Red-black trees and 2-3 trees

- There is a 1-1 correspondence between red-black BSTs and 2-3 trees.

- To see this, imagine that red links are collapsed: the collapsed nodes correspond to 3-nodes, all others to 2-nodes.

We show: For every red-black tree there is a 2-3 tree

- Given a red-black tree T, we build a 2-3 tree T'
- The nodes of the 2-3 tree T' are obtained as follows.
  - for every node v_{rb} in the red-black tree with key k that is incident to black edges only create a 2-node v_{23} for the 2-3 tree with key k. That is: $23(v_{rb}) = v_{23}$
  - for every red edge and its incident two nodes v_{rb} and w_{rb}—where w_{rb} is the parent of v_{rb}—containing keys k_1 and k_2, respectively, create a 3-node vw_{23} containing keys k_1 and k_2 (with k_1 being the left entry). That is: $23(v_{rb}) = 23(w_{rb}) = vw_{23}$

We show: For every red-black tree there is a 2-3 tree

- The edges of the 2-3 tree T' are obtained as follows
  - Let $u_{rb}v_{rb}$ be a **black** edge in the red-black tree. Then $23(u_{rb})23(v_{rb})$ is an edge in the 2-3 tree. Further,
    - if u_{rb} is the left child of v_{rb} then $23(u_{rb})$ is the left child of $23(v_{rb})$
    - else if u_{rb} is the right child of v_{rb}, and the edge between the sibling of u_{rb} and v_{rb} is red, then $23(u_{rb})$ then is $23(v_{rb})$'s middle child
    - else $23(u_{rb})$ is the right child of $23(v_{rb})$

We show: For every red-black tree there is a 2-3 tree

- Let $u_{rb}v_{rb}$ be a **red** edge in the red-black tree where v_{rb} is the parent of u_{rb} and T_0 rooted by a_{rb} and T_1 rooted by b_{rb} are u_{rb}'s left and right subtree, respectively. Further, T_2 rooted by c_{rb} is the right subtree of v_{rb}. Then the following edges are in the 2-3-tree:
  - $23(a_{rb})23(u_{rb})$, $23(b_{rb})23(u_{rb})$, $23(c_{rb})23(v_{rb})$, and $23(v)23(w)$
  - Here, $23(a_{rb})$ is $23(u_{rb})$'s left child, $23(b_{rb})$ is $23(u_{rb})$'s middle child, and $23(c_{rb})$ is $23(u_{rb})$'s right child

We show: For every 2-3 tree there is a red-black tree

- Given a 2-3 tree T, we build a red-black tree T'
- The nodes of the red-black tree T' are obtained as follows.
  - for every 2-node v_{23} in the red-black tree with key k create node v_{rb} with key k.
  - for every 3-node vw_{23} with keys k_1 and k_2, $k_1 \leq k_2$, create nodes v_{rb} and w_{rb} containing keys k_1 and k_2, respectively.

We show: For every red-black tree there is a 2-3 tree

- The edges of the red-black tree are obtained as follows
  - for every remaining edge between 2-node w_{23} and child v_{23} the corresponding red-black tree nodes are connected by a black edge
  - for every 3-node vw_{23} with keys k_1 and k_2, $k_1 \leq k_2$, and its corresponding red-black tree nodes v_{rb} with key k_1 and w_{rb} with key k_2 we define recursively:
    - add **red** edge $v_{rb}w_{rb}$ is in red-black tree T' with v_{rb} being w_{rb}'s left child
    - the red-black tree of vw_{23}'s left subtree is v_{rb}'s left subtree, the red-black tree of vw_{23}'s middle subtree is v_{rb}'s right subtree, and the red-black tree of vw_{23}'s right subtree is w_{rb}'s right subtree

The height of a red-black tree with n keys is $O(\log(n))$

- *Proof Idea.* The (black) path length from leaf to roof the same as the height of its corresponding 2-3 trees, that is $O(\log(n))$. The height of a red-black tree can be twice the height of the 2-3 tree, namely in the case that every second edge in a path from leaf to root is red. The height remains $O(\log(n))$.

Insertion of key k

- Recall that inserting a key into a red-black tree consists of the following steps:
  - Search for key k (as for BSTs)
  - Replace the found leaf by node v containing key k. The link to v's parent w is red (if w exists)
  - Restructure and re-color if necessary

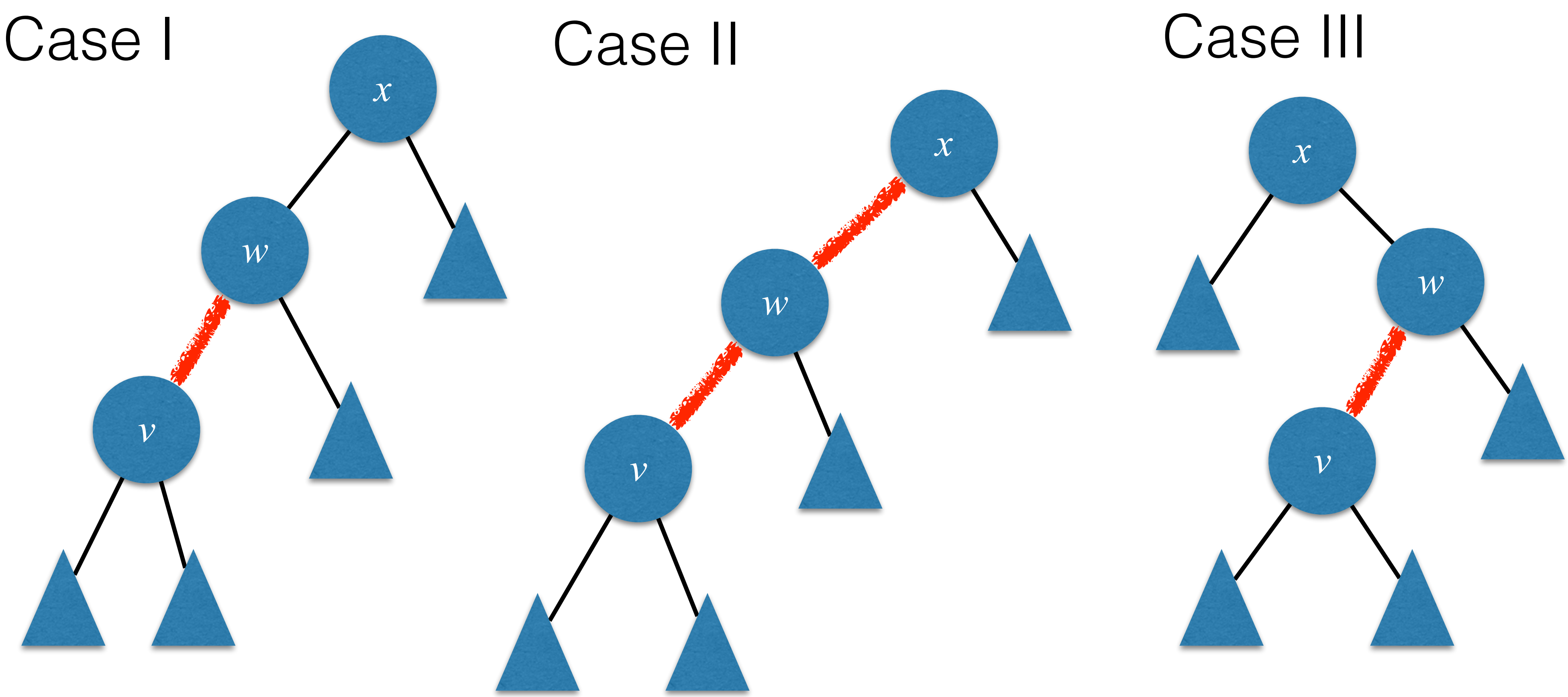
Inserting into a red-black tree: the cases

1. Key k is inserted into an empty tree

2. When inserting key k into new node v, v is a left child

3. When inserting key k into new node v, v is a right child

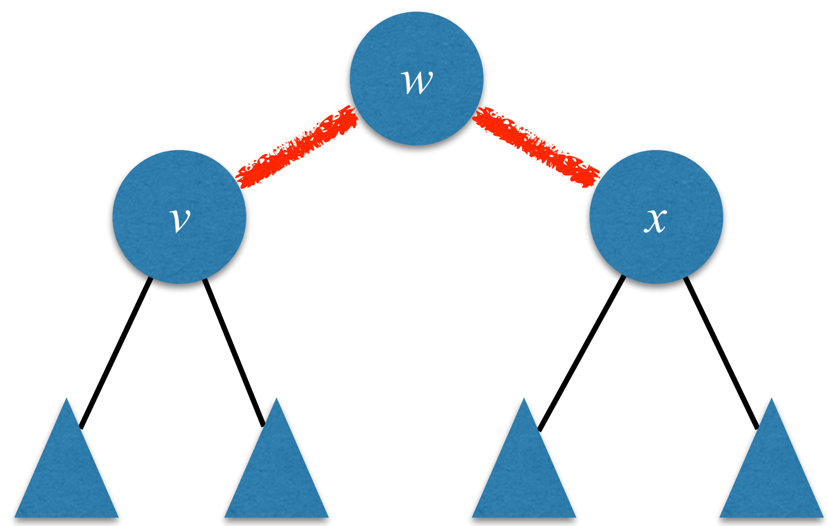
When inserting key k into new node v, v is a left child



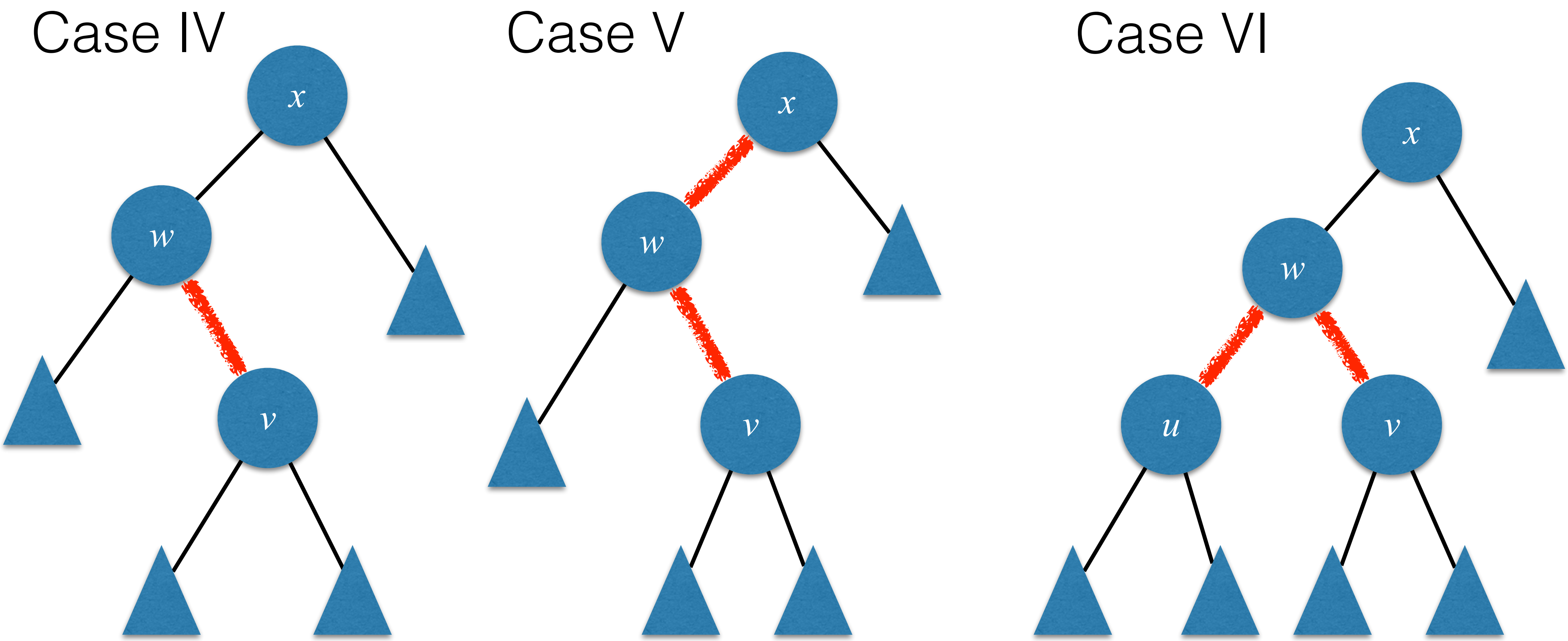
Case Study

- Cases I,III: No red-black tree property violated
- Case II: w has two incident red edges. Restructure subtree rooted by x and obtain:

Note that the edge from w to its parent (if existent) is black. This results in Case IV or VIII

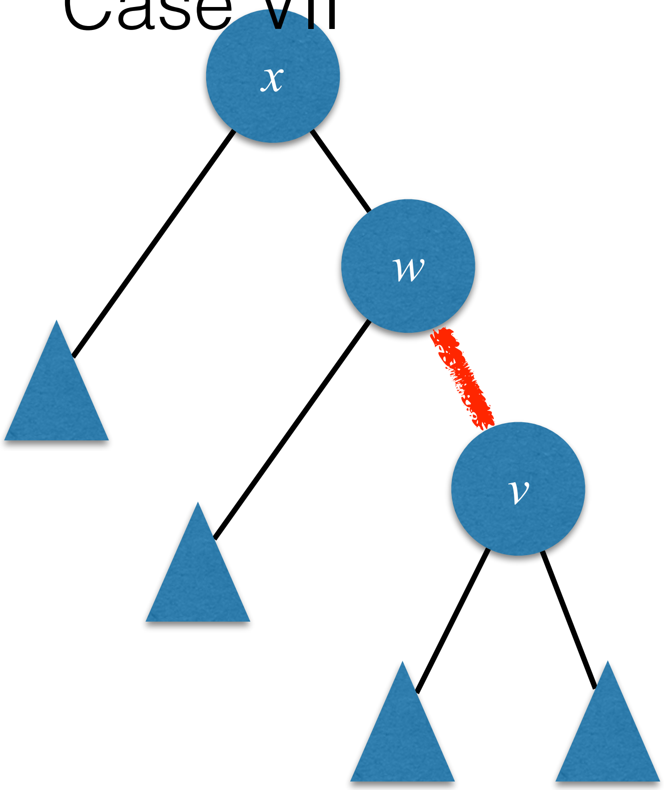


When inserting key k into new node v, v is a right child

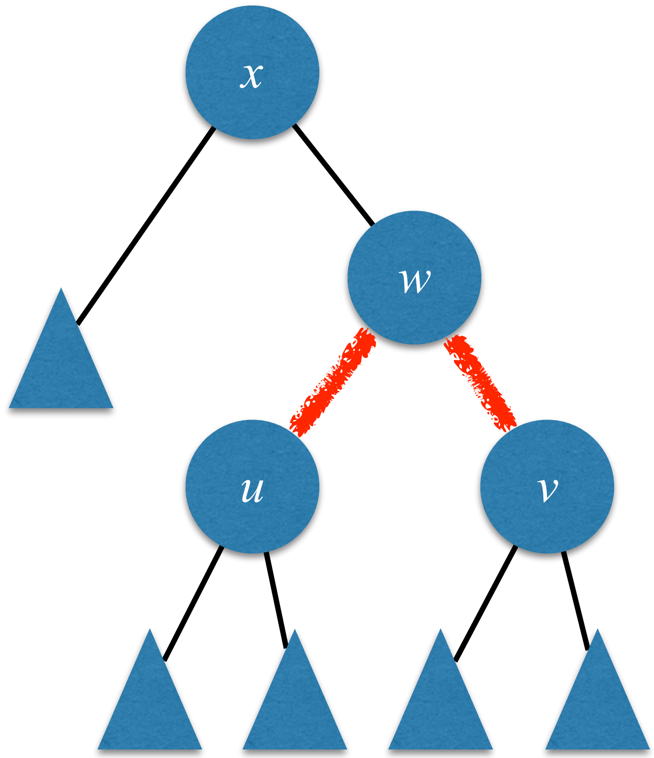


When inserting key k into new node v, v is a right child

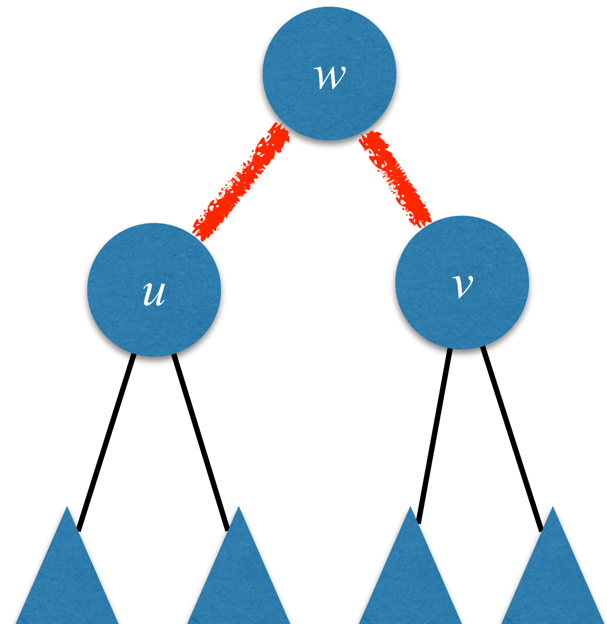
Case VII



Case VIII



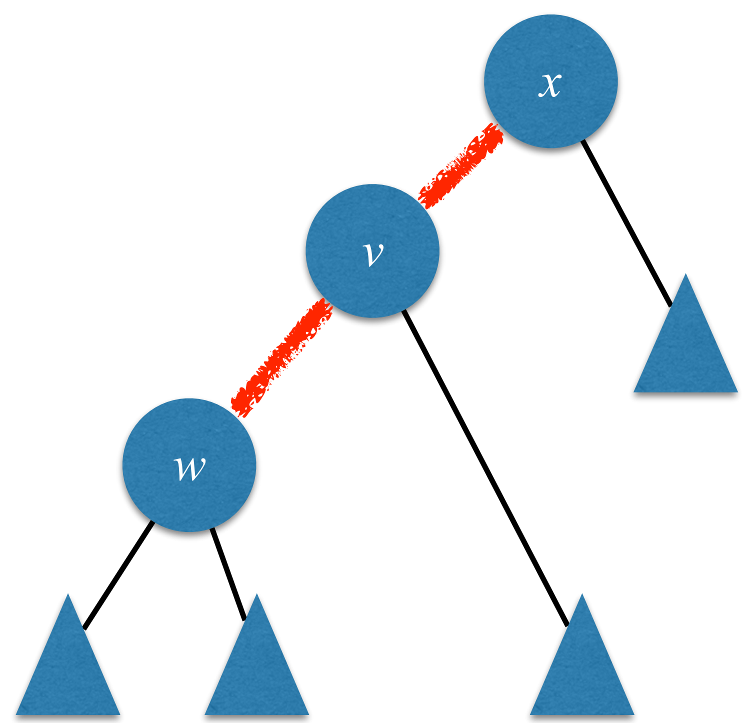
Case IX



w is root

Case Study

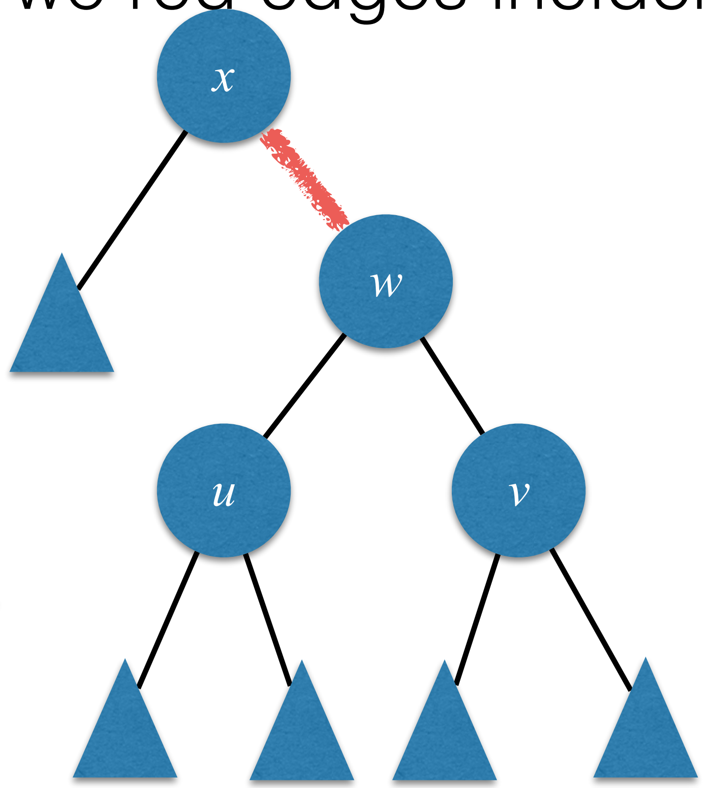
- Cases IV,VII: No red-black tree property violated
- Case V: w has two incident edges. Rotate edge wv and obtain:



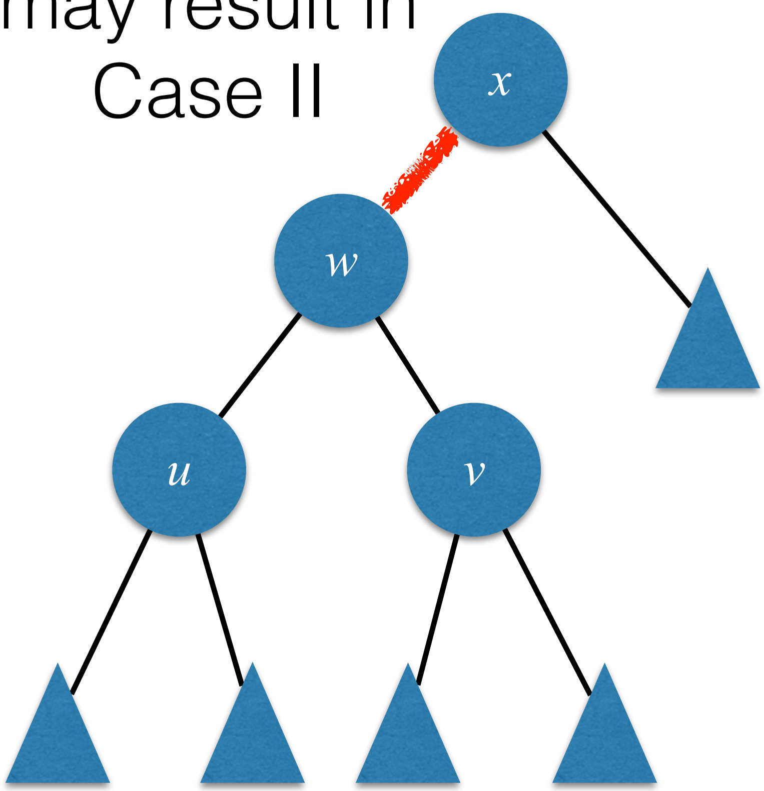
This results in Case II.

Case Study

- Case VI, VIII, IX: Two red edges incident to w. Flip colors:



may result in Case II



may result in Case V