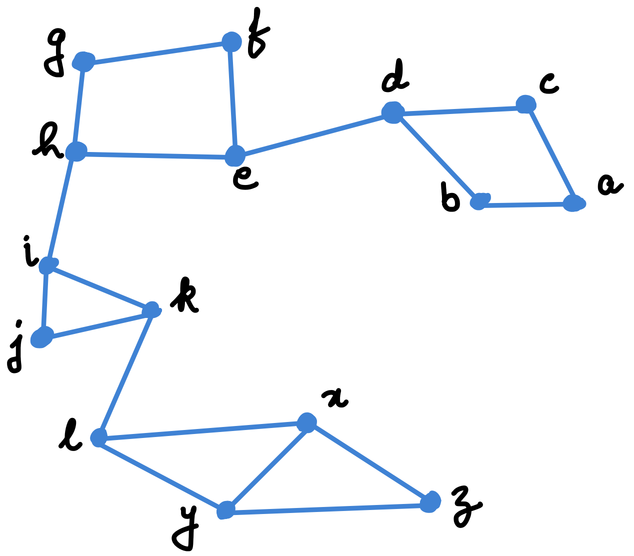
Lecture 09

DFS Application: Finding ALL bridges

Bridge Edge:

An edge ***(x,y)*** is said to be a bridge edge in *G* if ***x*** and ***y*** are disconnected in ***G\(x,y)***.



(d,c)
(i,h)
(k,l)
Three bridges

Check (x,y) is bridge edge or not

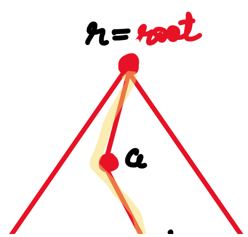
Trivial way – Is x,y dis-connected in G\(x,y)?

Naive – $O(\underbrace{m}_{\text{edges}} * \underbrace{m}_{\text{time per edge}})$

Ancestors of x in a tree T:

All vertices lying on root to ***x*** path in T, including both root and ***x*** .

r, a, b, c, x



Proper-Ancestors of x in a tree T:
Ancestors of x in T other than itself.

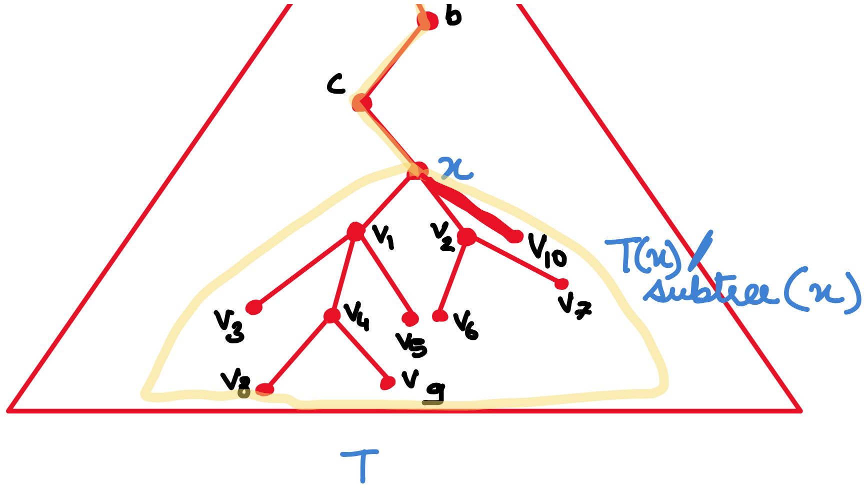
r, a, b, c

Descendants of x in a tree T:
All vertices lying in subtree of x , including the vertex x .

$x, v_1, \ldots, v_{10}$

Proper-Descendants of x in a tree T:
Descendants of x in T other than itself.

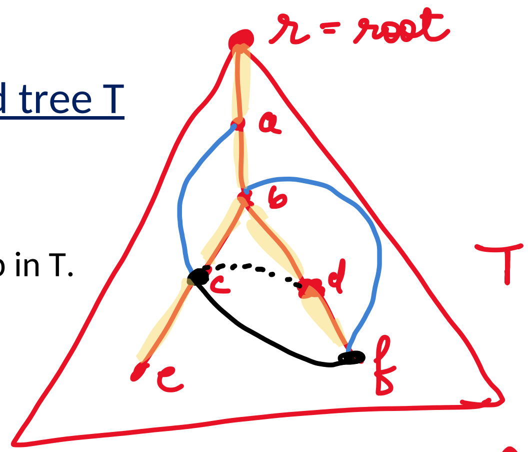
$v_1 \ldots v_{10}$



Classification of edges of undirected G with respect to Arbitrary rooted tree T

- ~~Tree edges~~ - Edges parts of tree (6 edges)
- ~~Back edges~~ - Non tree Edges whose endpoint have ancestor descendant relationship in T.
  (a, c) and (b, f)
- ~~Cross edges~~ - Edges whose endpoint have NO ancestor descendant relationship.
  (c, f) and (c, d)

Partition of edges of G.



10 edges

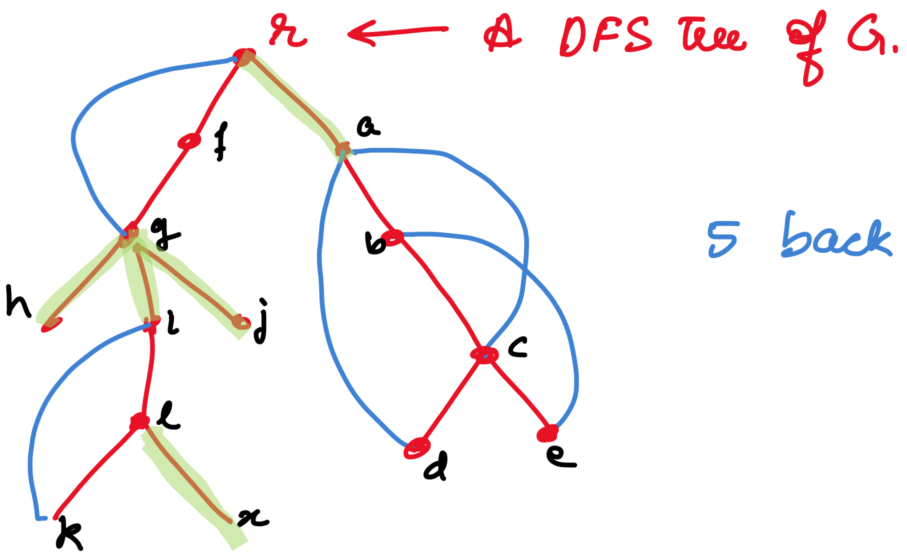
$G = (V, E)$

r, a, b, c, d, e, f

(r,a) (a,b) (b,c), (b,d)

⋮

Lemma (last class): *Let T be a DFS tree of G=(V,E), then all non-tree edges in G are back edges.*



5 back edges.

Ques 1: If (x,y) is bridge edge in a connected graph G. Then (x,y) is a tree-edge

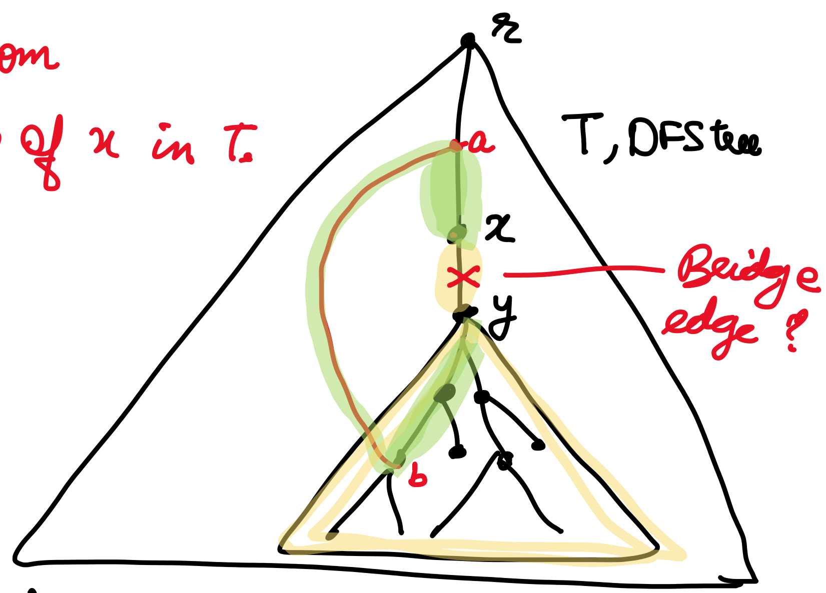
Proof: There is a path from x to y in tree, and we know by def^n of bridge such a path is just edge (x,y).

Ques 2: A tree edge (x,y), with $x = parent(y,T)$, we have (x,y) is bridge edge if

" There is no back edge from T(y) to ancestors of x"

There is no back edge from T(y) to ancestors of x

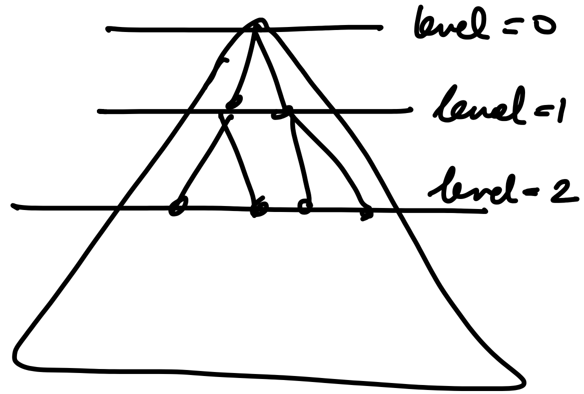
Proof: (i) If there is a back edge (b,a) from $b \in$ subtree(x) to 'a' – ancestor of x in T.
Then (x,y) is not a bridge edge.



Proof of (i)

tree path(x,a) • (a,b) • treepath(b,y)

is a path from x to y in $G \backslash (x,y)$.

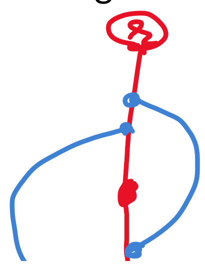


(ii) Converse

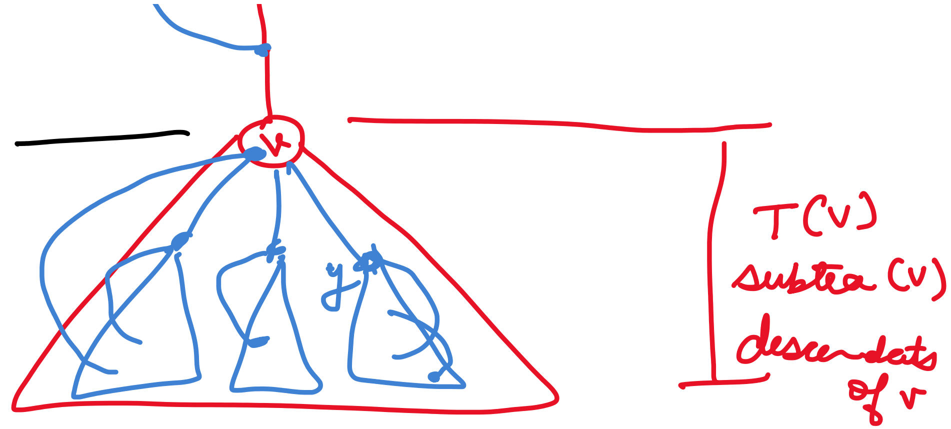
H.W.

High-point(*v*):

The level of the *highest ancestor* of ***v*** to which there is a back edge from descendants of ***v*** (is such a back edge exists).
Otherwise set it to be just level(v).

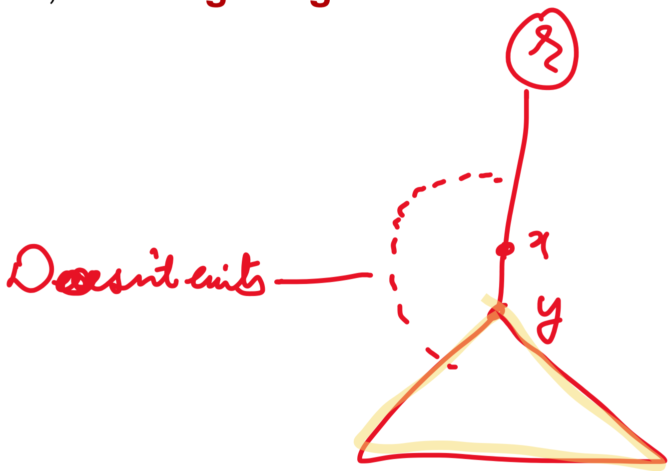


High point(v) = level(v).



Theorem : A tree edge (***x*,*y***), with x being parent of y in DFS tree, is a **bridge edge iff**

High-point(***y***) = **Level**(***y***).



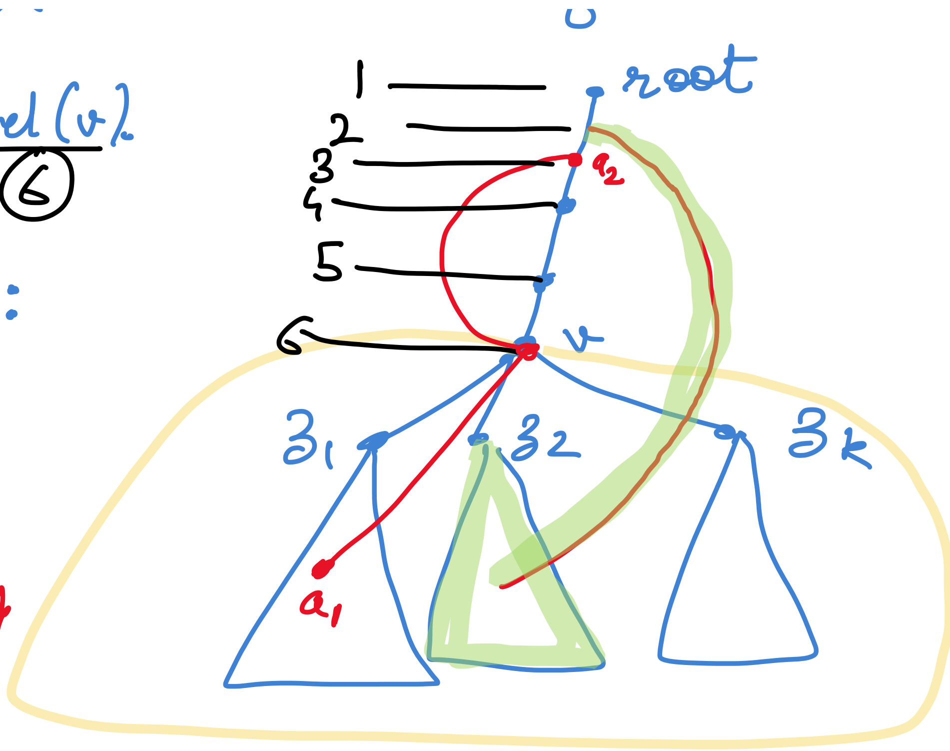
$\underbrace{\text{level}(v),\ \forall v}$

can be computed in $O(n)$ time if we have already DFS tree.

Ques: How to compute High-point for all vertices.

Take a vertex v. Let $z_1 \dots z_k$ be children of v.

① Initialize High-point(v) to be level(v).
⑥



② Scan all non-tree edges (a,v):

if level(a) < level(v).

then high-point(v) =

min{highpoint(v), level(a)}

③ For $i = 1$ to k:

if high-point(3_i) < high-point(v)

then set high-point(v) to be high-point(3_i).

$$\text{Total time} = \sum_{v \in V(G)} \deg(v) = O(m)$$

① Find DFS & levels

② Compute high points

③ $\forall v$ compare high-point & level(v).

All bridges can be reported in $O(m)$ time a connected graph.