CS 381 – FALL 2019

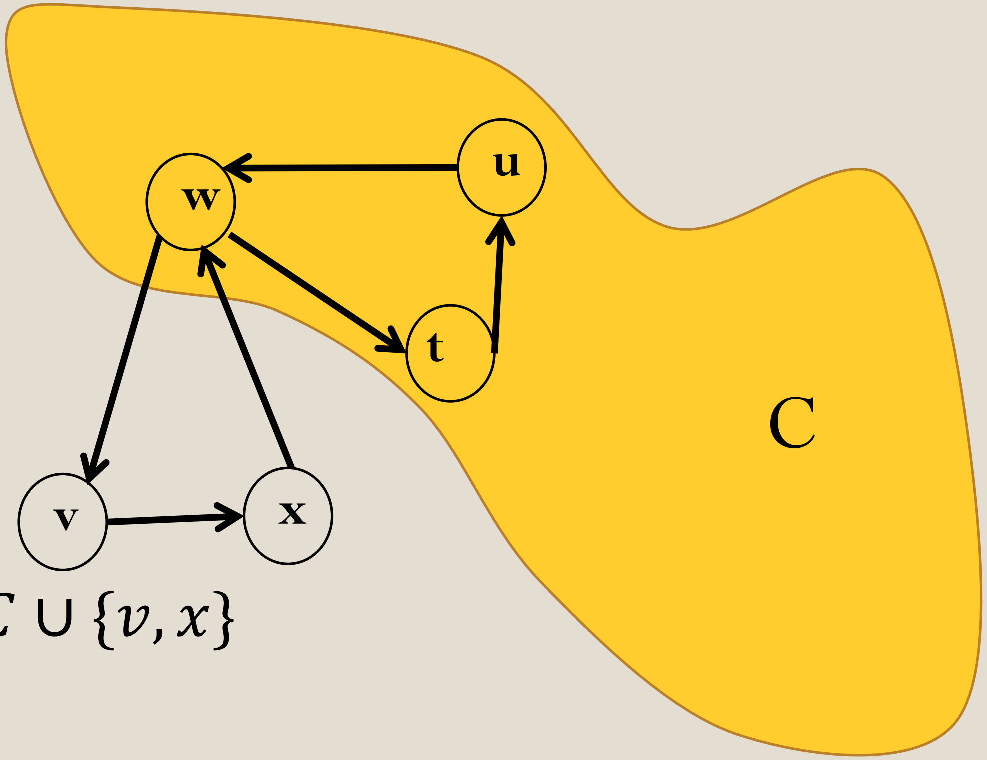
Week 9.2, Wed, Oct 16

Homework 5 Released Soon

SCC

Definition 1: A subset $C \subset V$ of nodes is strongly connected if for all $\mathrm{u}, w \in C$ the directed graph G contains a path from u to w (and vice versa)

Definition 2: A Strongly Connected Component $C \subset V$ of a directed graph G is maximal if for all sets $\mathrm{C}' \supset C$ with $\mathrm{C}' \neq C$ the set $\mathrm{C}' \subset V$ is not strongly connected

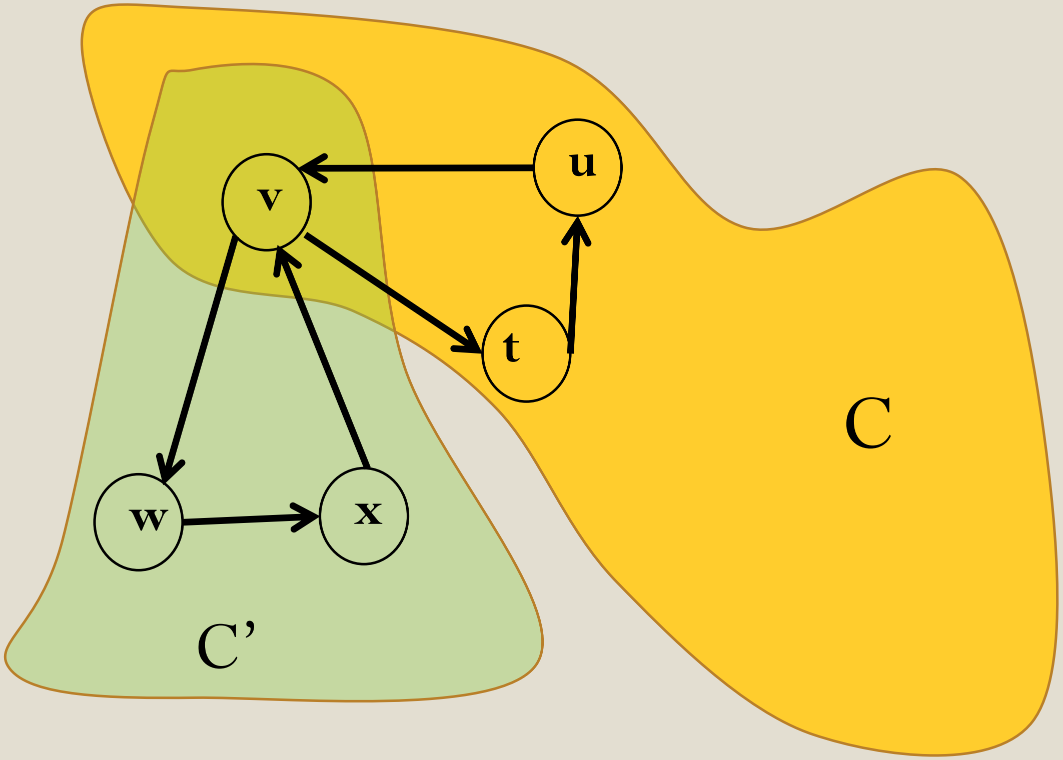


Not Maximal: $\mathrm{C}' = C \cup \{v, x\}$ is strongly connected

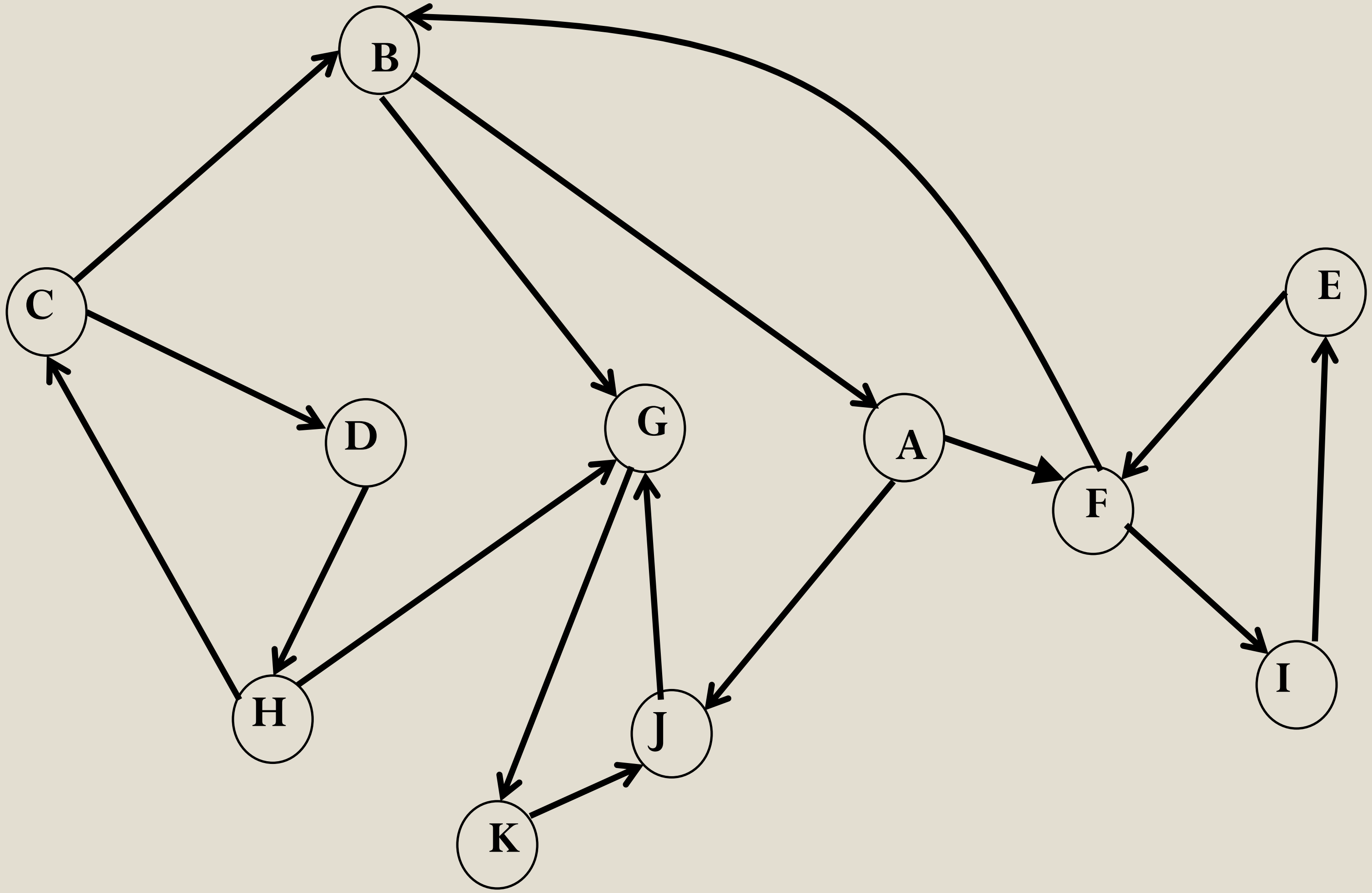
SCCs Partition V

Claim: If $C' \neq C$ are maximal SCCs then C and C' are disjoint.

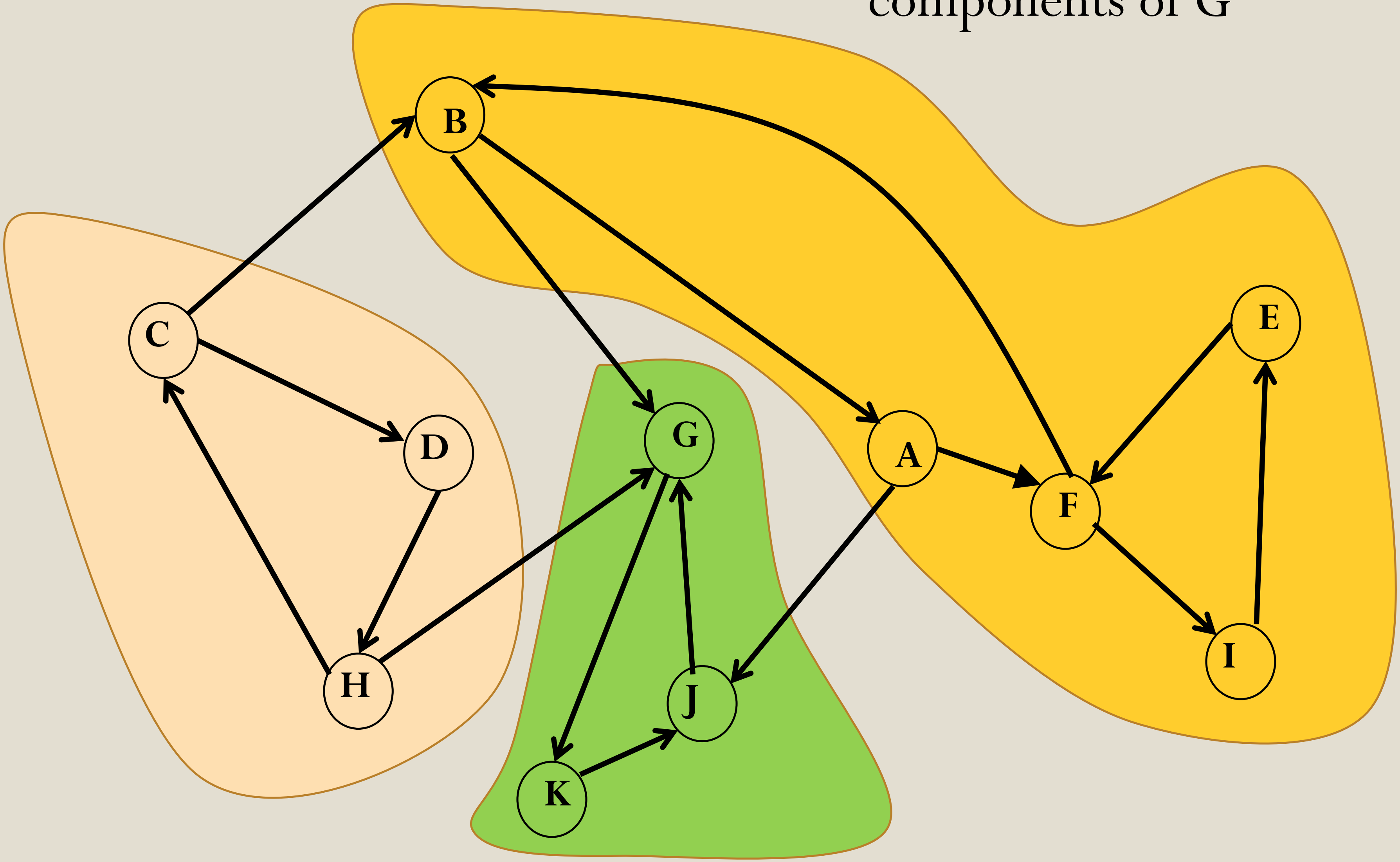
Proof: Otherwise if $v \in C \cap C'$ then $C'' = C \cup C'$ is strongly connected. Contradicts maximality of C' and C! For any pair $w \in C'$ and $u \in C$ we can find directed path from w to u (via v e.g., $w \rightarrow x \rightarrow v \rightarrow t \rightarrow u$) and can find directed path from u to w (via v e.g., $u \rightarrow v \rightarrow w$)



Directed Graph G



Strongly connected components of G



Strongly Connected Components (22.5)

Let G be a directed graph.

- G is **strongly connected** if there exists a path between any pair of vertices.
- If G is not strongly connected, decompose G into *strongly* connected components:
  - sets of vertices in which any two vertices are *mutually reachable*
  - each vertex set cannot be enlarged by adding more vertices without destroying this property.

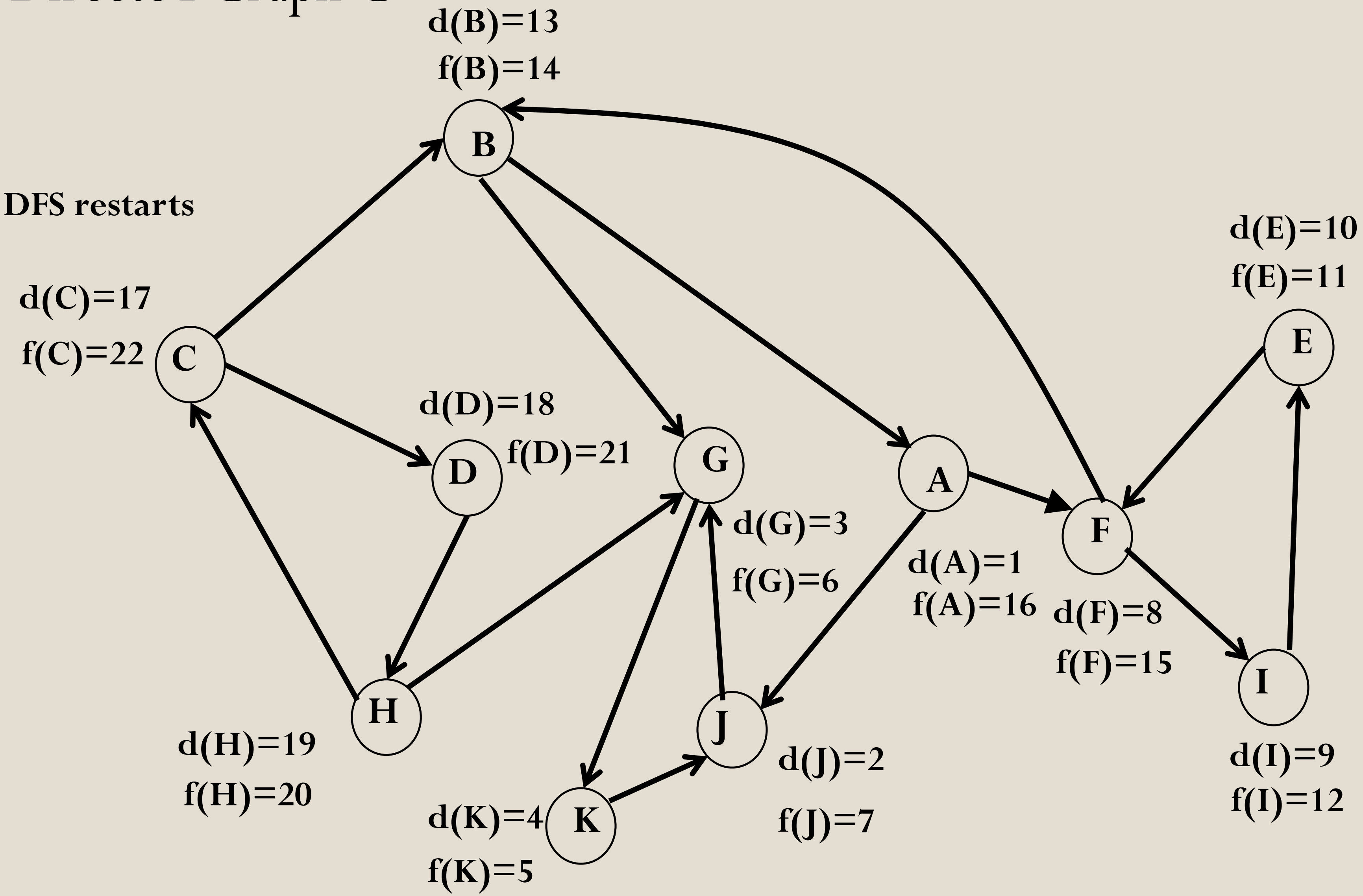
Determine the strongly connected components (stcc) in O(n+m) time

Perform 2 DFS's
On what graphs?

- G^T is the transpose of G generated by reversing the direction of every edge
- G^T and G have the same strongly connected components

Record discovery and finish times

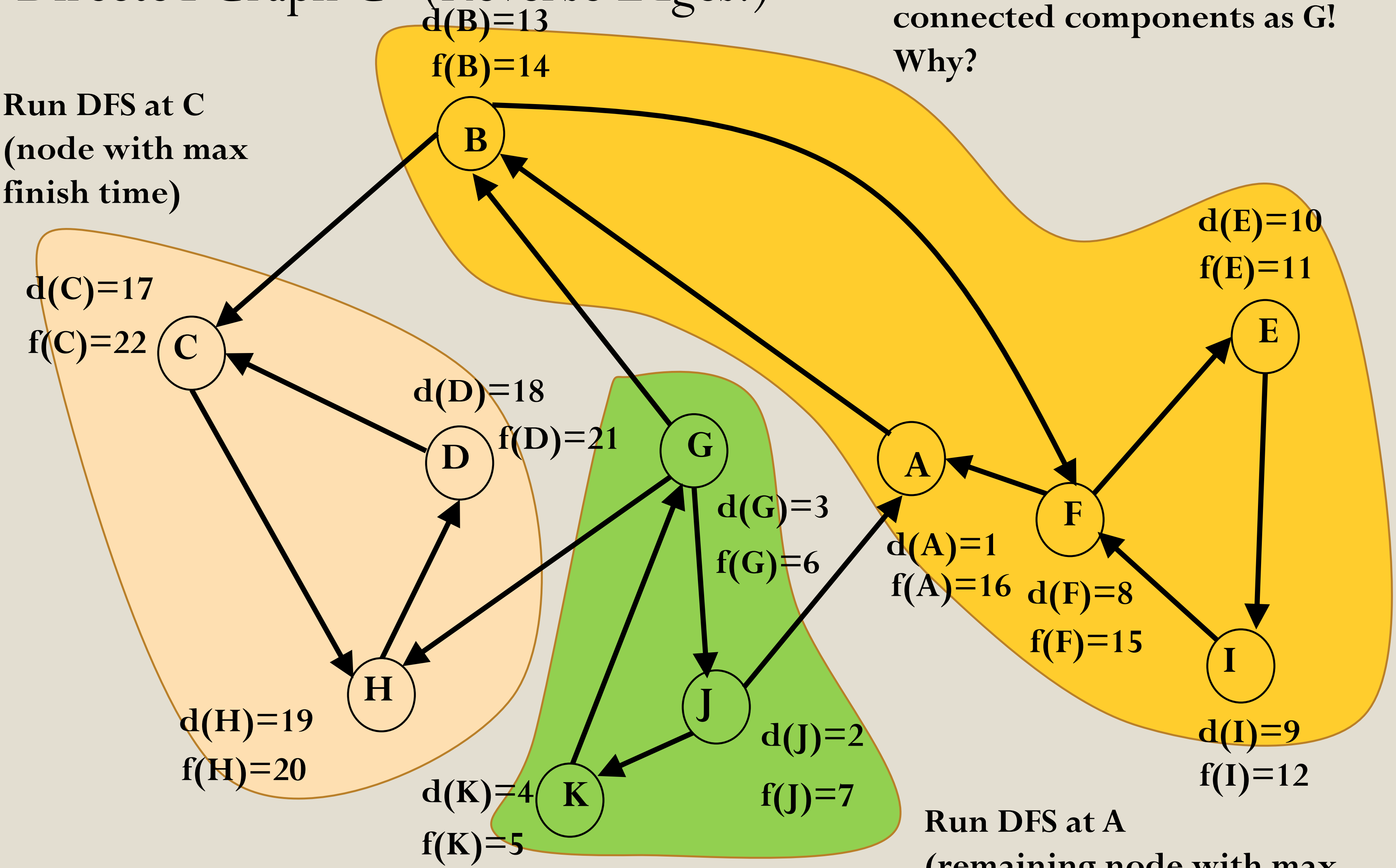
Directed Graph G



Directed Graph G^T (Reverse Edges!)

Claim: Same strongly connected components as G! Why?

Run DFS at C (node with max finish time)



Run DFS at A (remaining node with max finish time)

Run DFS at J (remaining node with max finish time)

Sketch of algorithm finding the stcc

1. call DFS on G to compute f[u] for each vertex u
   - A. Sort nodes in decreasing order of **f[u]**
   - B. (Only requires time O(n) since $1 \le f[u] \le 2n$)
2. find G^T, the transpose of G
3. call DFS on G^T
   - consider the vertices in order of **decreasing f[u]**
4. the second DFS generates one or more tree
   - the vertices in each tree form one strongly connected component

Why does the algorithm find the stcc?

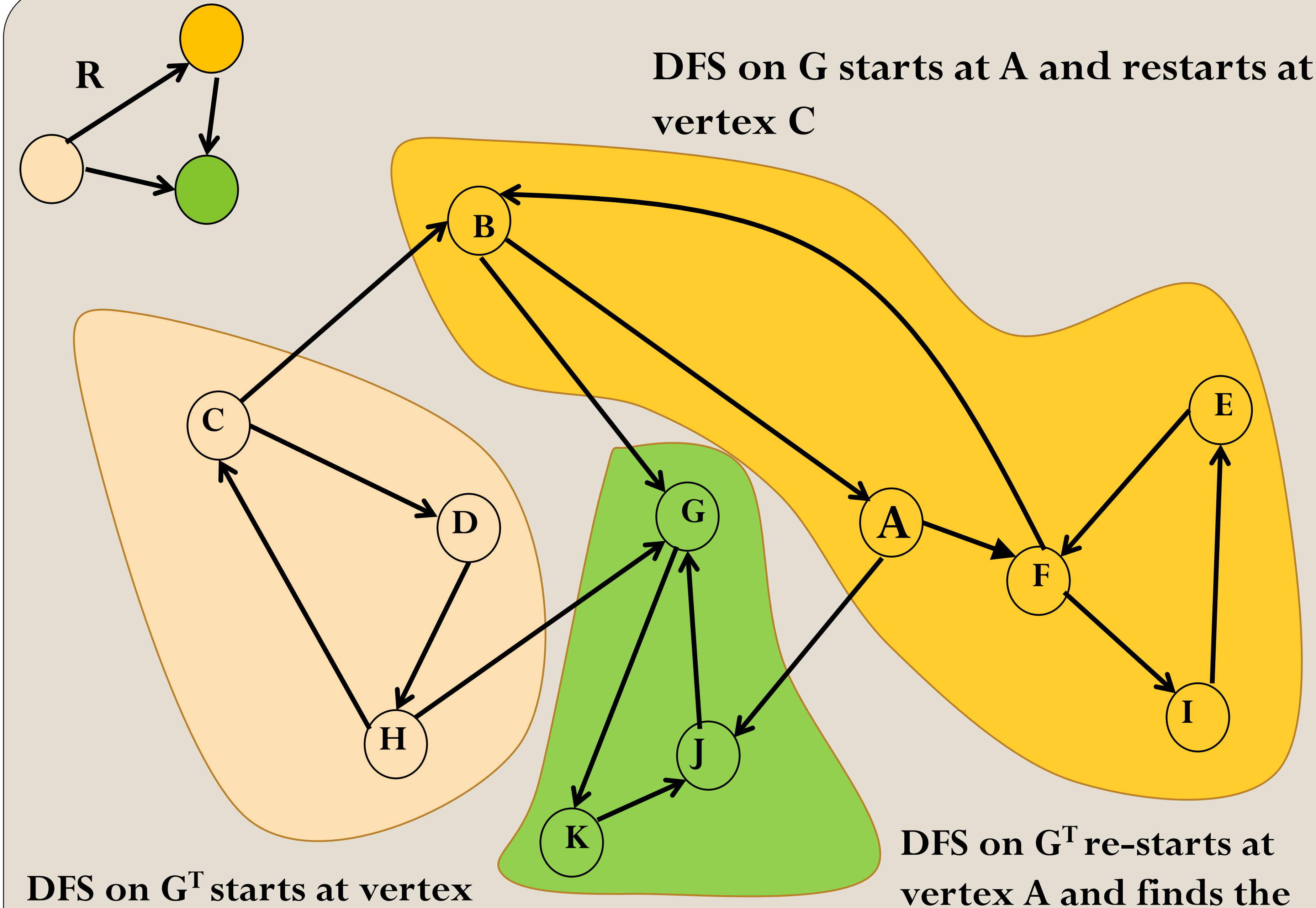
Not obvious.

Create the following “reduced” graph $R=(V_R,E_R)$

- Shrink every stcc into a single vertex.
- Put edges not in a stcc into graph R and remove duplicate edges.

Graph R is a dag

There must exist at least one “vertex” that has no incoming edges and at least one vertex with no outgoing edges.



DFS on G starts at A and restarts at vertex C

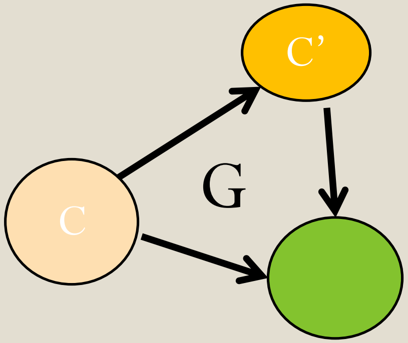
DFS on G^T starts at vertex C and finds the first stcc

DFS on G^T re-starts at vertex A and finds the second stcc

Let U be a set of vertices of directed graph G

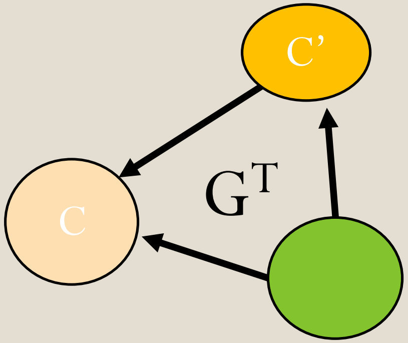
- d(U) is the smallest discovery time of any vertex in U
- f(U) is the largest finishing time of any vertex in U

Assume C and C′ are two strongly connected components of G.



Claim 1: If there is an edge (u, v) in G with u in C and v in C′, then f(C) > f(C′).

Claim 2: If there is an edge (v,u) in the *transpose* of G with v in C’ and u in C, then f(C’) < f(C).



Main Idea - Summary

Second DFS on G^T

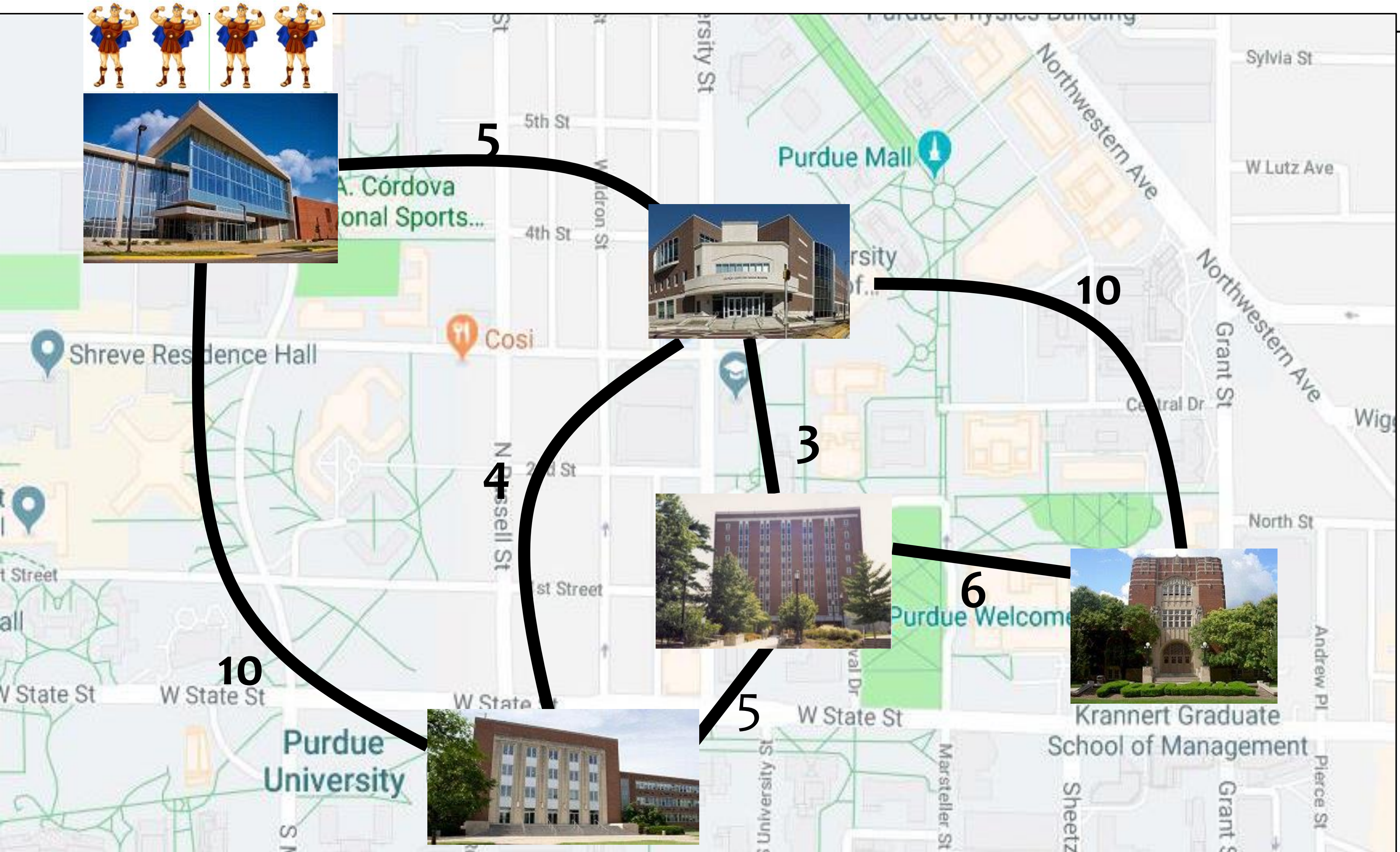
- we start with the component C whose f(C) is the biggest (actually we start with x in C where f(x) is the biggest).
- No edges go from inside C to any other component.
- The tree rooted at x contains exactly the vertices in C and we generated one strongly connected component.

Repeat the argument for the next sink in graph R until all strongly connected components have been generated.

Hence, the strongly connected components can be found in O(n+m) time by doing two DFS's.

Tarjan[72]: One pass is sufficient with "low numbers"

4.4 Shortest Paths in a Graph



Shortest Path Problem

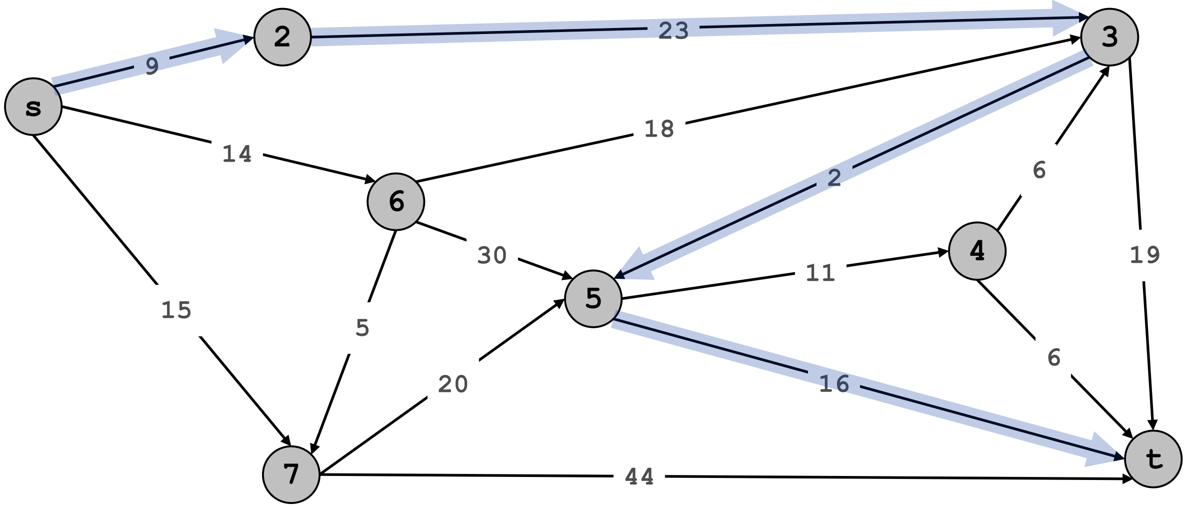
Shortest path network.

- Directed graph G = (V, E).
- Source s, destination t.
- Length ℓ_e = length of edge e.

Shortest path problem: find shortest directed path from s to t.

cost of path = sum of edge costs in path

Cost of path s-2-3-5-t
= 9 + 23 + 2 + 16
= 50.



Dijkstra's Algorithm

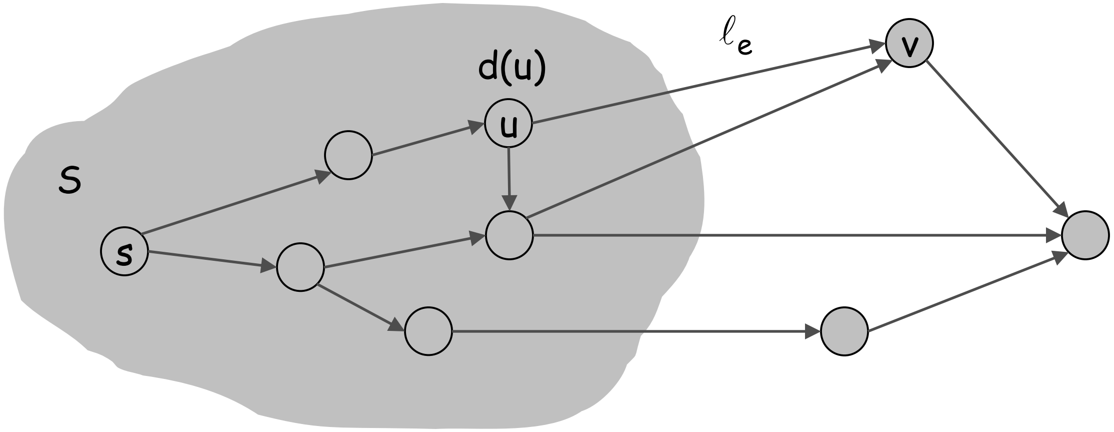
Dijkstra's algorithm (Greedy).

- Maintain a set of explored nodes S for which we have determined the shortest path distance d(u) from s to u.
- Initialize S = { s }, d(s) = 0.
- Repeatedly choose unexplored node v which minimizes

$$\pi(v) = \min_{e = (u,v) : u \in S} d(u) + \ell_e ,$$

  add v to S, and set d(v) = π(v).

shortest path to some u in explored part, followed by a single edge (u, v)



Dijkstra's Algorithm

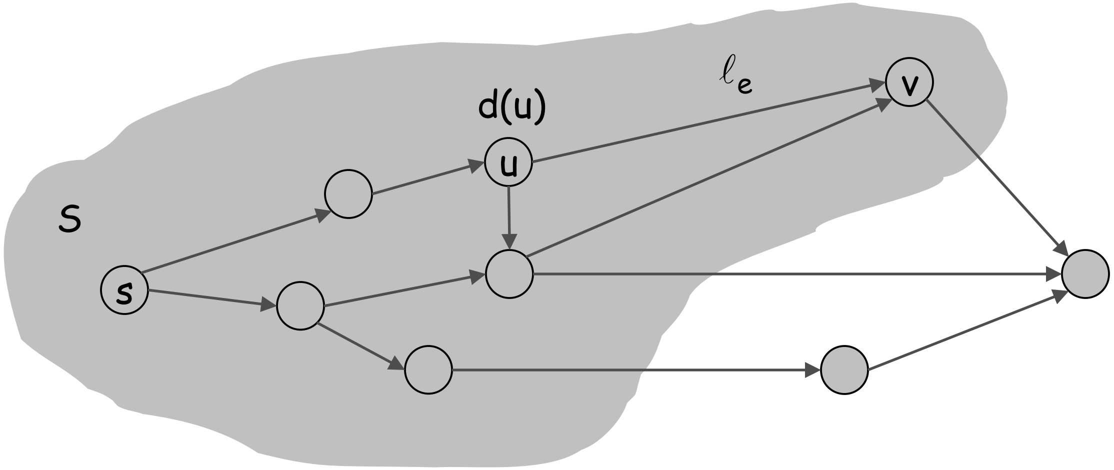
Dijkstra's algorithm.

- Maintain a set of explored nodes S for which we have determined the shortest path distance d(u) from s to u.
- Initialize S = { s }, d(s) = 0.
- Repeatedly choose unexplored node v which minimizes

$$\pi(v) = \min_{e = (u,v) : u \in S} d(u) + \ell_e ,$$

shortest path to some u in explored part, followed by a single edge (u, v)

add v to S, and set d(v) = π(v).



Dijkstra's Algorithm: Proof of Correctness

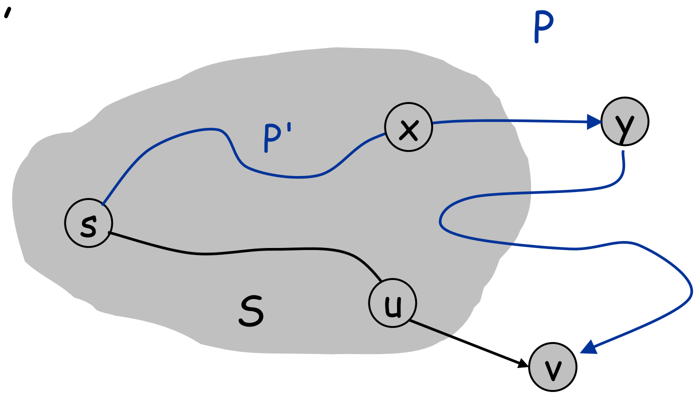
Invariant. For each node $u \in S$, $d(u)$ is the length of the shortest s-u path.

Pf. (by induction on $|S|$)

Base case: $|S| = 1$ is trivial.

Inductive hypothesis: Assume true for $|S| = k \geq 1$.

- Let v be next node added to S, and let u-v be the chosen edge.
- The shortest s-u path plus (u, v) is an s-v path of length $\pi(v)$.
- Consider any s-v path P. We'll see that it's no shorter than $\pi(v)$.
- Let x-y be the first edge in P that leaves S, and let P' be the subpath to x.
- P is already too long as soon as it leaves S.



$$\ell(P) \underset{\substack{\uparrow \\ \text{nonnegative} \\ \text{weights}}}{\geq} \ell(P') + \ell(x,y) \underset{\substack{\uparrow \\ \text{inductive} \\ \text{hypothesis}}}{\geq} d(x) + \ell(x, y) \underset{\substack{\uparrow \\ \text{defn of } \pi(y)}}{\geq} \pi(y) \underset{\substack{\uparrow \\ \text{Dijkstra chose v} \\ \text{instead of y}}}{\geq} \pi(v)$$