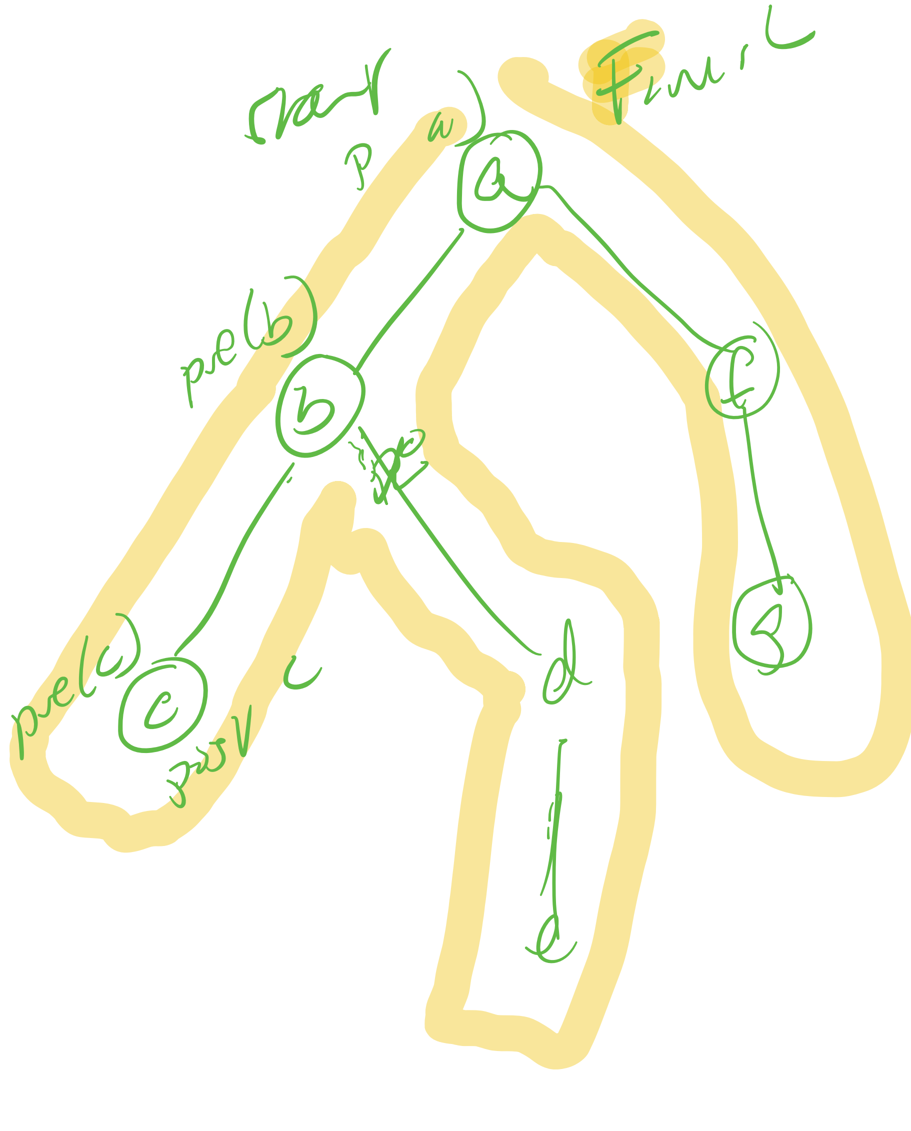
        explore(G, u)

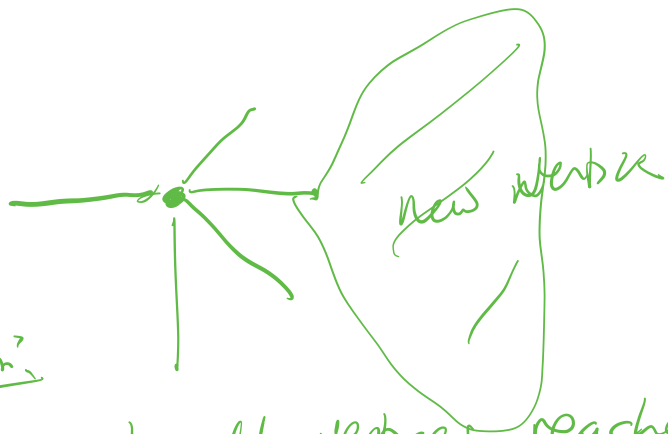
post visit(v)




pre b
post d
post b
post c
pre d
d
b
c

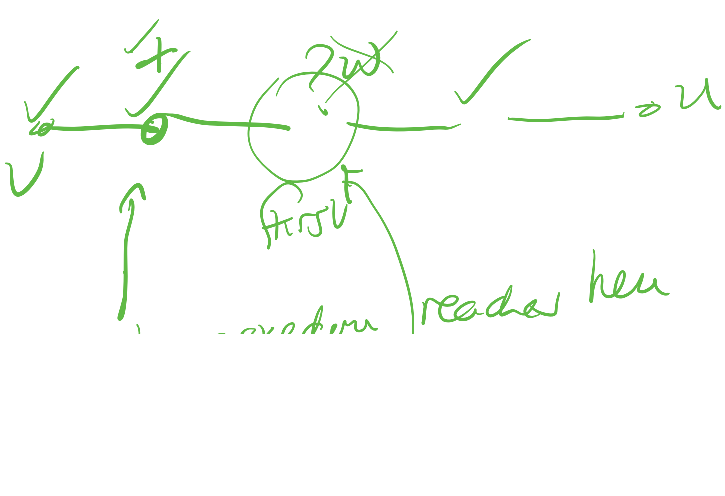

h
i
j


Start
a
pre(b)
b
pre(c)
c
post c
d
e
f
g



Claim:
We visit all vertices reachable from v

Pf: By contradiction:
If v is connected to u then there is a path
& if Explore does not reach u, then there is a first vertex on path that was not reachable



Informal

explore proceeds

but not

Contradiction bc we would reach the next vertex if not yet visited

And we only reach vertices reachable by edges

Does it halt?

Yes because each edge is visited twice!

DFS(G)

for all v in V
    visited(v) = false

for all v ∈ V
    if not visited[v]:
        explore(G, v)

[...] : visit: E15

Explore [G, conn...

G
B
H
P
N
R
conn comp

Explore [G, Cuba] !

visit : Cuba conn comp

Proc:

previsit (v)

pre(v) = clock++

postvisit (v)

post(v) = clock++

Explore