Algorithms and Probability

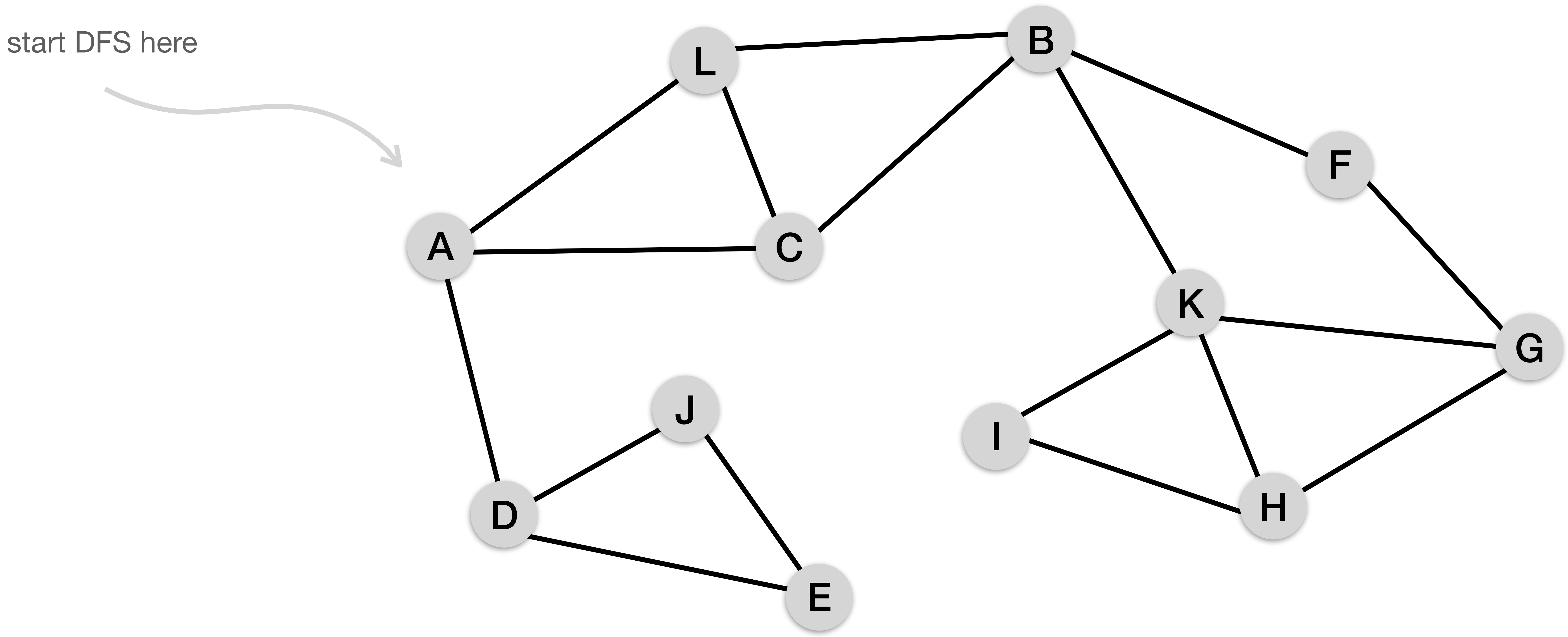
Week 2

2025/02/27 — Georg Hasebe

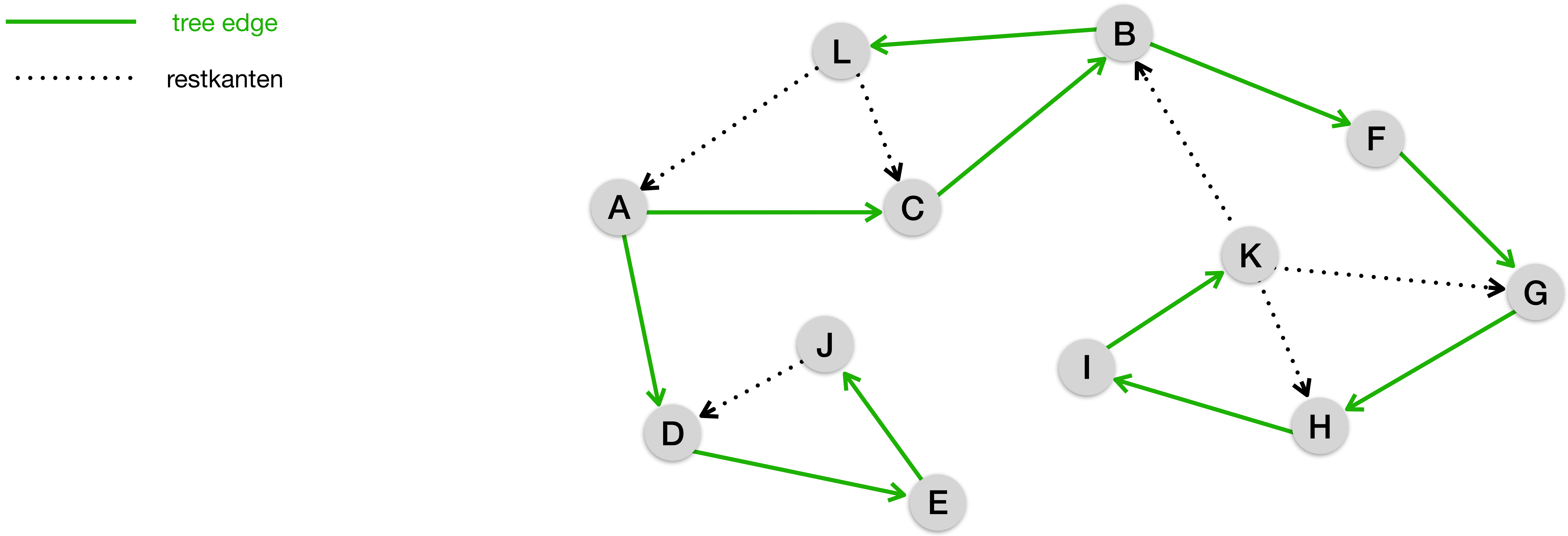
DFS for finding bridges/cut vertices

- Given an undirected graph $G = (V, E)$, find the bridges and cut vertices.
- Naive: remove edges/vertices and check for connectedness.
- As we will see, there is a more efficient approach using DFS.

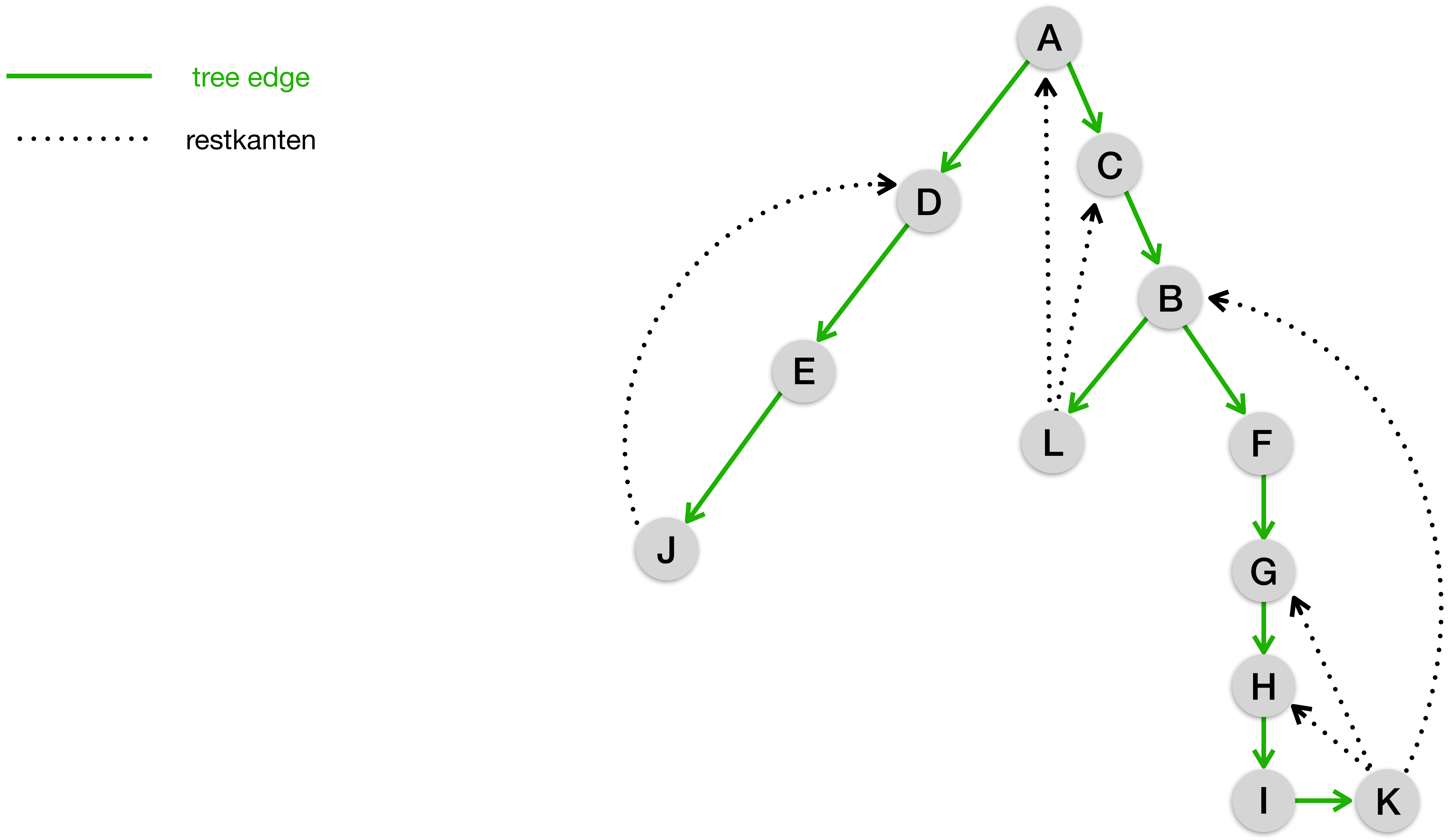
DFS recap



DFS recap

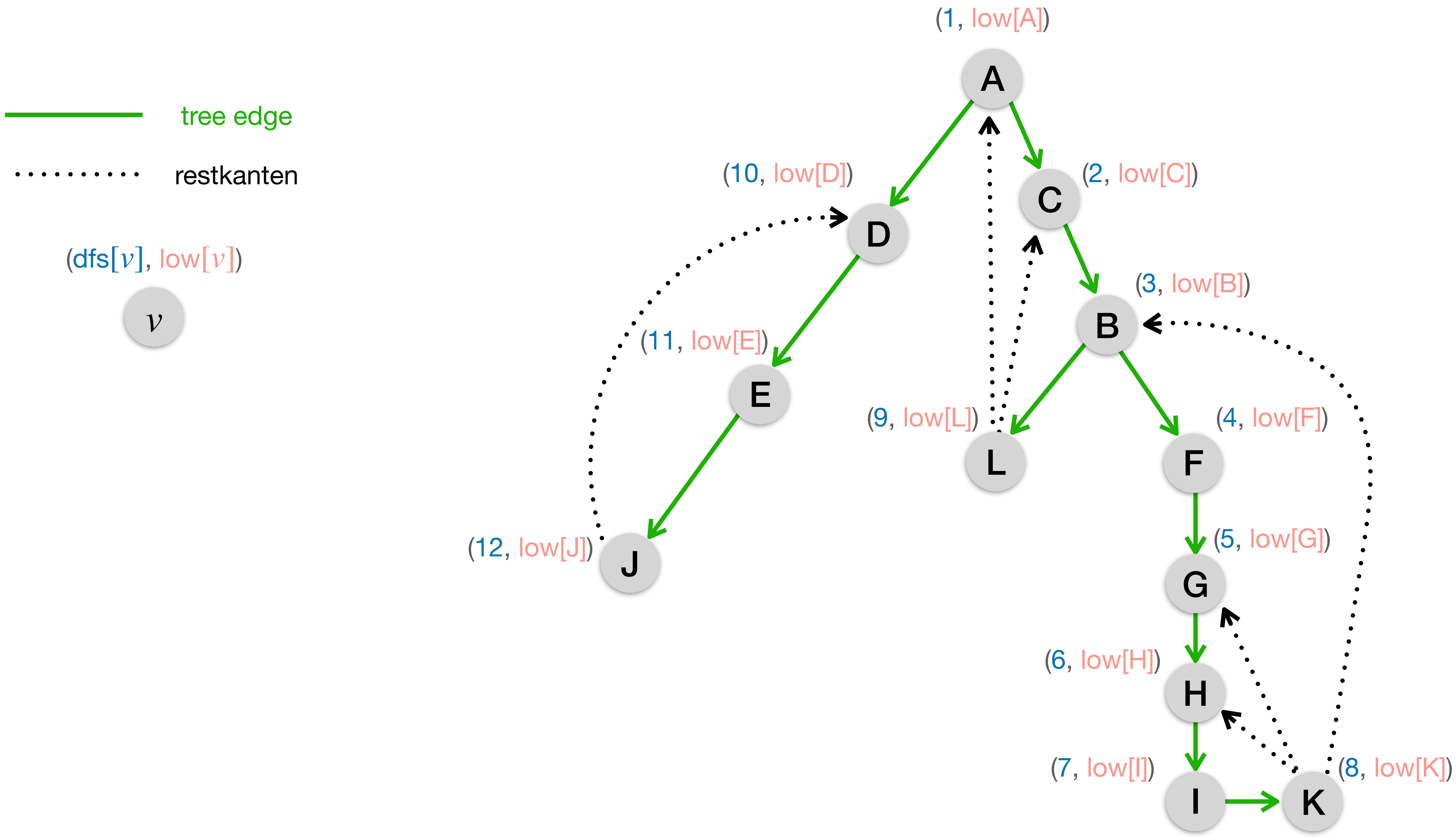


DFS recap

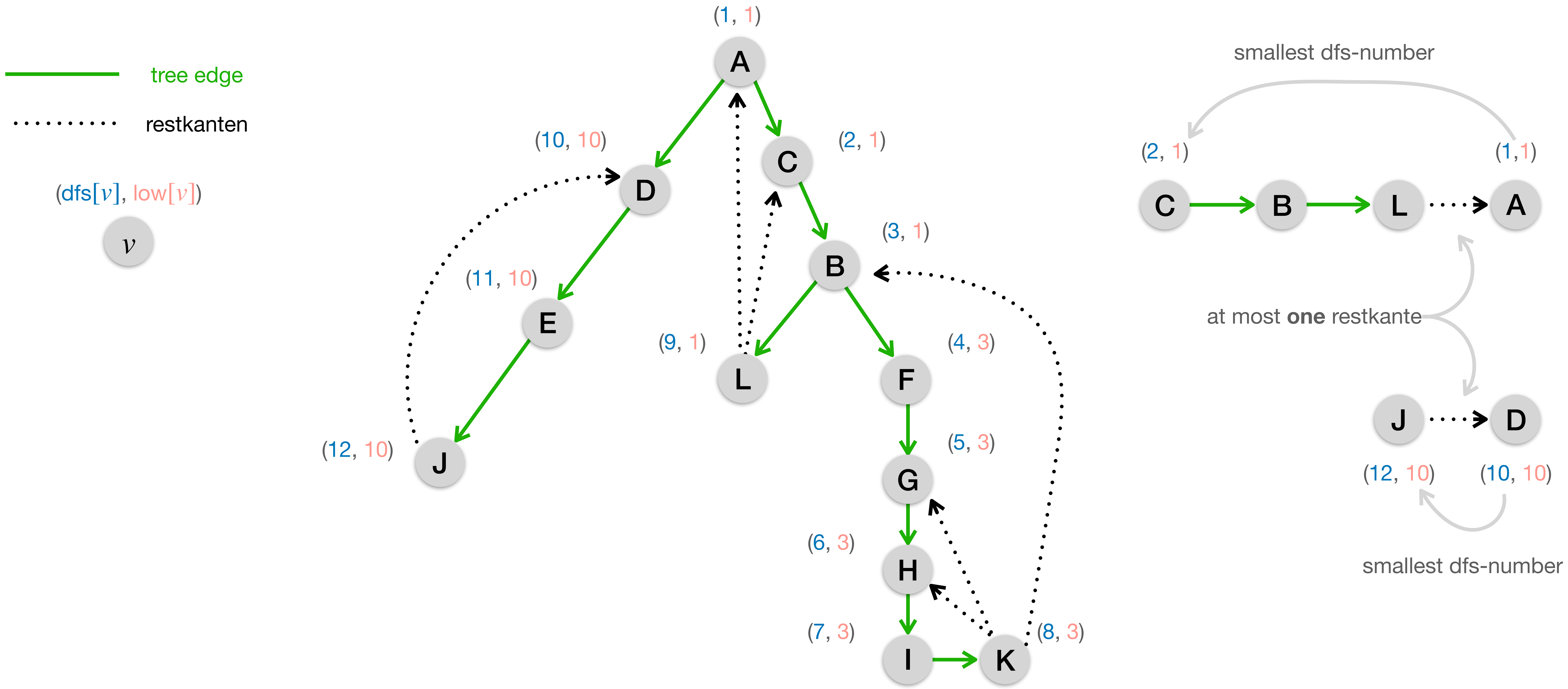


DFS for finding bridges/cut vertices

- We extend traditional DFS by maintaining the following information throughout iteration:
  - dfs$[v]$ … the time DFS "entered" vertex v (dfs$[r] = 1$, where r is the root of the DFS tree).
  - low$[v]$ … the lowest entry time dfs$[w]$ we can reach from v through a directed path consisting of an arbitrary number of tree edges and at most one restkante.



tree edge
restkanten
(dfs[v], low[v])
v
(1, low[A])
A
(10, low[D])
D
(2, low[C])
C
(3, low[B])
B
(11, low[E])
E
(9, low[L])
L
(4, low[F])
F
(12, low[J])
J
(5, low[G])
G
(6, low[H])
H
(7, low[I])
I
(8, low[K])
K


tree edge
restkanten
(dfs[v], low[v])
v
(1, 1)
A
(10, 10)
D
(2, 1)
C
(3, 1)
B
(11, 10)
E
(9, 1)
L
(4, 3)
F
(12, 10)
J
(5, 3)
G
(6, 3)
H
(7, 3)
I
(8, 3)
K
smallest dfs-number
(2, 1)
(1,1)
C
B
L
A
at most **one** restkante
J
D
(12, 10)
(10, 10)
smallest dfs-number

Proof

Let $v \in V$ such that v is **not the root** of the DFS tree.

We show that v is a cut vertex if and only if v has a neighbor u in the DFS tree T such that low$[u] \geq$ dfs$[v]$.

$(\Rightarrow)$

Proof

Assume that v is a cut vertex. Then $G[V \setminus \{v\}]$ has at least 2 connected components Z_1 and Z_2. Without loss of generality, assume $s \in Z_1$.

Every path from s to a vertex in Z_2 must include v, we have
$1 = \text{dfs}[s] < \text{dfs}[v] < \text{dfs}[w] \quad \forall w \in Z_2$.

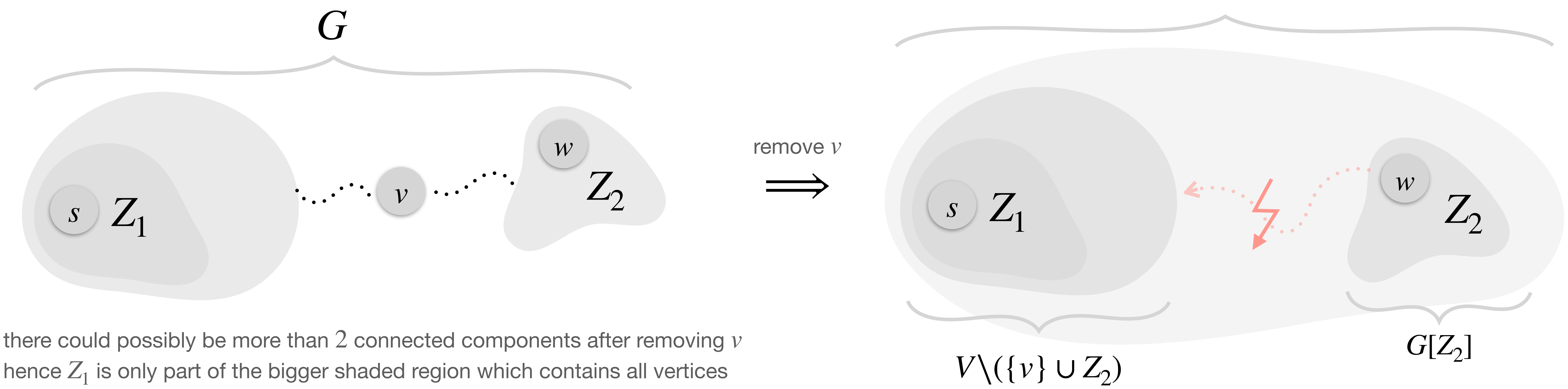
Since $G[Z_2]$ is a connected component in $G[V \setminus \{v\}]$, there cannot be an edge from $w \in Z_2$ to a vertex $u \in V \setminus (\{v\} \cup Z_2)$.

($\Rightarrow$)

Proof

…

Since $G[Z_2]$ is a connected component in $G[V\backslash\{v\}]$, there cannot be an edge from $w \in Z_2$ to a vertex $u \in V\backslash(\{v\} \cup Z_2)$.



there could possibly be more than 2 connected components after removing v hence Z_1 is only part of the bigger shaded region which contains all vertices that are not in Z_2 or v itself.

if there was such an edge, then u would be connected to Z_2 and therefore element of Z_2 but $V\backslash(\{v\} \cup Z_2)$ doesn't contain vertices from Z_2.

Proof

$(\Rightarrow)$

…

Since $G[Z_2]$ is a connected component in $G[V \setminus \{v\}]$, there cannot be an edge from $w \in Z_2$ to a vertex in $u \in V \setminus (\{v\} \cup Z_2)$.

Thus low$[w]$ is at least dfs$[v]$ for all $w \in Z_2$. Since v is connected to Z_2, it has at least one neighbor $w \in Z_2$ such that low$[w] \geq$ dfs$[v]$.

DiskMath Recap

Contraposition:

$P \to Q$ is equivalent to $\neg Q \to \neg P$

Here:

v is a cut vertex $\Leftarrow$ v has a neighbor u such that low$[u] \geq$ dfs$[v]$

$\Leftrightarrow$

v is **not** a cut vertex $\Rightarrow$ v **has no** neighbor u such that low$[u] \geq$ dfs$[v]$

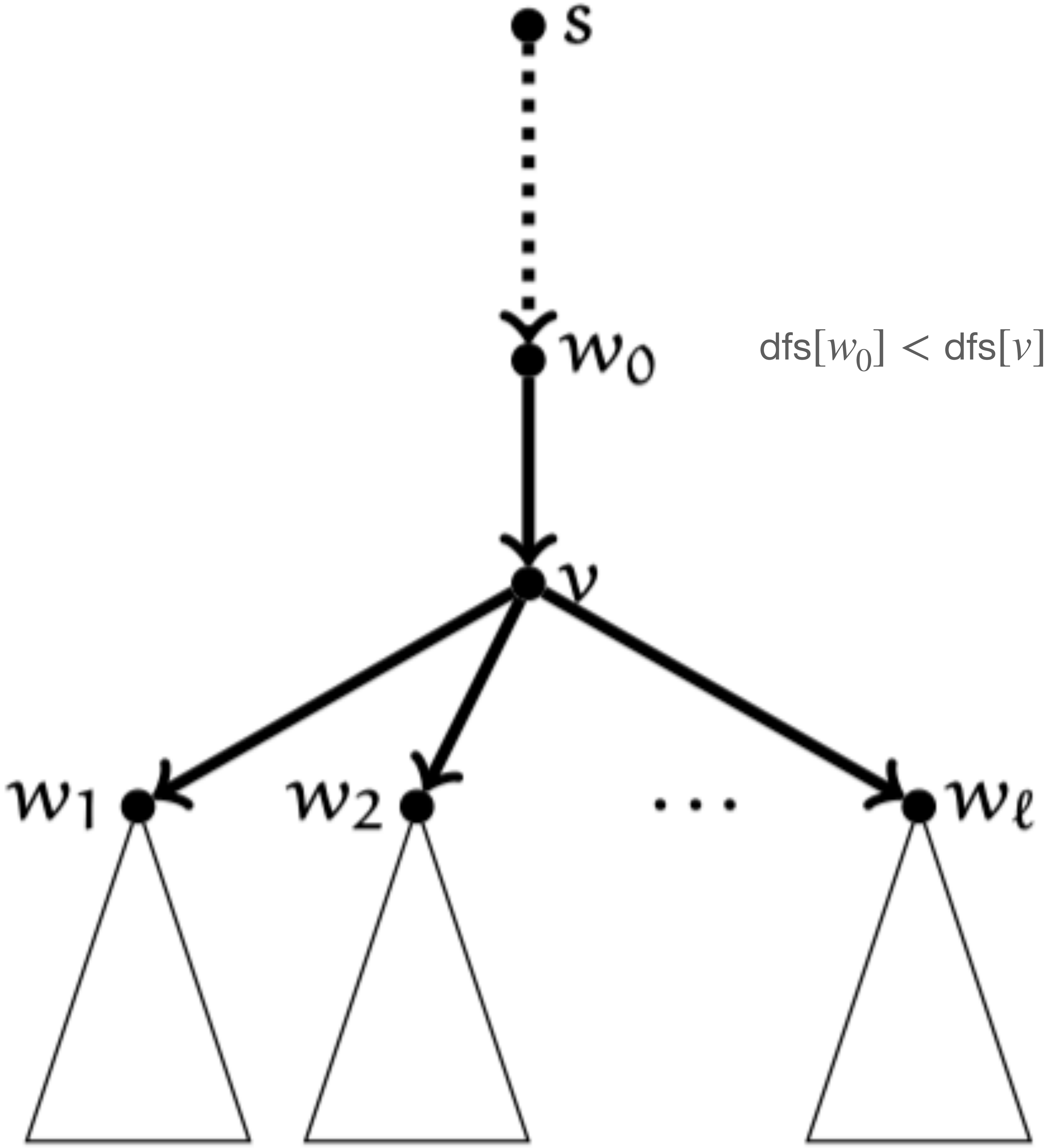
$(\Leftarrow)$

Proof

Assume v is not a cut vertex.

Let T be the DFS tree rooted at s and let $w_0, \ldots, w_l$ be all the neighbors of v in G. Without loss of generality, assume dfs$[w_0] <$ dfs$[v]$.

Proof



dfs$[w_0] <$ dfs$[v]$ since we reach w_0 earlier from s than v.

by construction of the DFS algorithm, the subtrees rooted at $w_1, \ldots, w_l$ cannot be connected.

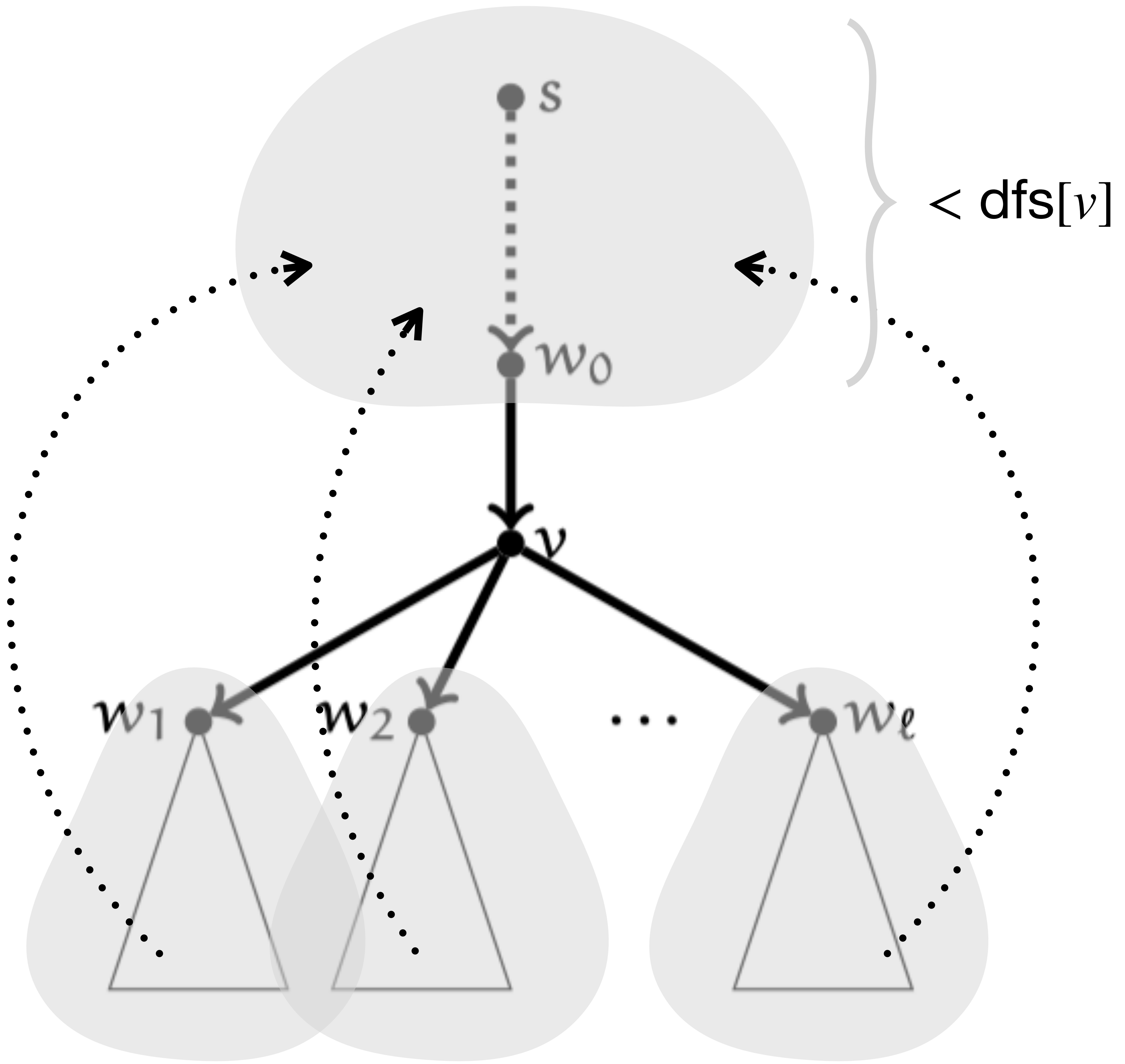
but they have to be connected in G, otherwise v would be a cut vertex.

$(\Leftarrow)$

Proof

…

For every neighbor $w_1, \ldots, w_l$ there exists a path using a restkante to a vertex with smaller dfs-number.



using contraposition

$(\Leftarrow)$

Proof

…

For every neighbor $w_1, \ldots, w_l$ there exists a path using a restkante to a vertex with smaller dfs-number.

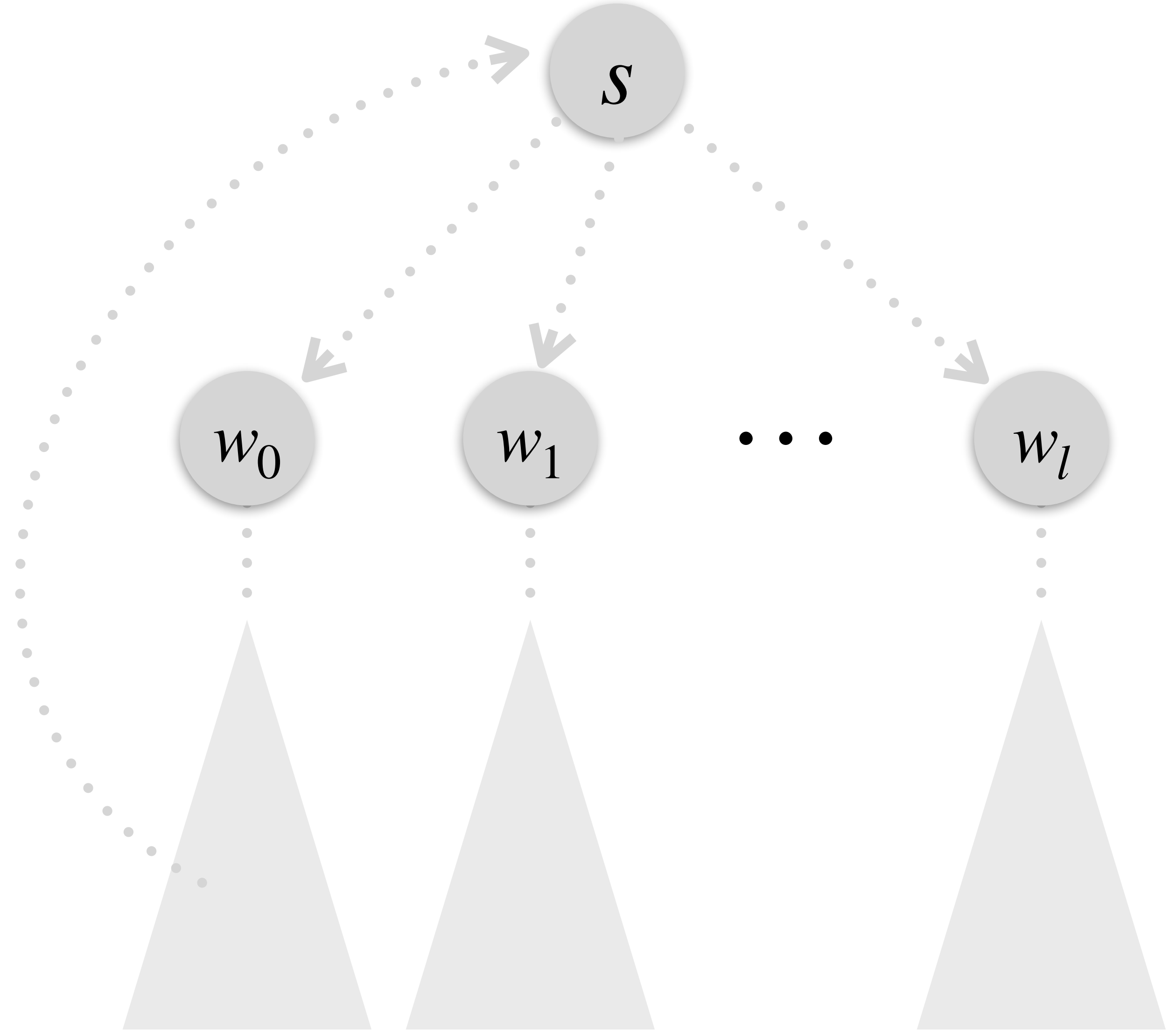
Thus low$[w]$ is greater than dfs$[v]$ for all neighbors w of v, meaning that there is no neighbor w of v such that low$[v] \geq$ dfs$[u]$.

What about the root?

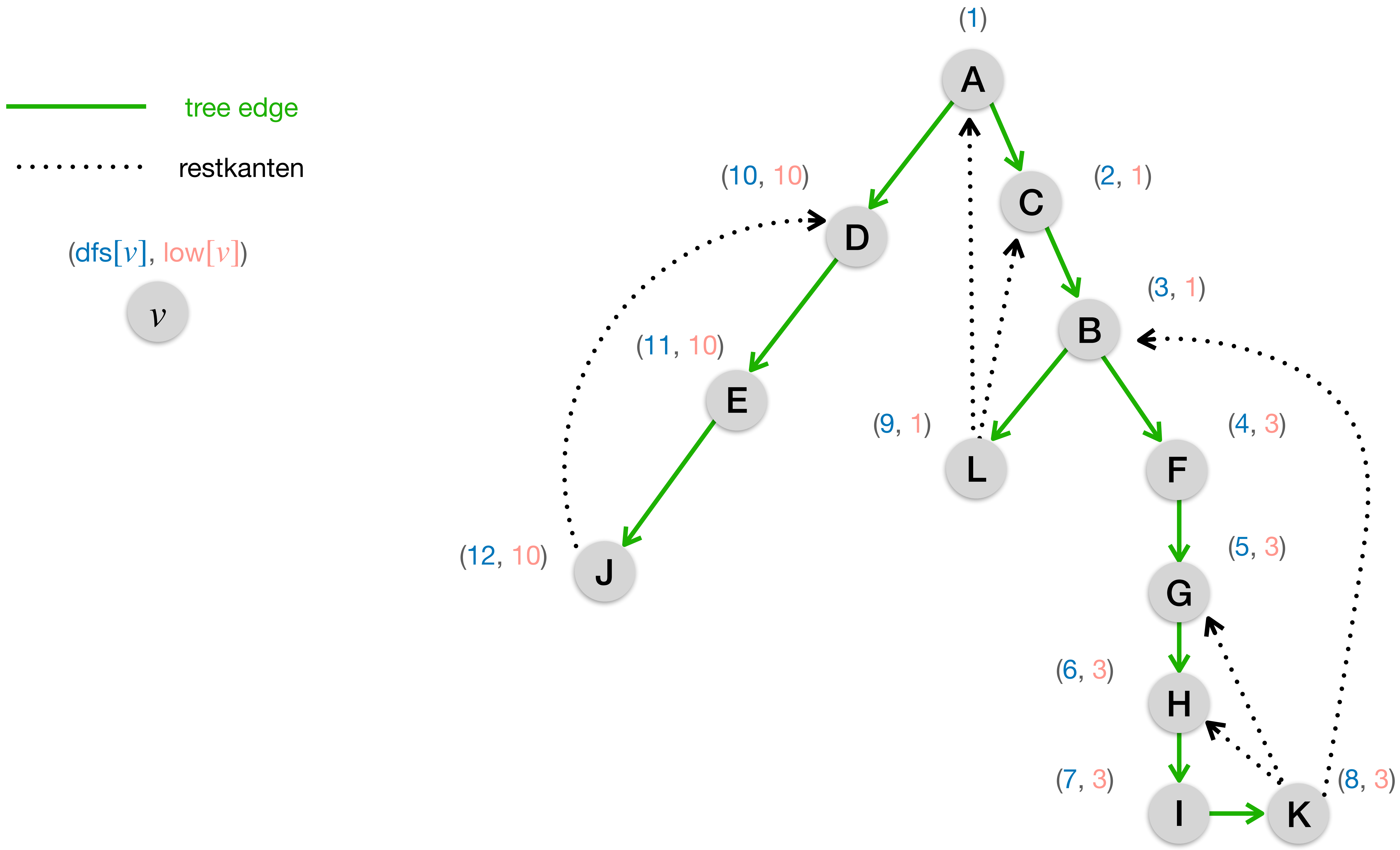
Let T be the DFS tree **rooted at** s. If $\deg(s) \geq 2$ then s is a cut vertex.

Proof. Assume $\deg(s) = l \geq 2$. By construction of the DFS algorithm, the subtrees rooted at $w_1, \ldots, w_l$ cannot be connected. Even if there was a restkante from a subtree to s, after removing s the vertices contained in the subtrees become disconnected in $G[V \backslash \{s\}]$.

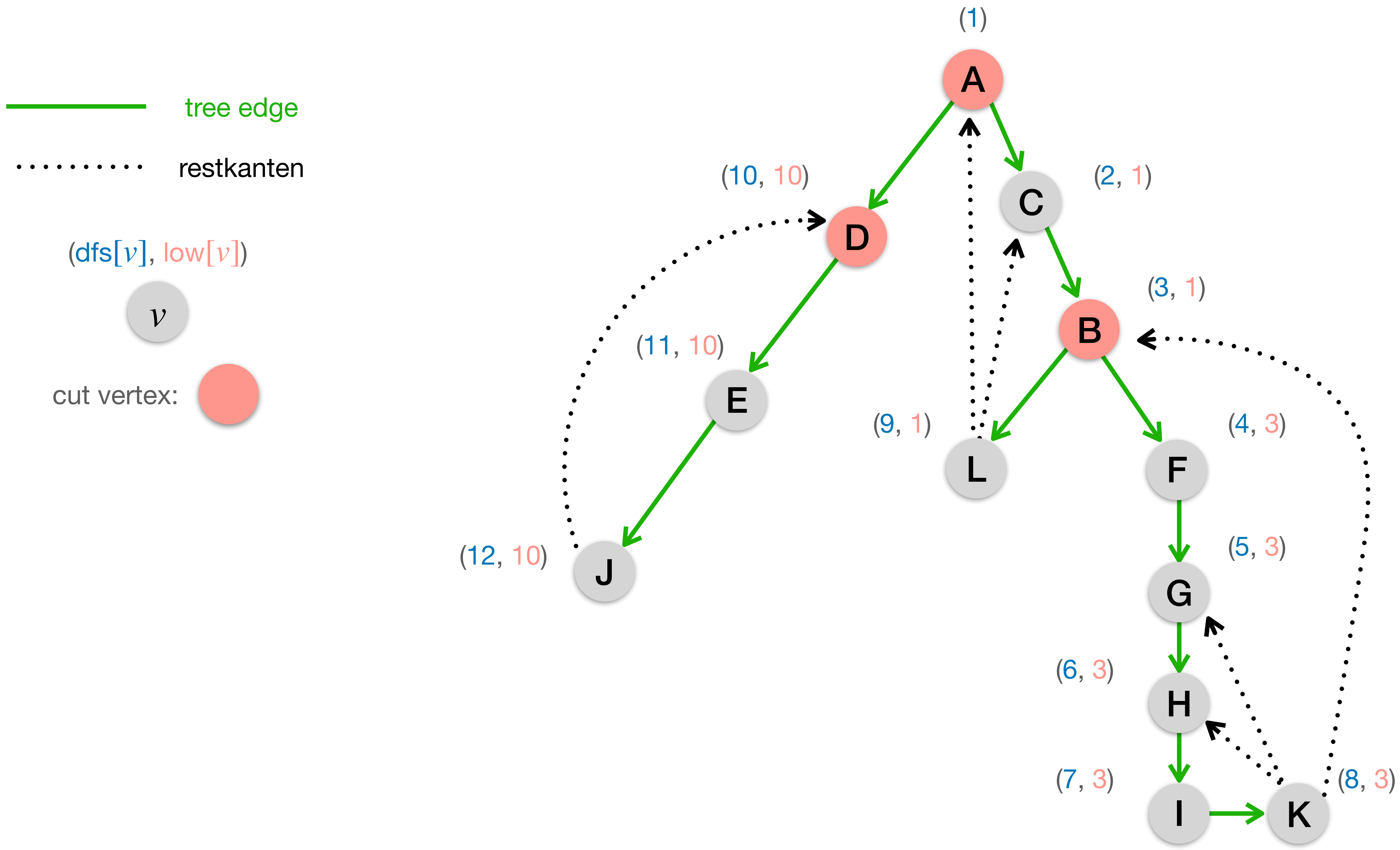
Thus s is a cut vertex.



Identify the cut vertices



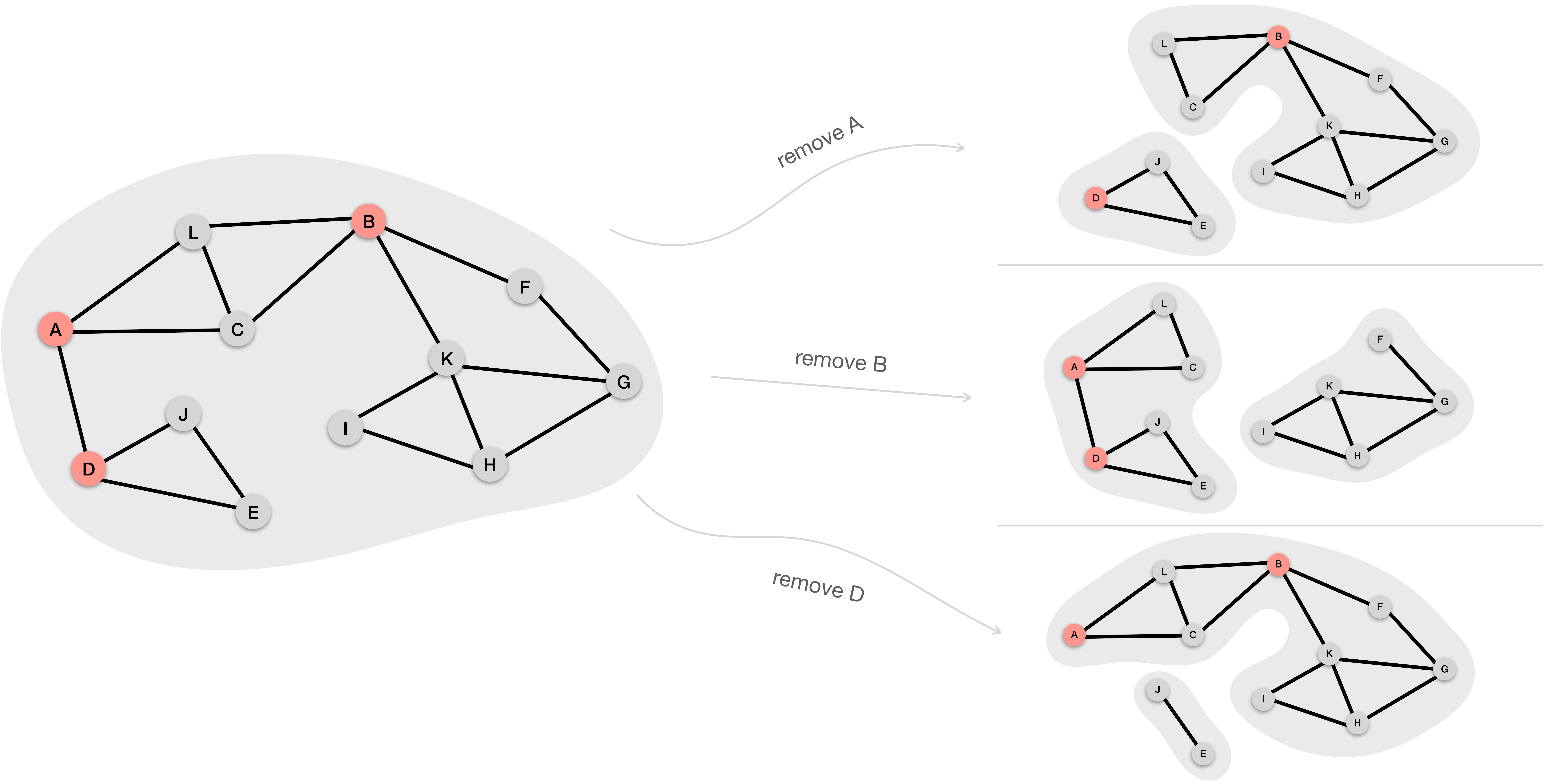
Identify the cut vertices



A — since it is the root and $\deg(A) \geq 2$ in the DFS tree.

D — since E is a neighbor of D and low$[E] = 10 \geq$ dfs$[D] = 10$.

B — since F is a neighbor of B and low$[F] = 3 \geq$ dfs$[D] = 3$.



A
B
C
D
E
F
G
H
I
J
K
L
remove A
remove B
remove D

Pseudocode

```
DFS-Visit(G, v)
num ← num + 1
dfs[v] ← num
low[v] ← dfs[v]
isArtVert[v] ← FALSE
for all {v, w} ∈ E do
    if dfs[w] = 0 then
        T ← T + {v, w}
        val ← DFS-Visit(G, w)
        if val ≥ dfs[v] then
            isArtVert[v] ← TRUE
        low[v] ← min{low[v], val}
    else dfs[w] ≠ 0 and {v, w} ∉ T
        low[v] ← min{low[v], dfs[w]}
return low[v]
```

```
DFS(G, s)
∀v ∈ V :    dfs[v] ← 0
num ← 0
T ← ∅
DFS-Visit(G, s)
if s hat in T Grad mindestens zwei then
    isArtVert[s] ← TRUE
else
    isArtVert[s] ← FALSE
```

Kapitel 1 — Graphentheorie, p. 39

Result (cut vertices)

Satz 1.27. Für zusammenhängende Graphen $G = (V, E)$, die mit Adjazenzlisten gespeichert sind, kann man in Zeit $O(|E|)$ alle Artikulationsknoten berechnen.

Note that DFS normally runs in $O(|V| + |E|)$ but since we assume G is connected, we know that $|E| \geq |V| - 1$ thus $|V| + |E| \leq 2 \cdot |E| \leq O(|E|)$.

What about bridges?

- First, notice that if G (connected) contains a bridge $e \in E$, any spanning tree of G must contain e.
- Hence the DFS tree must contain e, as it is a spanning tree of G.
- We reuse our Lemma from earlier:

Lemma: Let $G = (V, E)$ be a connected graph. If $\{u, v\} \in E$ is a bridge, then u and v are cut vertices unless they have degree 1.

What about bridges?

Lemma: Let $G = (V, E)$ be a connected graph. If $\{u, v\} \in E$ is a bridge, then u and v are cut vertices unless they have degree 1.

Let $e = (v, w)$ be an edge in the DFS tree T, then e is a bridge if and only if low$[w] >$ dfs$[v]$.

On finding Hamiltonian cycles

- Finding Hamiltonian cycles is **hard** (NP-hard)
- The only known algorithms are **exponential**
- **Naive:** try out all possibilities for a Hamiltonian cycle. How many?
  - At most $(n-1)!/2$. Why?

DP Algorithm

$G = (V, E)$, $V = [n] = \{1,2,\ldots,n\}$.

Let $S \subseteq V$ where $1 \in S$. Consider the following notation for all $x \in S, x \neq 1$

$$P_{S,x} = \begin{cases} 1, & \text{there exists a } 1\text{-}x \text{ path in } G \text{ that contains all vertices in } S \\ 0, & \text{otherwise.} \end{cases}$$

Now if there exists some $x \in N(1)$ where $P_{[n],x} = 1$, G contains a Hamiltonian cycle.

We can calculate the values for $P_{S,x}$ using **dynamic programming**.

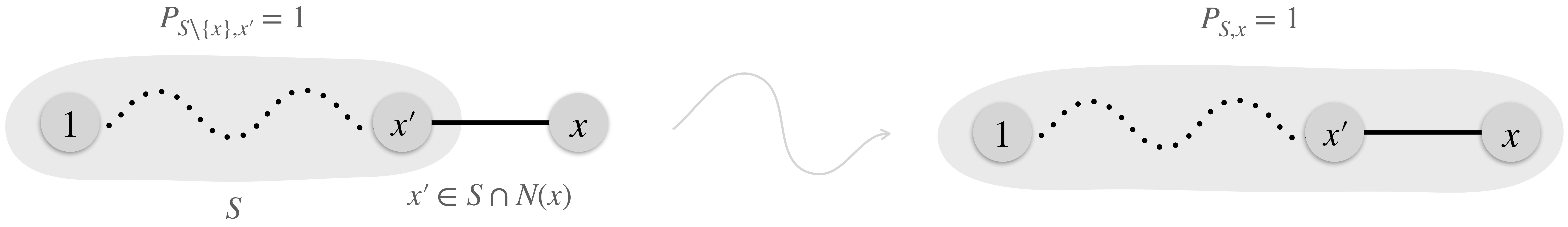
DP Algorithm

$G = (V, E)$, $V = [n] = \{1, 2, \ldots, n\}$.

Base cases: If $S = \{1, x\}$ for some $x \in V$, then $P_{S,x} = 1$ if $\{1, x\} \in E$.

Recursion:

$$P_{S,x} = \max\{P_{S \setminus \{x\}, x'} \mid x' \in S \cap N(x), x' \neq 1\}$$



Pseudocode

HAMILTONKREIS $(G = ([n], E))$

// *Initialisierung*
for all $x \in [n], x \neq 1$ **do**
    $P_{\{1,x\},x} := \begin{cases} 1, & \text{falls } \{1,x\} \in E \\ 0, & \text{sonst} \end{cases}$
// *Rekursion*
for all $s = 3$ **to** n **do**
    for all $S \subseteq [n]$ mit $1 \in S$ und $|S| = s$ **do**
        for all $x \in S, x \neq 1$ **do**
            $P_{S,x} = \max\{P_{S\setminus\{x\},x'} \mid x' \in S \cap N(x), x' \neq 1\}.$
// *Ausgabe*
if $\exists\, x \in N(1)$ mit $P_{[n],x} = 1$ **then**
    return G enthält Hamiltonkreis
else
    return G enthält keinen Hamiltonkreis

The initialization part covers all subsets of size 2. We therefore start with subsets of size 3 and work our way up to n.

We go through all subsets of size s. If $s = n$, then the only subset will be $[n]$ itself. Remember, we try to determine $P_{[n],x}$.

We demand that $x \neq 1$, because we started with 1 already.

$S\setminus\{x\}$ ensures that any path that we extend by x does not contain x.

Here we attempt to “close” a Hamiltonian 1-x path in order to get a Hamiltonian cycle.

Result

Satz 1.34. Algorithmus HAMILTONKREIS ist korrekt und benötigt Speicher $O(n \cdot 2^n)$ und Laufzeit $O(n^2 \cdot 2^n)$, wobei $n = |V|$.

Satz 1.34. Algorithmus HAMILTONKREIS ist korrekt und benötigt Speicher $O(n \cdot 2^n)$ und Laufzeit $O(n^2 \cdot 2^n)$, wobei $n = |V|$.

Proof

```
HAMILTONKREIS (G = ([n], E))
// Initialisierung
for all x ∈ [n], x ≠ 1 do
    P_{1,x},x := { 1, falls {1, x} ∈ E
                 { 0, sonst
// Rekursion
for all s = 3 to n do
    for all S ⊆ [n] mit 1 ∈ S und |S| = s do
        for all x ∈ S, x ≠ 1 do
            P_S,x = max{P_S\{x},x' | x' ∈ S ∩ N(x), x' ≠ 1}.
// Ausgabe
if ∃ x ∈ N(1) mit P_[n],x = 1 then
    return G enthält Hamiltonkreis
else
    return G enthält keinen Hamiltonkreis
```

$$\sum_{s=3}^{n} \sum_{S \subseteq [n], 1 \in S, |S|=s} \sum_{x \in S, x \neq 1} O(???)$$

$$= \sum_{s=3}^{n} \sum_{S \subseteq [n], 1 \in S, |S|=s} \sum_{x \in S, x \neq 1} O(n)$$

$$= \sum_{s=3}^{n} \binom{n-1}{s-1} (s-1) O(n)$$

$$\leq O(n^2 \cdot 2^n)$$

$|S| = s$, but we exclude 1.

A subset of S' of size $s-1$ from $n-1$ vertices; then $S = S' \cup \{1\}$.

(*) where we used $\sum_{s=0}^{n-1} \binom{n-1}{s} = 2^{n-1}$.