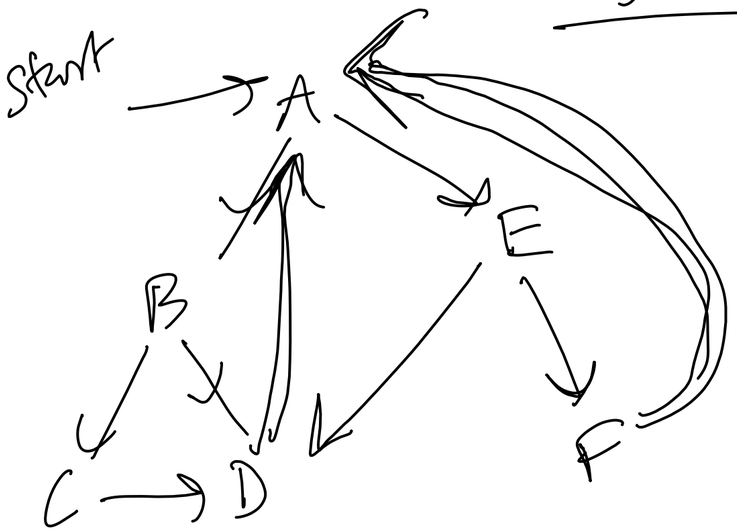
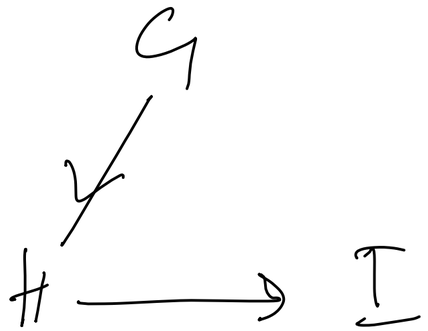
19.08.2024





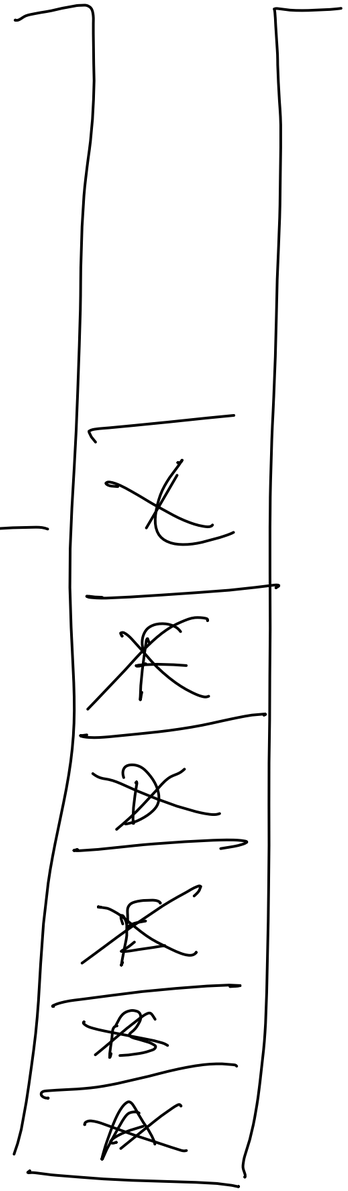
Adjacency list

A → B → E

B → C → D

C → D

A B E D E F G H I



Procedure DFS(G, v, A)

start vertex

// G = (V, E)

S ← ϕ

for each v ∈ V

A[v] ← 0 → Θ(n)

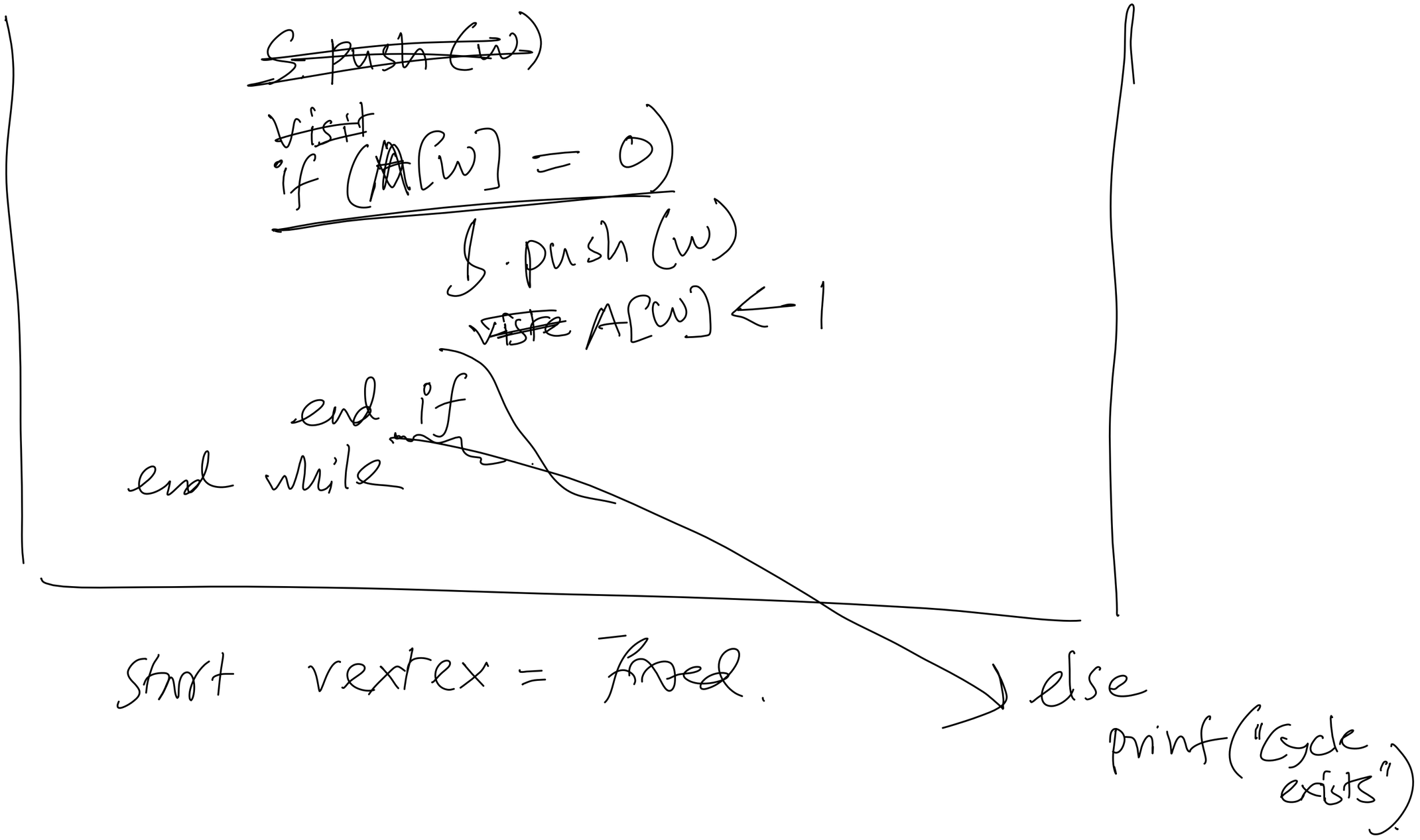
S.push(v).

A[v] ← 1

while (S.top ≠ 0)

u ← S.pop()

→ for each (u, w) ∈ E


~~S.push(w)~~
~~visit~~
if (~~A~~[w] = 0)
S.push(w)
~~visit~~ A[w] ← 1
end if
end while
start vertex = fixed.
else
print("cycle exists")

```
Procedure Explore(G)
// G = (V, E)
~~for each~~ -
A ← empty array of size |V| all zeros
~~for each i = 1, to |.~~
for each v ∈ V
    if (A[v] ≠ 0)
        DFS(G, v, A)
    end if
end for.
```

Correctness

Claim: DFS(G, v, A) visits all nodes reachable from v only & none else.

Proof: By induction on the # of hops from v.

I.H.: All ℓ hop vertices are reachable.

I.S.: proof is established. ($v_\ell \to u$) edge is there.

Base case: $\ell = 1$ is true.

Claim: Explore is going to mark all vertices as visited.

proof: skipped.

time complexity:

Claim: All edges $(u, v) \in E$ is seen by Explore exactly once.

proof: u is visited at some point.
Consider when u is ~~visited~~ popped, v is pushed.

Therefore u is not going to be pushed again because u is marked visited.

Therefore (u,v) will never be considered again □.

Time:

$\Theta(n) + \Theta(m)$

↓ for loop for A[v] = 0

↓ seeing each edge once.

$\Theta((n+m)\log n)$

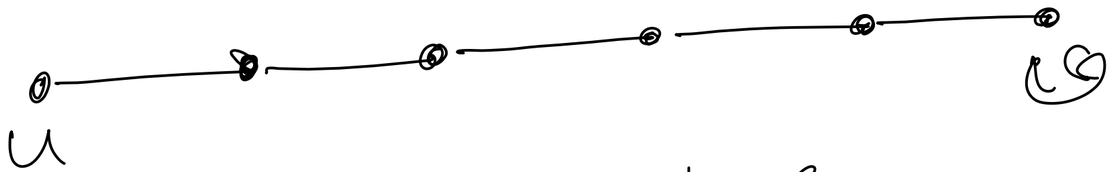
Assumption: Accessing each vertex takes unit time

C = constant.

2^{20}, 20 bits ≡ 1 machine instruction.

Remarks

- DFS algo. described also works for undirected graphs



reachability is symmetric here.

- DFS can be performed recursively

Procedure RDFS(G, v, A)

↓ start vertex

// G = (V, E)

A[v] ← 1

for each (v, w) ∈ E

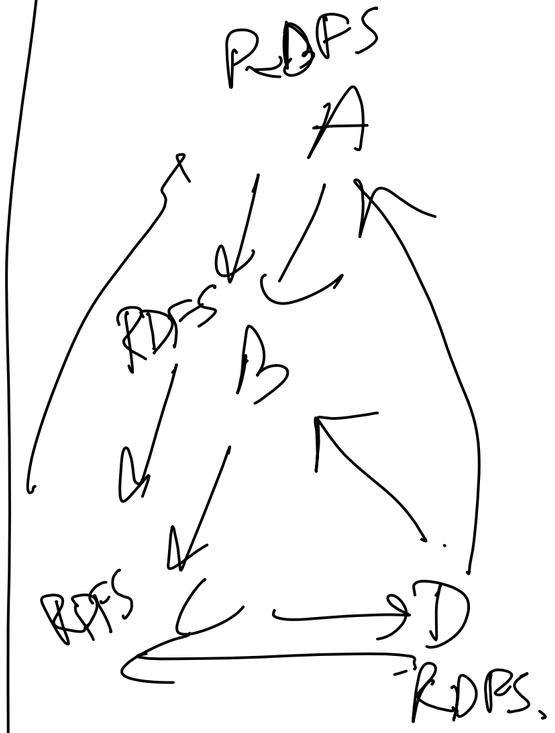
if (A[w] = 0)

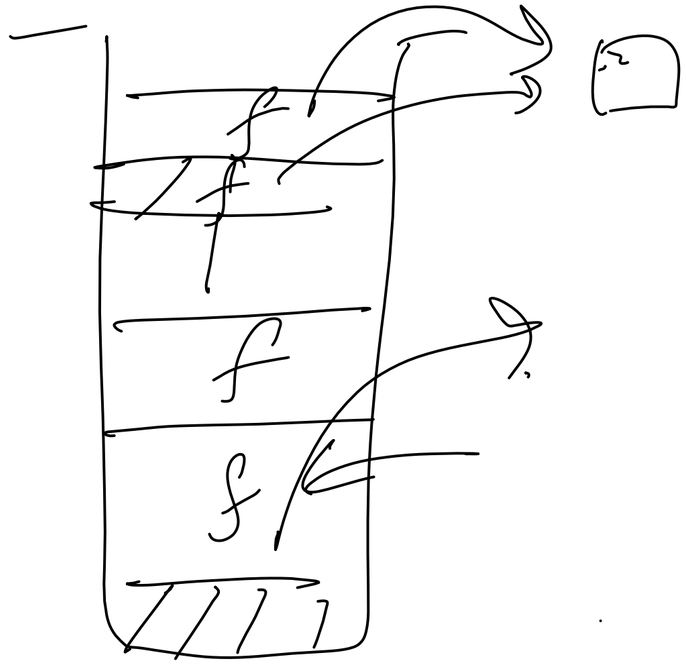
{A[w] ← 1

RDFS(G, w, A).

end if

end for.





Inside a programming language any recursive function is implemented using a stack!

Definition of Graph Theory Terminology.

Directed graphs.

Defn. [Parent & Child]

$A \longrightarrow B$
(parent) (child).

A is parent of B
B is a child of A.

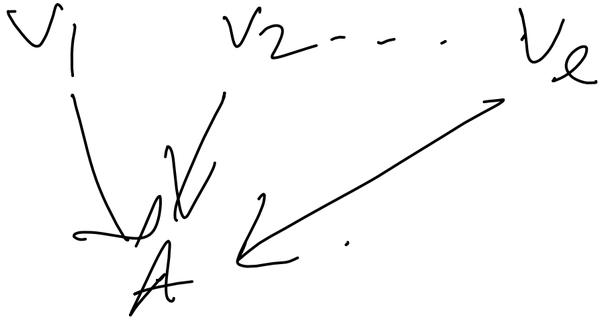
Defn [Ancestor & Descendant].

$A \to V_1 \to V_2 \to \cdots \to V_k \to B.$

A is an ancestor of B
B is a descendant of A.

Defⁿ [Indegree & Outdegree].

$\{v_1, v_2, \ldots, v_\ell\}$ = Set of parents of A.

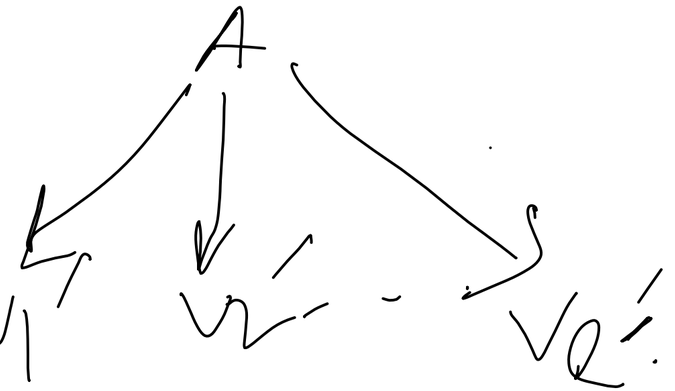


Cardinality of the parent set is called indegree.

indegree[A] = ℓ.

$\{v_1', \ldots, v_{\ell'}'\}$ = set of children of A

Cardinality of the children set is called out degree.



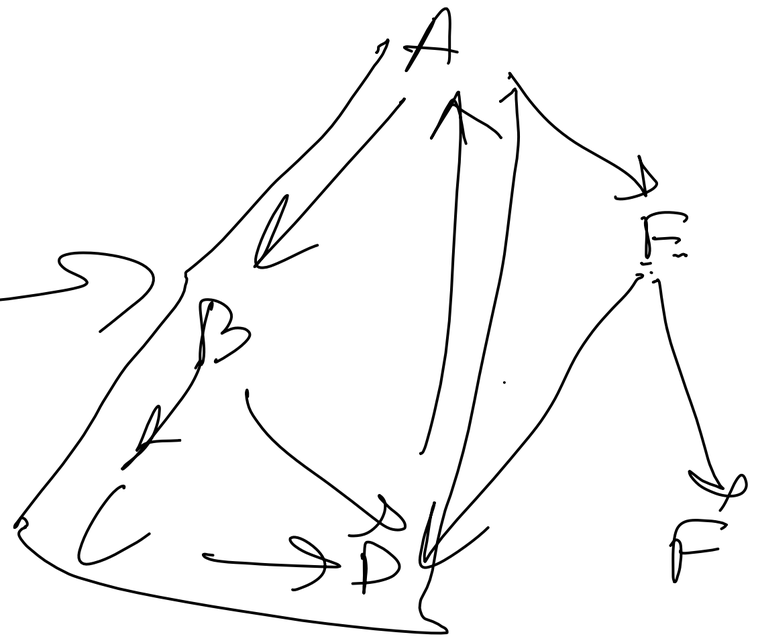
outdegree[A] = ℓ'

Defn [Source & Sink]

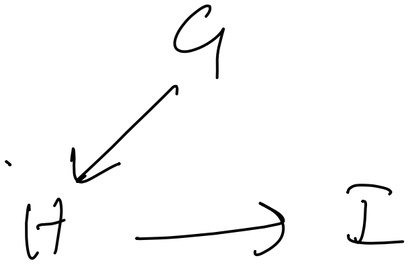
Source node ≡ Indegree = 0.

~~A is a S~~

There is no source node here.



G is a source node.



Sink node ≡ Outdegree = 0.

F, I are sink nodes.

Defn [Cycle]

A sequence of vertices $(v_1, v_2, \ldots v_k)$ is called a cycle

— $v_k = v_1$

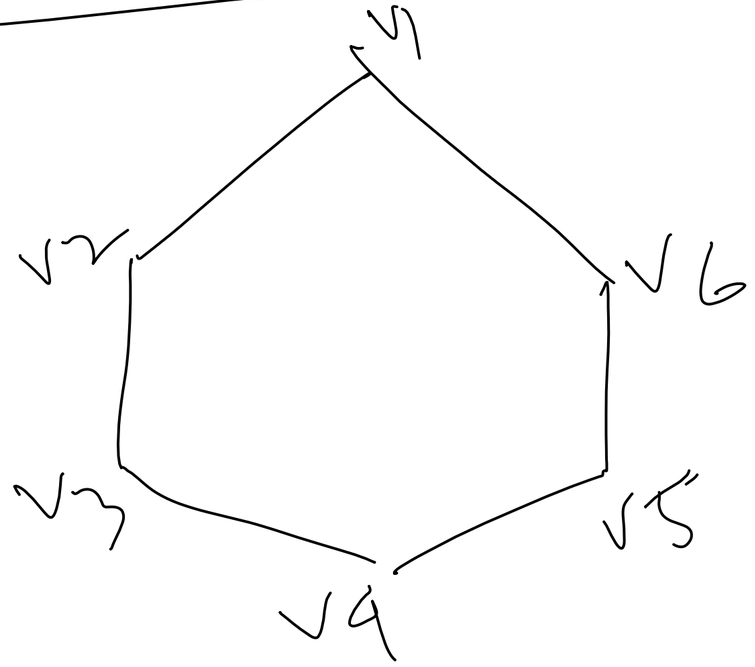
— $(v_1, v_2) \in E$

e.g. (A, B, C, D, A) is a cycle.

Undirected graph

Defn [Neighbors]

Neighbors of v is the set of nodes directly connected to v.



v_6's neighbors will be $\{v_1, v_5\}$

Defn [degree]

Degree of v = Cardinality of the set of neighbors of v.

degree(v_6) = 2.

Defn [Reachable] . A node u is reachable from v if $\exists$ a seq.

$v \; v_1 \; - - - \; v_{l-1} \; u$,

such that $(v_i, v_{i+1}) \in E \quad \forall \; i = 1 \text{ to } l-1$

$(v, v_1) \in E$

$(v_{l-1}, u) \in E$

Reachability is symmetric.

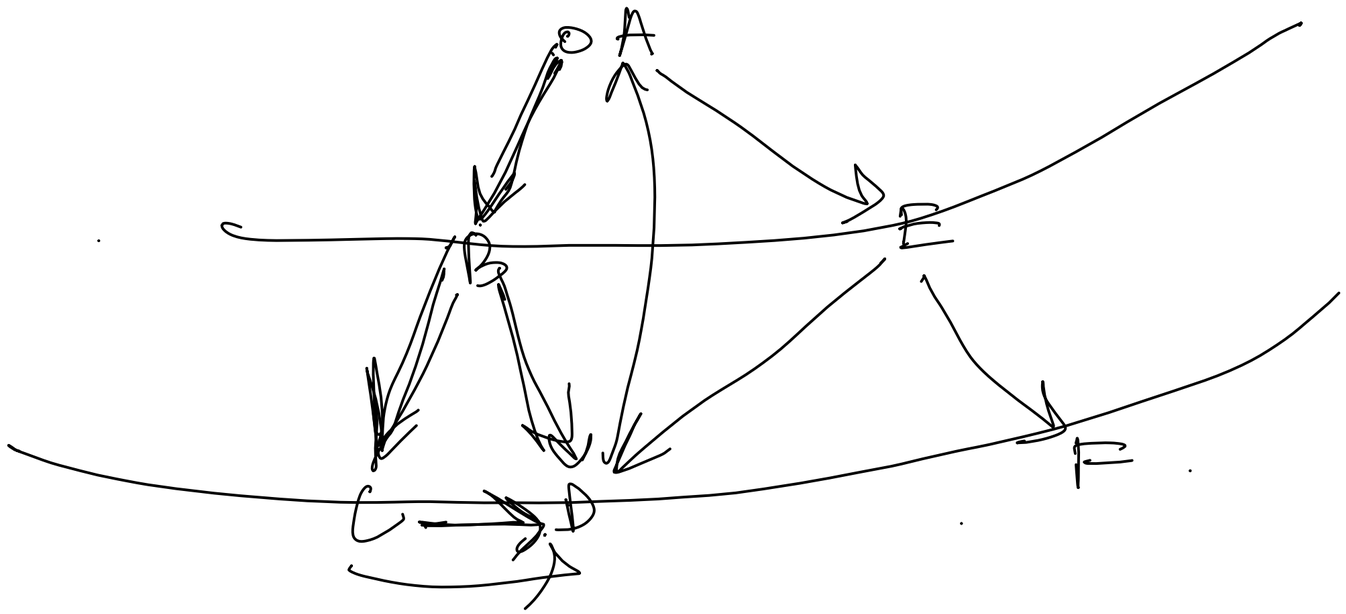
Some more observations about DFS.

Q: how to check if G has a cycle or not.

A: If we revisit a vertex in dfs using a "back edge"

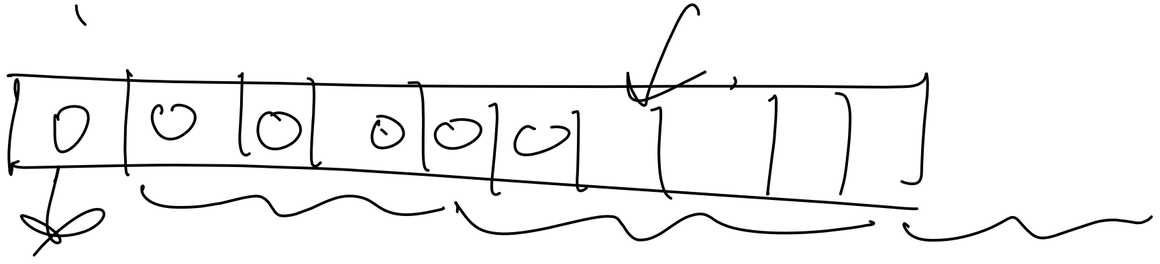
Breadth-First-Search

- explore the graph level by level.

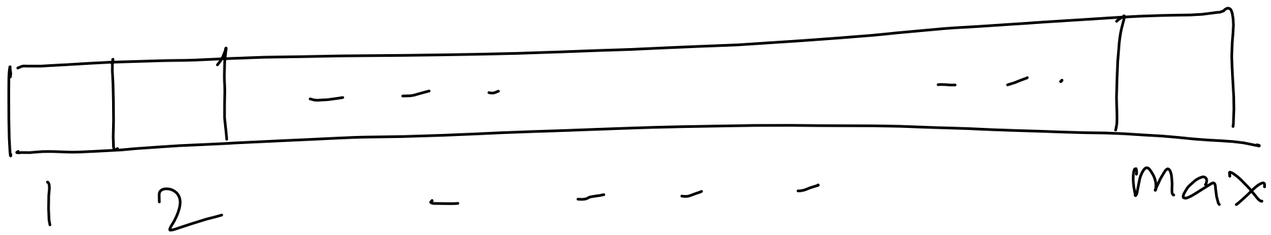


BFS always gives you the shortest path.

Queue.



Detour: Queue Data structure.



Dequeue: remove 1 item from Q in the front of the queue.

Enqueue: - add 1 item to Q in the end of the queue.

Q. head — front of the Queue.

Q. tail: end of the Queue.