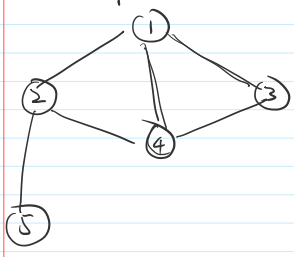
Lecture 11 DFS and BFS

Thursday, February 21, 2019 2:45 PM

- Recap: Pre/post order



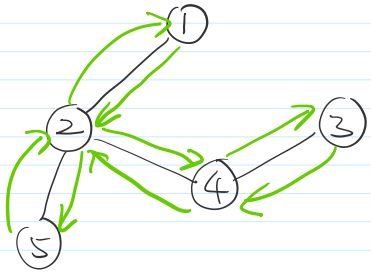
DFS_visit(1)

→ DFS_visit(2)

→ DFS_visit(4)

→ DFS_visit(3)

→ DFS_visit(5)

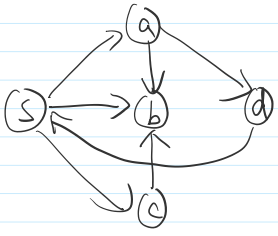


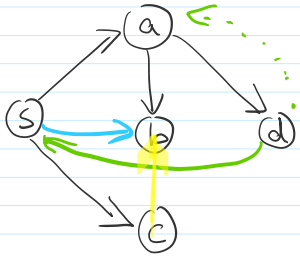
1 2 4 3 4 2 5 2 1

Pre-order 1 2 4 3 5

post-order 3 4 5 2 1

- type of edges



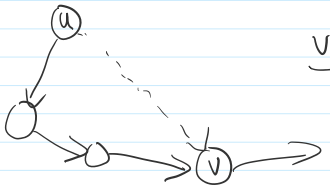


DFS-tree

- ① tree edges: edges in the DFS tree.
  ② forward edges: edge connecting a vertex to one of its descendants in the tree
  ③ backward edges: edge connecting a vertex to one of its ancestors in the tree.
  ④ cross edges: edges that connect a vertex to another vertex in a different branch.

- similar to DFS tree / pre-post order, edge-types also depend on the choice in DFS algorithm.

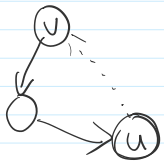
- forward edge (u, v)



visit u → visit v → DFS_visit(v) returns
→ consider (u,v)
→ DFS_visit(u) return

(( ))
u v v u

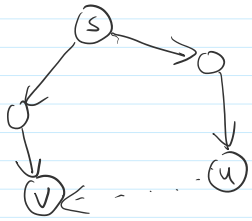
- back edge (u,v)



visit v → visit u → consider(u,v)
→ DFS_visit(u) return
→ DFS_visit(v) returns

(( ))
v u u v

- Cross edge (u,v)



visit v → DFS-visit(v) returns
→ visit u
→ consider (u,v)
→ DFS_visit(u) returns

( ) ( )
v v u u

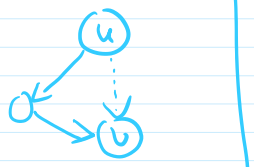
- Cycle-Finding
  - Proof of correctness.
    - Lemma: For a DFS algorithm, if when u is visited there is a path from u to v, and u is the only visited vertex on the path. then u is on the stack when v is visited. (u is later than v in post-order)
  - Proof: we prove this by induction on the length of the path.
  - Induction Hypothesis: For a DFS algorithm, if when u is visited, there is a path of length ≤ l from u to v such that u is the only visited vertex on the path. then u is on the stack when v is visited.
  - Base case: l=1
    in this case, (u,v) is an edge and v is not visited when u is visited.
    we know the sequence of following 3 events. by the DFS_visit algorithm.

visit(u) ① → consider edge(u,v) ② → DFS_visit(u) returns ③.

v must be visited [between events ① and ②] or [right after event ②]

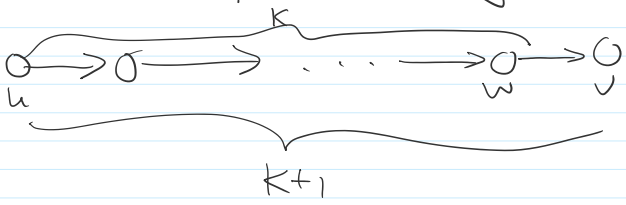
① ② ③

v must be visited [between events ① and ②] or [right after event ②]



so when v is visited we always know u is on the stack.

- Induction Step: Assume IH is true for L=k, consider the case when the path has length k+1, Consider w, the vertex before v on the path.



We know there is a path from u to w of length k, and u is the only visited vertex.

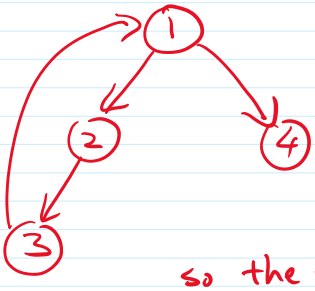
By IH, we know u is on stack when w is visited, by design of algorithm, we know the sequence of the following events

visit u ① → visit w ② → edge(w, v) considered ③ → DFS_visit(w) returns ④ → DFS_visit(u) returns ⑤

if v was visited before edge (w,v) is considered, then it is between ① and ③, so v is visited while u is on stack.

otherwise, v is going to be visited right after ③, at that time u is still on stack. this finishes the induction. □

- counter-example if we do not require u to be the only visited vertex.



let u = 3 v = 4

we run DFS at ①, and choose edge (1,2) first.

when ③ is visited, ④ is not visited, and there is a path (③, ①, ④) (except ① has been visited)

so the version of the lemma that only requires v to be visited after u will predict ③ is on stack when ④ is visited. However, this is not true. Only ① is on stack when ④ is visited.

- Proof for cycle finding:
  - assume there is a cycle $(v_1, v_2, \ldots, v_k)$

- Proof for cycle finding:
  - assume there is a cycle $(v_1, v_2, \ldots, v_k)$
  - assume (w.l.o.g) that v_1 is the first vertex visited in the cycle.
  - by lemma: we know v_1 is still on stack when DFS visits v_k.
  - now when edge (v_k, v_1) is considered, it must be a back edge.

- BFS

For a starting point u

- Lemma: BFS finds shortest path from u to any vertex v reachable from u.
- Proof: IH: BFS finds shortest path for all vertices v at a distance $\leq l$ to u.

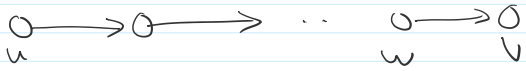
Base case: $l = 1$ (u, v) is an edge.

since BFS first considers all neighbors of u. (u,v) will be considered and BFS finds the shortest path

Induction step: assume IH is true for $l = k$.

consider a vertex v at distance $k+1$ to u.

the shortest path from u to v has length $k+1$



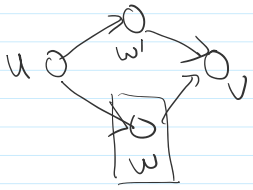
consider w: vertex before v on the shortest path.

distance from u to w is k.

using IH, BFS finds shortest path to w.

now consider the time w is processed in BFS.

① v is already in the queue

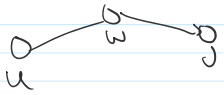


in this case, v is added to the queue by a vertex w' that is processed before w.

by design $dis(u, w') \leq dis(u, w) = k$

so BFS finds a path of length $\leq k+1$ ✓

② v is not in the queue



BFS will add v to the queue and find a path of length $k+1$ ✓