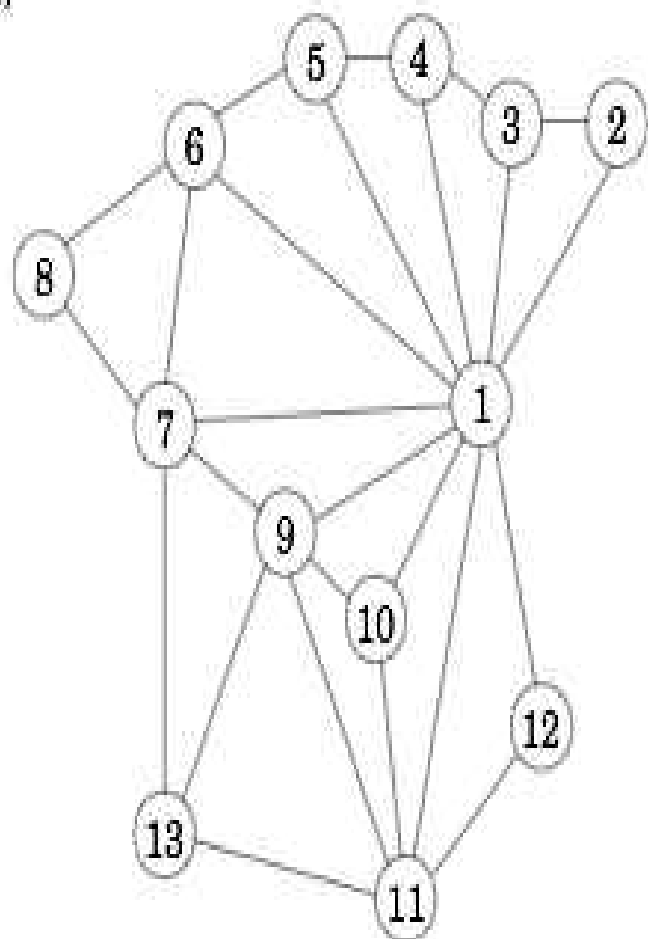






(b)



Graph $G = (V, E)$

Vertices

Edges $E \subseteq V \times V$

Map colouring :

Countries are vertices

$(c_1, c_2) \in E$ if c_1 & c_2 share
a border

Legal Map Colouring : $f: V \rightarrow C$ (C a set of
colours)

s.t. $(v_1, v_2) \in E \Rightarrow f(v_1) \neq f(v_2)$

Airline

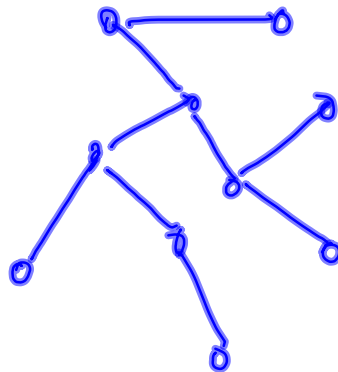
Cities served

Connections the airline has between cities

How to go from Amritsar to Coimbatore?

Cities = nodes
Vertices

Edges : direct flights



A & B share a border \Rightarrow B & A share a border

KF flies Delhi \rightarrow Amritsar $\stackrel{?}{\Rightarrow}$ KF flies Amritsar \rightarrow Delhi?

$E \subseteq V \times V$ may not be symmetric

Directed vs undirected graphs

asymmetric

symmetric edges

Usually assume 'irreflexive

Explicit vs implicit graphs

Airline route graph is explicit

Country graph is explicit

Field

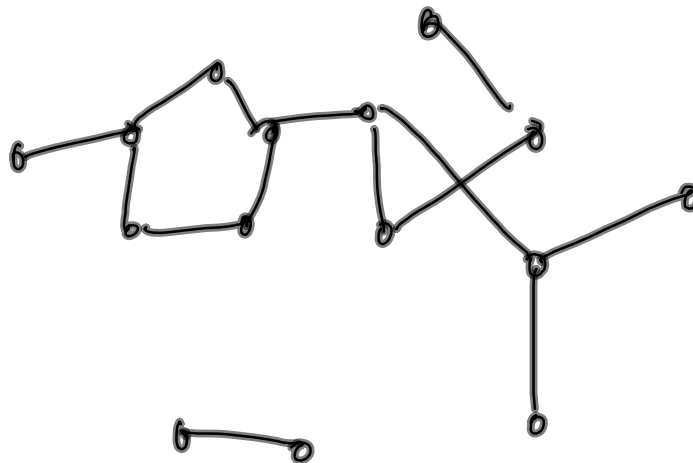
h_1	h_2	h_3	h_4	h_5
.	.	.	-	.
.	-	-	-	-
-	-	-	-	-
-	-	-	-	.

Go from one square patch to an adjacent square patch (N/S/E/W) if height diff \leq threshold

Questions about graphs:

Colouring

Colouring
Reachability - can I go from x to y ?



Visually

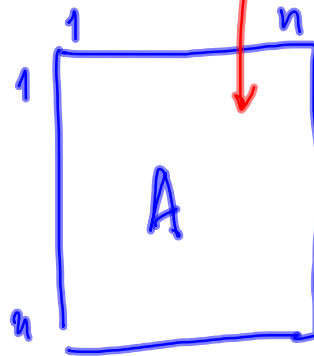
Check connectivity

Program cannot do this

Need to represent the graph

Adjacency Matrix

$V = \{1, 2, \dots, n\}$



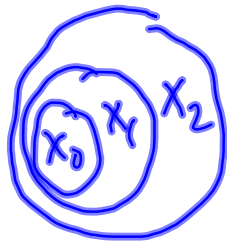
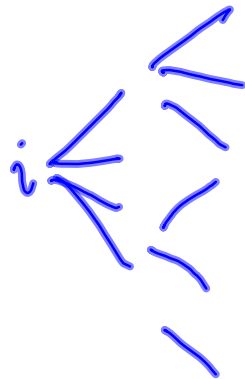
$A(i,i) = 0$, generally
(irreflexive)

symmetric if
 G is undirected

$A(i,j) =$

1, if $(i,j) \in E$
0, otherwise

Write a program to check if j is reachable from i



$$X_0 = \{i\}$$

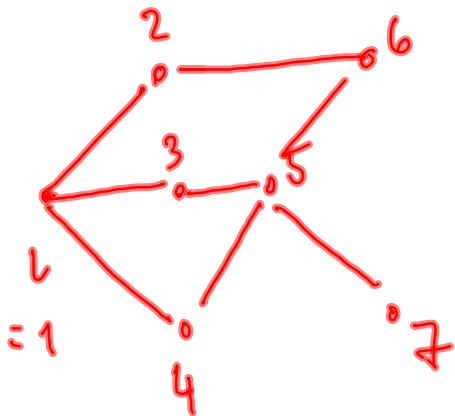
$$X_1 = \{j \mid (i,j) \in E\} \cup X_0$$

$$X_2 = \{j \mid \exists k \in X_1, (k,j) \in E\}$$

$$\vdots \quad \cup X_1$$

$$\exists m : X_{m+1} = X_m \quad (\text{V is finite!})$$

Check if $j \in X_m$



$$X_0 = \{1\}$$

$$X_1 = X_0 \cup \{2, 3, 4\} = \{1, 2, 3, 4\}$$

$$X_2 = X_1 \cup \{6, 5, \cancel{8}\}$$

Two attributes associated
with each vertex:

$$X_3 = X_2 \cup \{7\}$$

Has it been seen before?

Has it been explored (i.e. have its neighbours
been computed)?

level by level exploration

- "Mark" each vertex when we see it first

$\text{mark}[v] = 1$

When I explore an edge (j, k)

↓ if $\text{mark}[k] == 0$, set $\text{mark}[k] = 1$

↓
Process all outgoing
edges from j at one
shot

& add it to the list
of vertices to be
explored

list of unexplored vertices (i.e. reached, but yet
to explore neighbours)

Natural to use queue

explore a vertex v :

for each $(v, w) \in E$

Scan row v in A

if $\text{mark}(w) == 0$

$\text{mark}(w) = 1$

add w to queue

Breadth First Search (starting at v)

BFS (v):

mark [v] = 1

add v to queue Q

while Q is not empty

$w = \text{extract_head}(Q)$

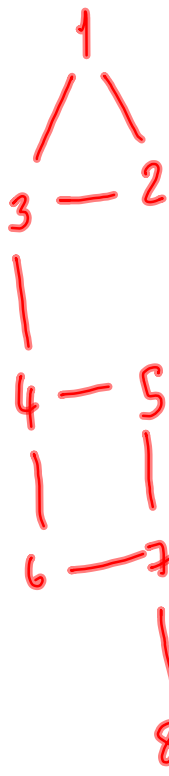
for each u s.t. $(w, u) \in E$

if mark [u] = 0:

mark [u] = 1

add u to Q

BFS(5)



Mark: 1 2 3 4 5 6 7 8 9

~~0~~ ~~0~~ ~~0~~ ~~0~~ ~~0~~ ~~0~~ ~~0~~ ~~0~~ ~~0~~

~~1~~ ~~1~~ ~~1~~ ~~1~~ ~~1~~ ~~1~~ ~~1~~ ~~1~~ ~~1~~

Queue = ~~5~~, ~~4~~, ~~7~~, ~~3~~, ~~6~~, ~~8~~, ~~1~~, ~~2~~

How much time does this take?

Each vertex enters Q at most once

Processing nbrs of v takes $O(n)$

$O(n^2)$ assuming adj. matrix

$|V| = n$ $|E| = m$ Clearly, undirected graph $\Rightarrow m \leq \binom{n}{2}$
 directed $\Rightarrow m \leq n(n-1)$

Sparse graphs m is $O(n)$

Can we improve on $O(n^2)$ BFS for sparse graphs?

Maintain edges as a list

Adjacency
list

$1 \rightarrow \{2, 7\}$

$2 \rightarrow \{1, 8\}$

\vdots

$8 \rightarrow \{2, \dots\}$

$(1, 2), (1, 7) \in E$

$(2, 1), (2, 8) \in E$

\vdots

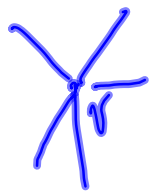
\cdot

What is the complexity of BFS now?

Outer loop is still $O(n)$ — each vertex enters Q at most once

Processing neighbours (v)

Proportional to no. of neighbours

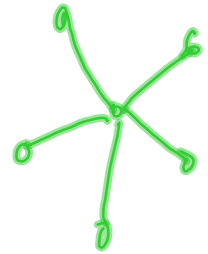


$$\Downarrow$$
$$\text{degree}(v) = |\{w \mid (v, w) \in E\}|$$

Accounting costs more carefully

$$\deg(v_1) + \deg(v_2) + \dots + \deg(v_n)$$

← n outer loops →



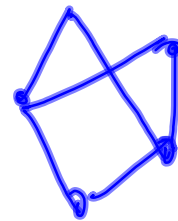
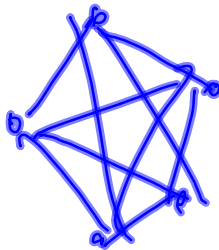
Assuming entire graph is connected

BFS takes $\sum_{v \in V} \deg(v) = 2m$

$(i,j) \in E$ contributes 1
each to $\deg(i), \deg(j)$

Graphs have 2 size parameters $|V|=n$
 $|E|=m$

An algorithm is said to be linear if it
works in time $O(n+m)$



To follow

① BFS identifies reachable vertices level by level.

Can we recover the length of the shortest path (in terms of no. of edges) to each reachable vertex?

② Extract an actual path from v to w for each reachable vertex w .