really reachable verts and edges

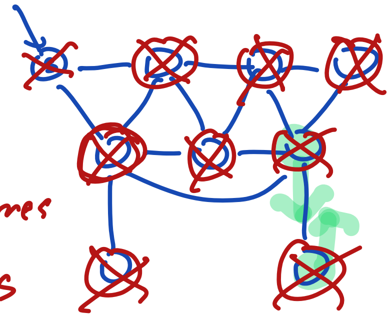
↓ ↓

$O(V+E)$

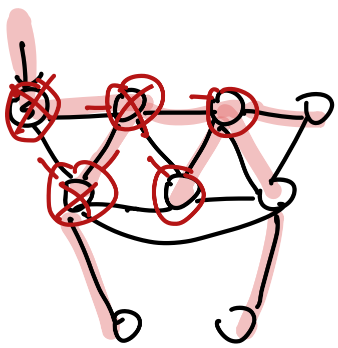
WFS marks every vertex reachable from s.

insert (if not already there)
extract one
empty?

$\underline{O(1)}$



```
WHATEVERFIRSTSEARCH(s):
  put s into the bag                  ←
  while the bag is not empty
      take v from the bag             ← 2E+1 times
      if v is unmarked
          mark v                      ← V times
          for each edge vw            = for each neighbor w of v
              put w into the bag      ← 2E times
```

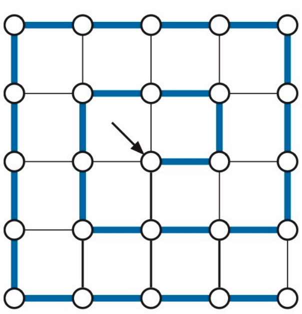


```
WHATEVERFIRSTSEARCH(s):
  put (∅, s) in bag
  while the bag is not empty
      take (p, v) from the bag                 (⋆)
      if v is unmarked
          mark v
          v.parent ← p
          for each edge vw                     (†)
              put (v, w) into the bag          (⋆⋆)
```

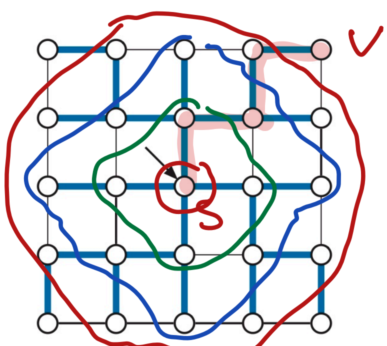
WFS assigns a parent to every vertex reachable from s (except s itself)

$v \to v.parent$ <u>no cycles</u> $\Longrightarrow$ parent edges define a <u>spanning tree</u> of reachable verts.

$V-1$ parent edges



DFS (stack)



BFS (queue) → <u>shortest-path tree</u>

"Best-first":

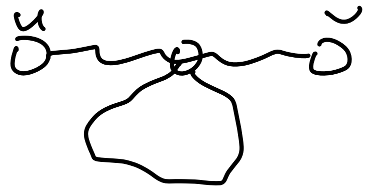
Dijkstra — priority queue

Prim/Jarník — MST

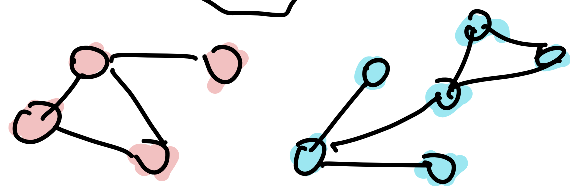
<u>widest</u> path

Undirected

connectivity $O(V+E)$



components $O(V+E)$



shortest paths

Breadth $O(V+E)$

Dijkstra $O(E \log V)$

$O(V_1+E_1) + O(V_2+E_2)$

```
WFSAll(G):
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            WhateverFirstSearch(v)
```

$O(V+E)$ time

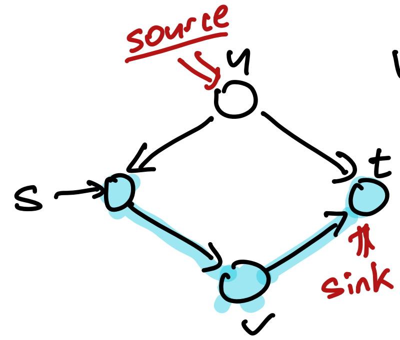
```
CountAndLabel(G):
    count ← 0
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            count ← count + 1
            LabelOne(v, count)
    return count
```

```
⟨⟨Label one component⟩⟩
LabelOne(v, count):
    while the bag is not empty
        take v from the bag
        if v is unmarked
            mark v
            v.comp ← count
            for each edge vw
                put w into the bag
```

Directed graphs

tail → head

```
WhateverFirstSearch(s):
    put s into the bag
    while the bag is not empty
        take v from the bag
        if v is unmarked
            mark v
            for each edge v→w
                put w into the bag
```

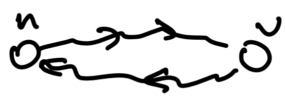


WFS builds a spanning tree of vertices reachable from s

reachability
O(V+E)

"u can reach v"
"v is reachable from u"

strong connectivity
O(V+E)



Entire graph strongly connected? O(V+E)

Acyclic? DAG? O(V+E) DFS

strong components O(V+E) DFS

```
DFSAll(G):
    Preprocess(G)
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            DFS(v)
```

count ← 0

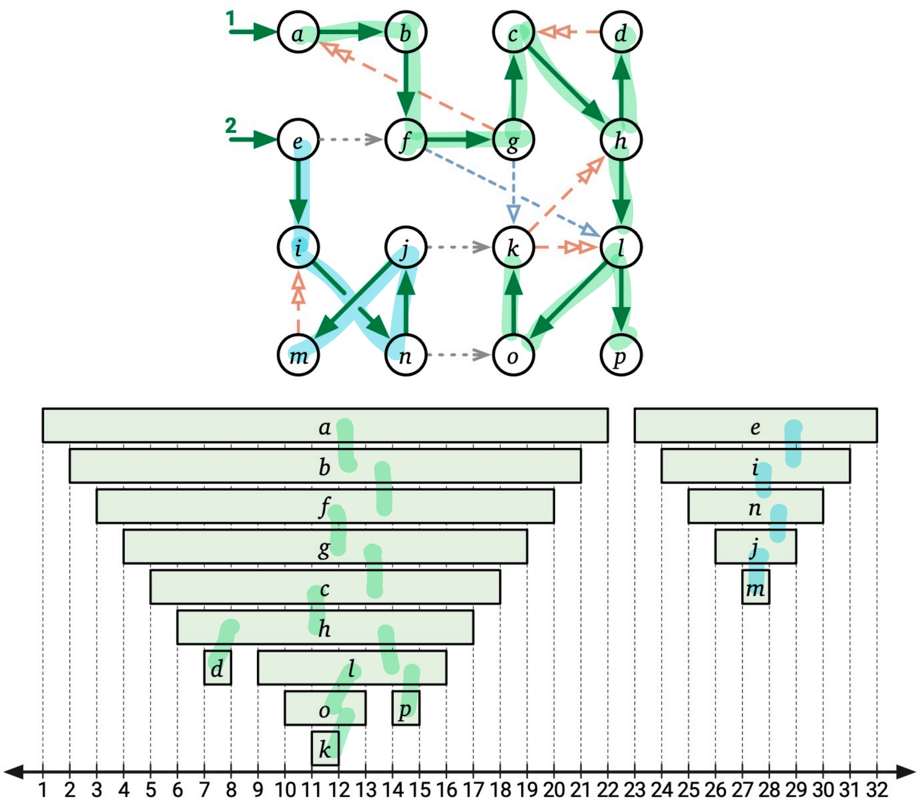
```
DFS(v):
    mark v
    PreVisit(v)
    for each edge vw
        if w is unmarked
            parent(w) ← v
            DFS(w)
    PostVisit(v)
```

count++
v.pre ← count

count++
v.post ← count

```
DFSAll(G):
    clock ← 0
    for all vertices v
        unmark v
    for all vertices v
        if v is unmarked
            clock ← DFS(v, clock)
```

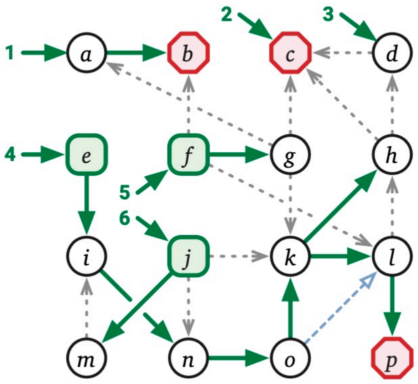
```
DFS(v, clock):
    mark v
    clock ← clock + 1;  v.pre ← clock
    for each edge v→w
        if w is unmarked
            w.parent ← v
            clock ← DFS(w, clock)
    clock ← clock + 1;  v.post ← clock
    return clock
```

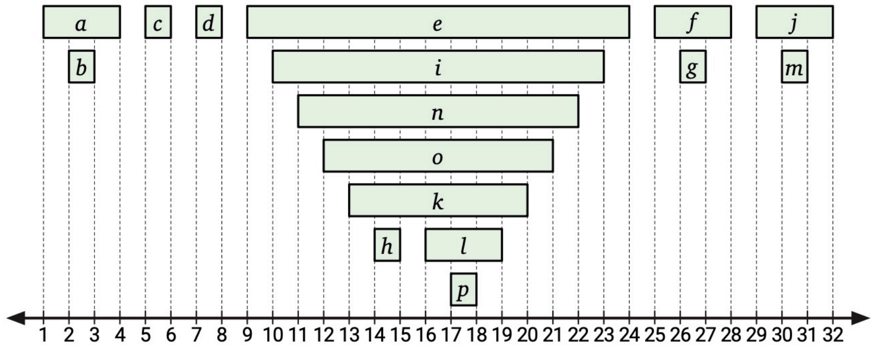


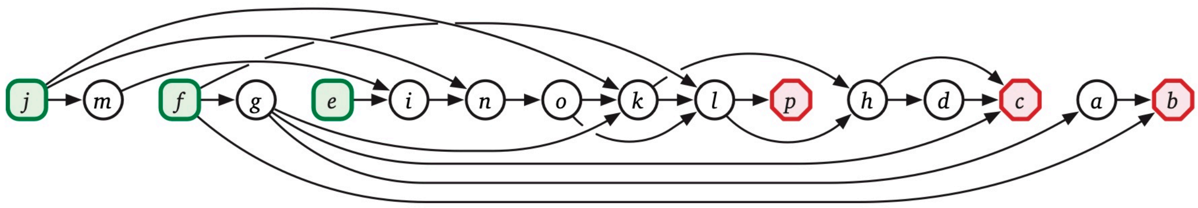
Pre: a b f g c h d l o k p e i n j m

Post: d k o p l h c g f b a m j n i e

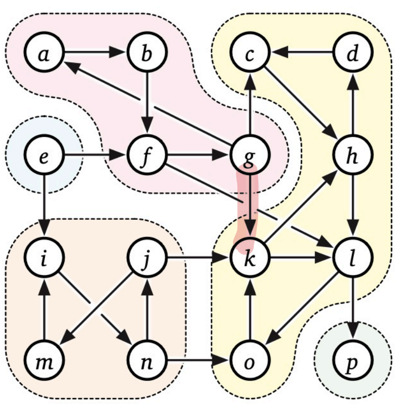
Lemma: G has a cycle iff for some edge $v \to w$ we have $v.post < w.post$





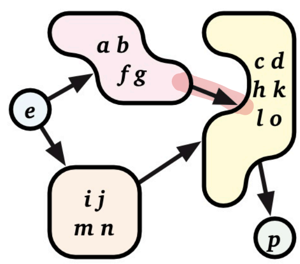



```
TopologicalSort(G):
    clock ← V
    for all vertices v in postorder
        S[clock] ← v
        clock ← clock − 1
    return S[1..V]
```



strong components

$\boxed{O(V+E) \text{ time}}$



metagraph ⟨G⟩

directed acyclic graph