Lecture 4 CS124

Graphs

Adjacency matrix representation

$V = \{1 \ldots n\}$

$$\begin{array}{c|c} & \begin{matrix} 1 & & & n \end{matrix} \\ \hline \begin{matrix} 1 \\ \\ \\ n \end{matrix} & \begin{bmatrix} 0 & 1 & \cdots & 0 & 1 \\ & & & & \\ & & & & \\ & & & & \end{bmatrix} \end{array}$$

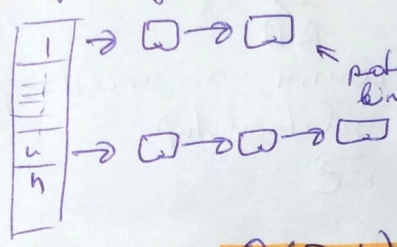
$a_{ij} = 1$ if there is a directed edge $i \to j$

Is $(i,j) \in E$? $O(1)$

what are i's neighbors? $O(n)$

$\Theta(n^2)$ space ← bad for sparse graphs

Adjacency list representation



$(i,j) \in E$? $O(\deg(i))$

potentially binary search tree

$O(\log \deg(i))$

what are i's neighbors? $O(1)$ pointer

$\Theta(E+V)$ space

$\Theta(\deg(i))$ list copy

graph/data representation choice:
→ graph sparsity
→ which operations expected

Depth First Search (DFS)

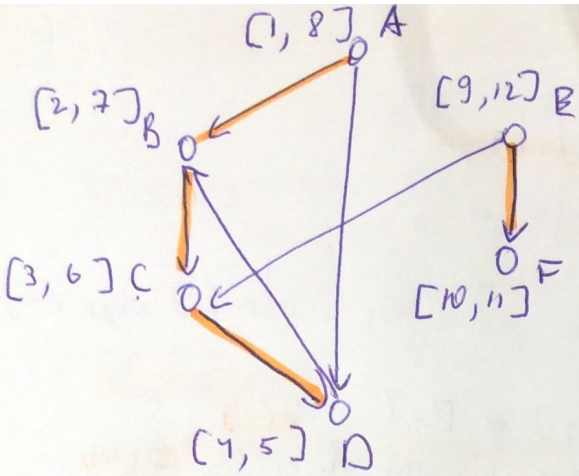
→ adjacency list representation
→ a local search algorithm
↳ exploit local information to traverse, can keep track record of the past

```
Proc Search (v)
    explored(v) := 1
    previsit (v)
    for (v,w) ∈ E          ← O(E)
        if explored(w) = 0
            search (w)
    postvisit (v)
```

each edge crossed only once

```
DFS (G)
    for each v ∈ V                 O(V)
        explored(v) := 0
    for each v ∈ V                 O(V)
        if explored(v) = 0
            search (v)
```

$O(E+V)$ ← can be graph with only vertices, or with a lot of edges.



[Start time, Finish time] disjoint or within each other.

(Forest)

Tree edges: edge that is passed in DFS

Back edges: go from a descendent to ancestor in the forest (tree)

$\overrightarrow{DB}$

Forward edges: from ancestor to a descendent in the forest

$\overrightarrow{AD}$

cross edges: between cousins, no ancestor / descendent relationship

$\overrightarrow{EC}$

DFS
⇕
Nodes on a stack: intervals either disjoint or within each other.

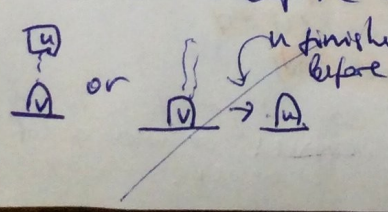
[redacted] $(u,v) \in E$, [redacted]

prove **postorder(u) < postorder(v) ⟺ (u, v) is a back edge.**

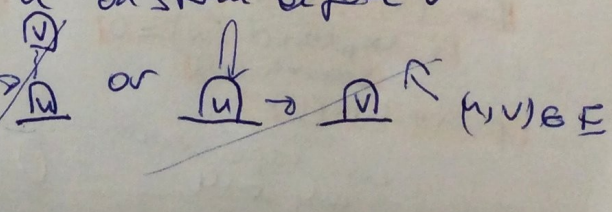
⟹ u finishes before v
v put on stack before u

[redacted] ⟹ ∃ path from v to u that is used
⟹ (u, v) must be a back edge

v on stack before u

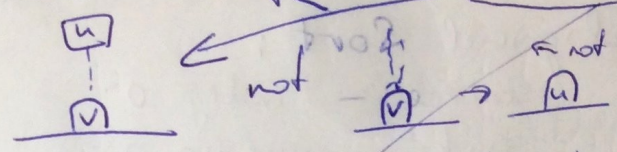


u on stack before v



$\Leftarrow$ [redacted]

by definition
u is descendant of v
in DFS forest



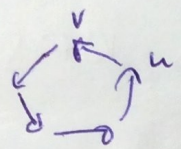
$\Rightarrow$ postorder(u) < postorder(v)

Tree, back, cross, forward edges are relative to a specific search. complete and disjoint classification

G has a cycle $\Leftrightarrow$ DFS has a back edge

$\Leftarrow$ obvious, by definition of cycle

$\Rightarrow$



let u be the vertex in the cycle with the smallest postorder

since u is in the cycle $\exists (u,v) \in E$

$\Rightarrow$ where v is part of the cycle.

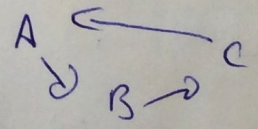
$\Rightarrow$ postorder(u) < postorder(v)

$\Rightarrow$ (u, v) is a back edge

<u>Debugging</u> Proc A calls Proc B, debug B before A, A depends on B

[redacted]

but mutual recursion problem



assume no mutual recursion for now

↳ no cycles

↳ directed acyclic graph (DAG)

Given a DAG, sort it

if $\exists\ (a,b) \in E$, then a comes before b in the list

Topological Sort:

source - indeg 0

sink - outdeg 0

$\boxed{\text{Do a DFS} \quad \text{Sort decreasing postorder \#}}$ $\quad O(E+V)$

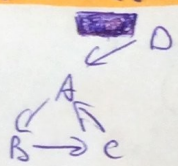
Proof

acyclic $\Leftrightarrow$ no backedges

$\forall (u,v) \in E \Rightarrow$ postorder $(u) >$ postorder (v)

Reduction: Topological search to DFS

Now assume mutual recursion



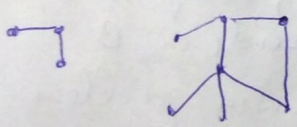
debug D first

then treat A B C as one

$\rightarrow$ debugging order

Strongly Connected Components (SCC)

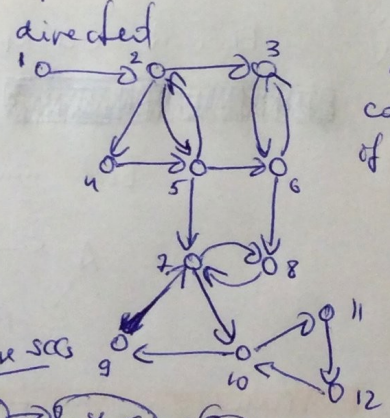
undirected



connected

$\Rightarrow$ strongly connected

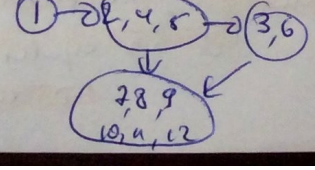
directed



a is strongly connected to b if $\exists$ path $a \to b$ and $b \to a$

- reflexive
- symmetric
- transitive

collapse SCCs



get DAG $\rightarrow$ do topological sort