Graphs: $G = (V, E)$ $E \subseteq V \times V$ < undirected / directed

Represent as adjacency matrix A $n \times n$ 0-1 matrix

$|V| = n$

adjacency list $i \mapsto$ nbrs of i

Exploring a graph via breadth first search (BFS)

```
BFS(s)
    ∀v. mark[v] := 0          ← O(n) where |V| = n
    mark[s] := 1
    insertqueue(Q, s)
    while Q is not empty
        v := extract_head(Q)
        for each nbr u of v
            if u is not marked
                mark[u] := 1
                insertqueue(Q, u)
```

With adjacency list

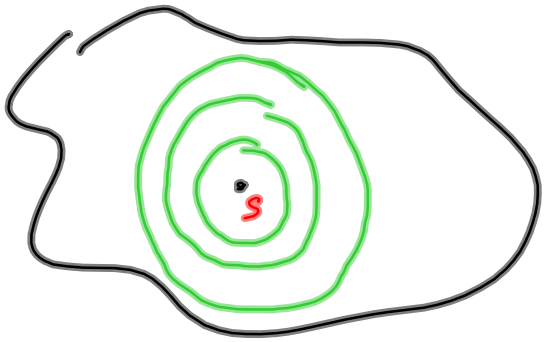
$O(m)$

where $|E| = m$

$O(n+m)$

by defn, linear time

BFS visits vertices in order of distance (no. of hops) from source



Can record this distance in BFS

Initalize distance[s] = 0

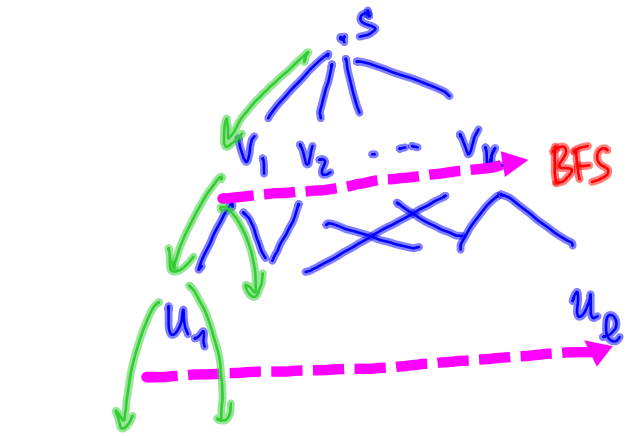
Each time v adds neighbour u to the queue

distance[u] = distance[v] + 1

Record path to each reachable vertex

parent[u] = v when v adds u to queue

An alternative strategy for exploration

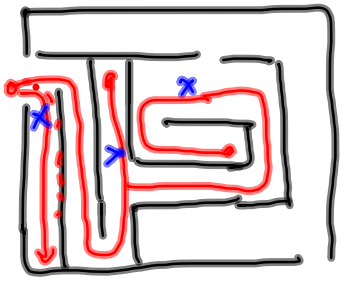


DFS

Depth first

Mark as before

Store unfinished work on a stack



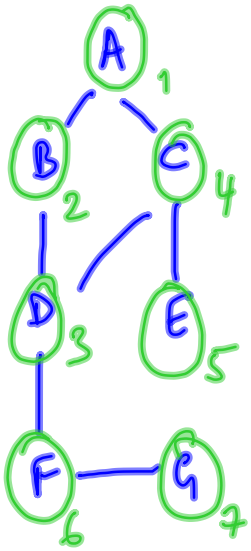
explore (v)

mark[v] = 1

for each nbr u of v

if u is unmarked

explore_(u)



explore(A) → explore(B) → explore(D) → explore(C) → explore(E)

explore(F)

explore(G)

```
dfs(G)
    for all v  mark[v] = 0
    for each u in V
        if u is unmarked
            explore(u)

explore(v)
    mark[v] = 1
    previsit(v)
    for each nbr u of v
        if u unmarked
            explore(u)
    postvisit(v)
```

e.g. previsit(v)

```
    sequence[v] = count
    count = count + 1   // count set to 0 initially
```

Using previsit(v) & postvisit(v) can maintain two values pre[v], post[v] that refer to a common counter

Initially count = 0

previsit(v)

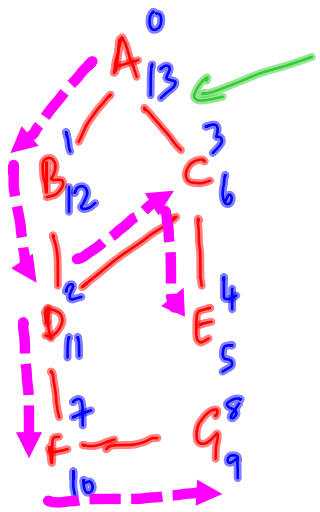
pre[v] = count

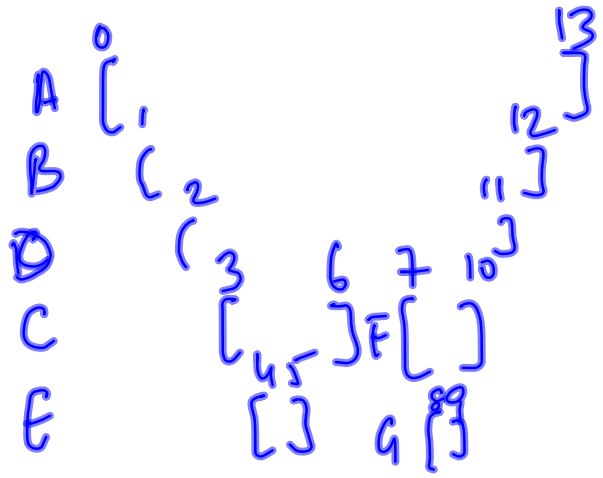
count = count + 1

postvisit(v)

post[v] = count

count = count + 1

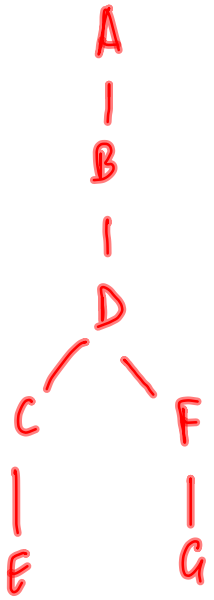




DFS Tree

Connected, no loops

DFS tree



Pre(v), Post(v) intervals are well behaved

Must be nested or disjoint

Cannot overlap in some nontrivial way

Interval(u) ⊆ Interval(v)

u was explored via v

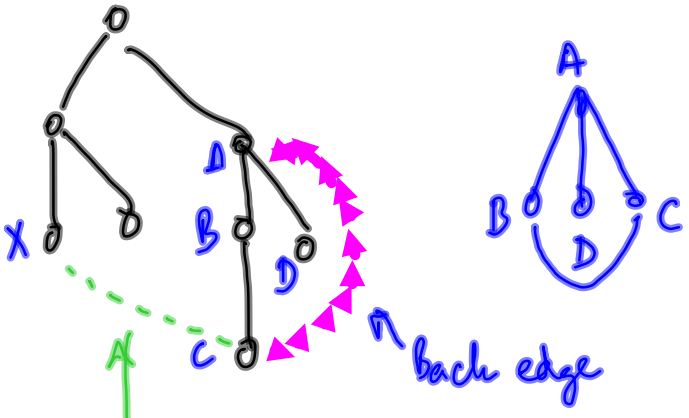
Interval(u) ∩ Interval(v) = ∅

unconnected in DFS tree

G has DFS tree edges & back edges

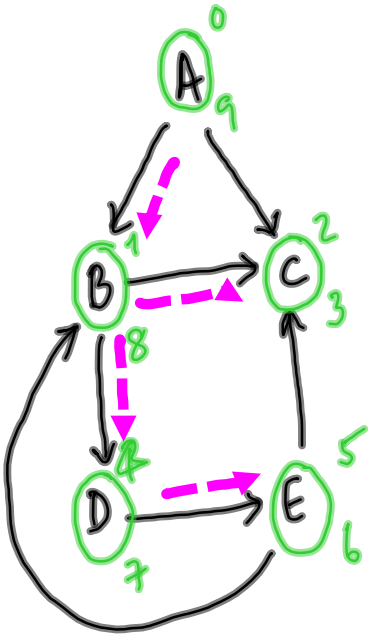
undirected

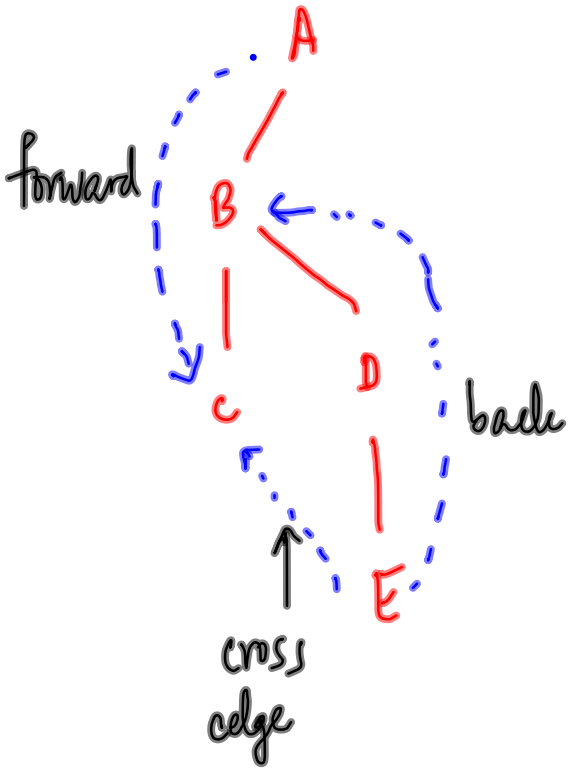
G has a cycle iff DFS tree has a back edge



not possible!
Should have visited C from X

What if G is directed?





Supplementary Reading

Algorithm Design — Kleinberg & Tardos

Algorithms — Dasgupta, Papadimitriou & Vazirani