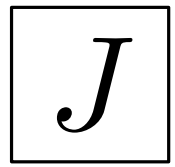
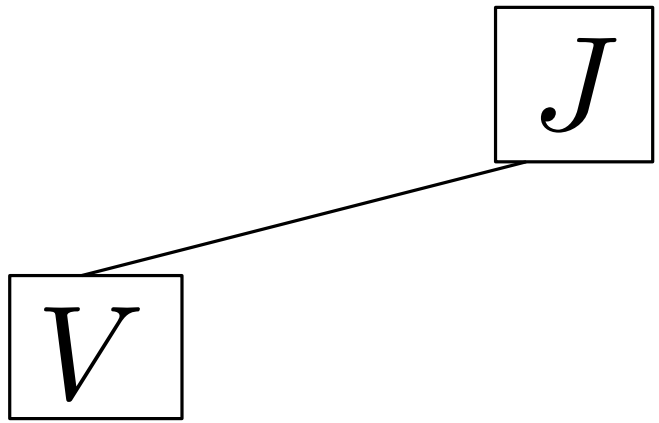
Phase I: Organize your data into a heap.

$J \;\; V \;\; D \;\; E \;\; H \;\; O \;\; R \;\; C$

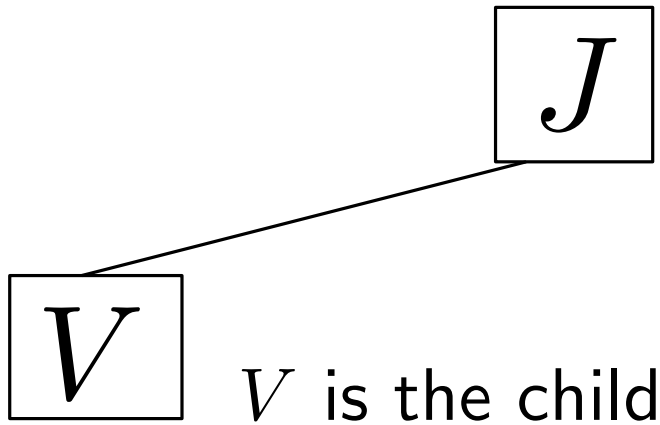
$$J \quad V \quad D \quad E \quad H \quad O \quad R \quad C$$



$V\ D\ E\ H\ O\ R\ C$



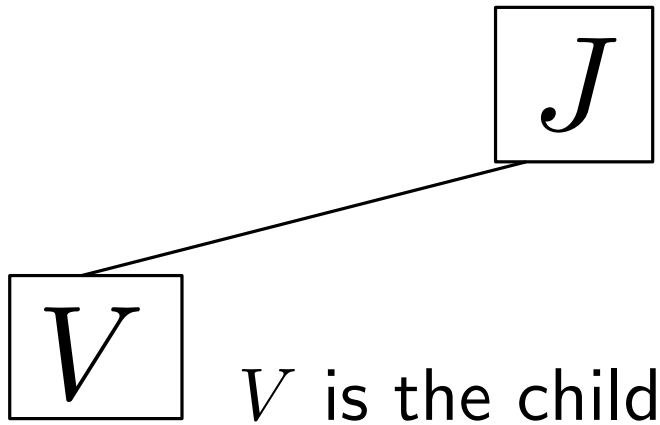
$$D \quad E \quad H \quad O \quad R \quad C$$



V is the child of J.
J is the parent of V.

$$D \;\; E \;\; H \;\; O \;\; R \;\; C$$

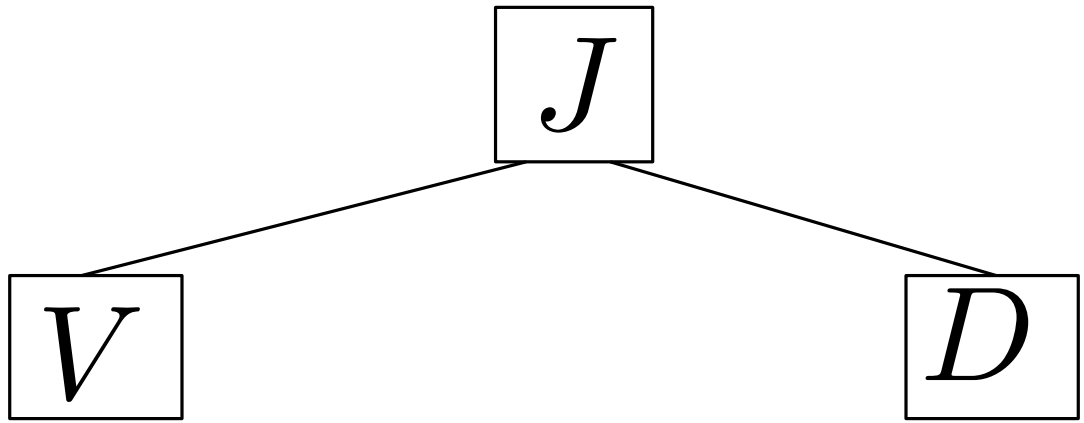
Heap Property: Children come later than their parent (alphabetically)



V is the child of J.
J is the parent of V.

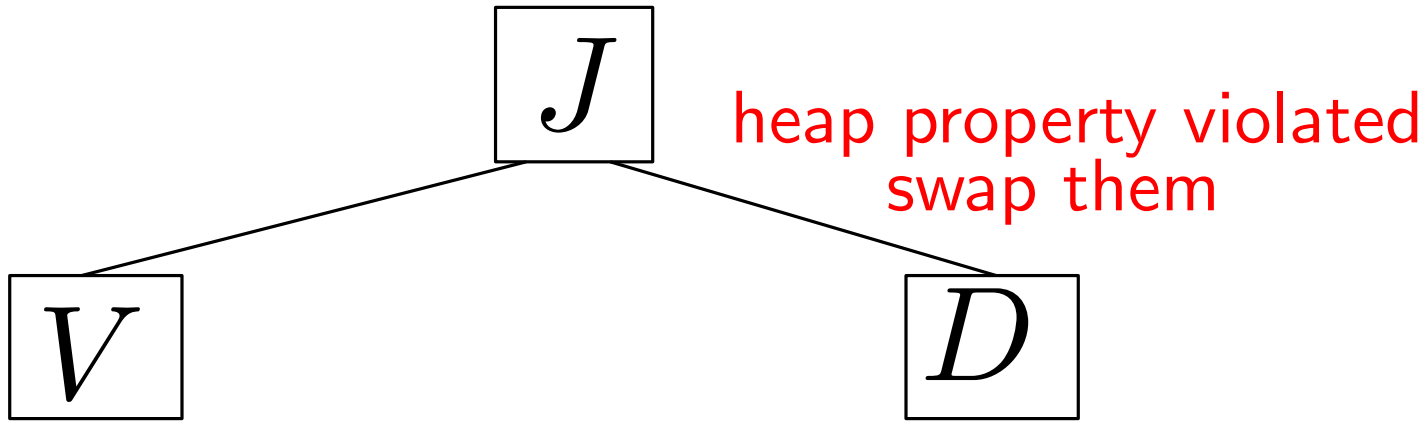
$$D \quad E \quad H \quad O \quad R \quad C$$

Heap Property: Children come later than their parent (alphabetically)



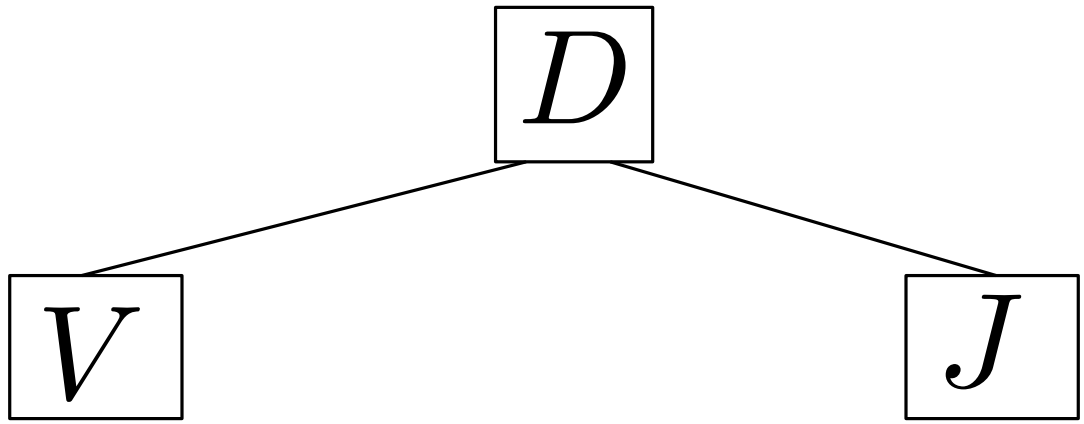
$$E \;\; H \;\; O \;\; R \;\; C$$

Heap Property: Children come later than their parent (alphabetically)



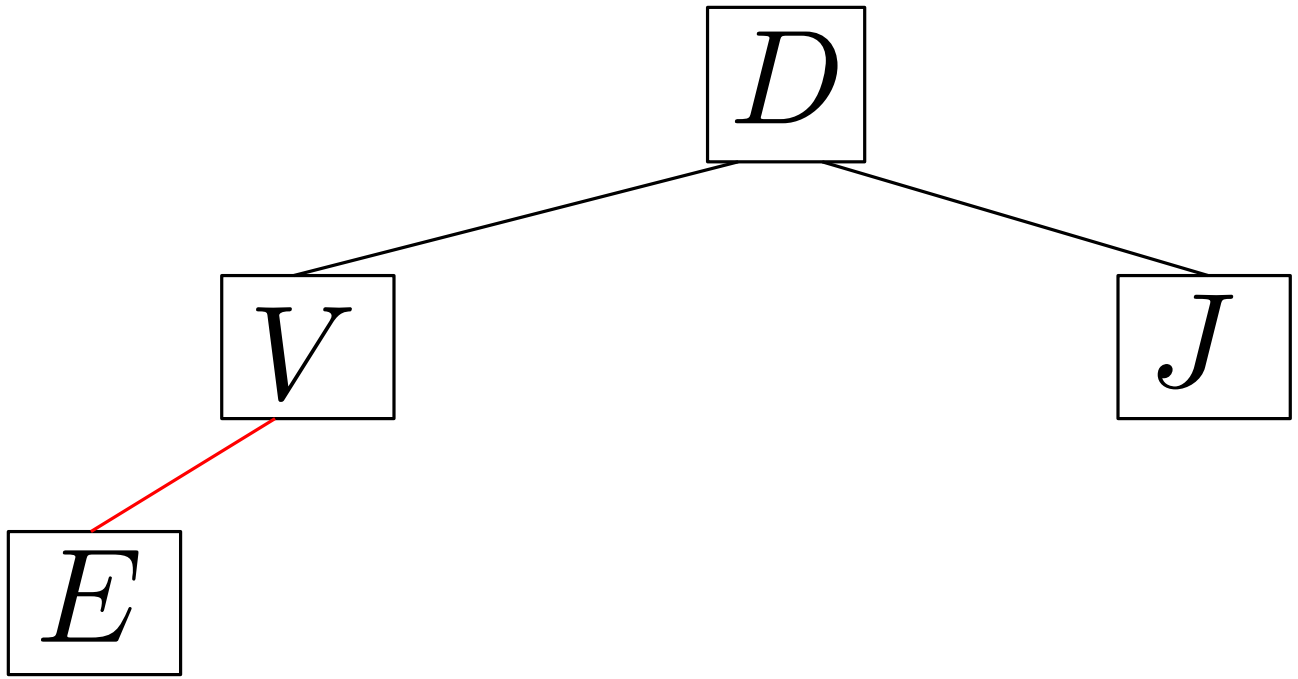
$$E \quad H \quad O \quad R \quad C$$

Heap Property: Children come later than their parent (alphabetically)



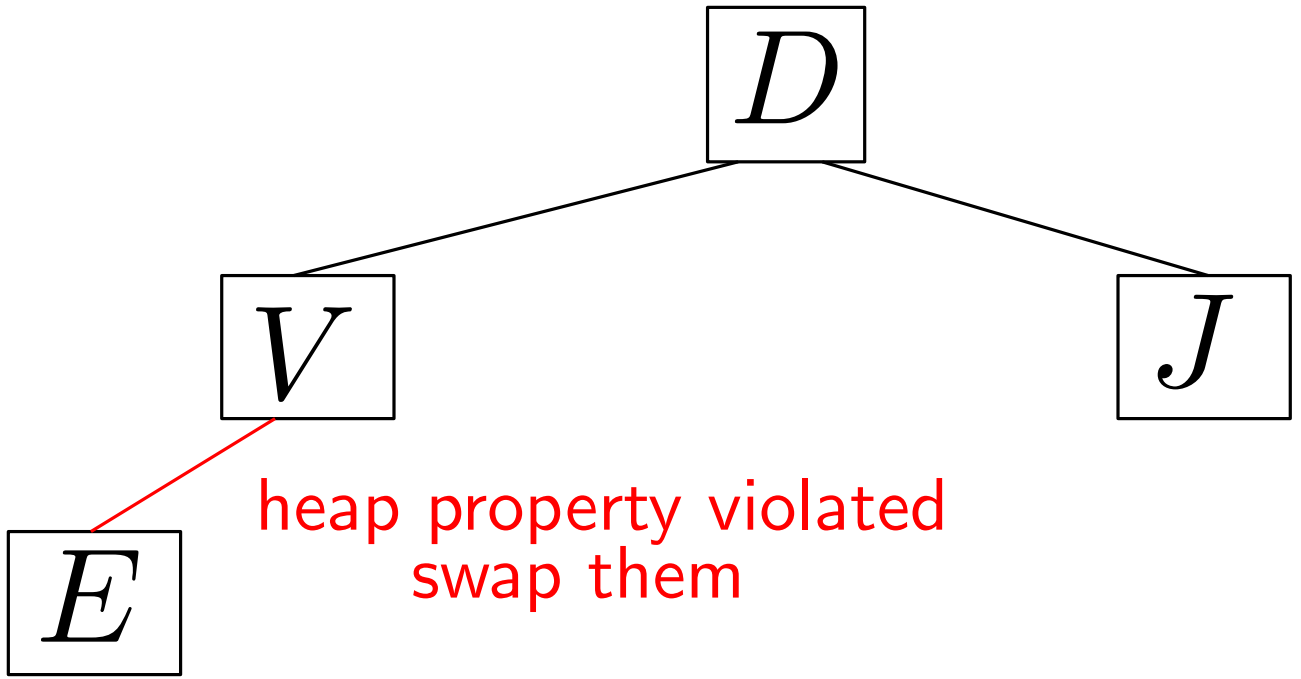
$$E \;\; H \;\; O \;\; R \;\; C$$

Heap Property: Children come later than their parent (alphabetically)



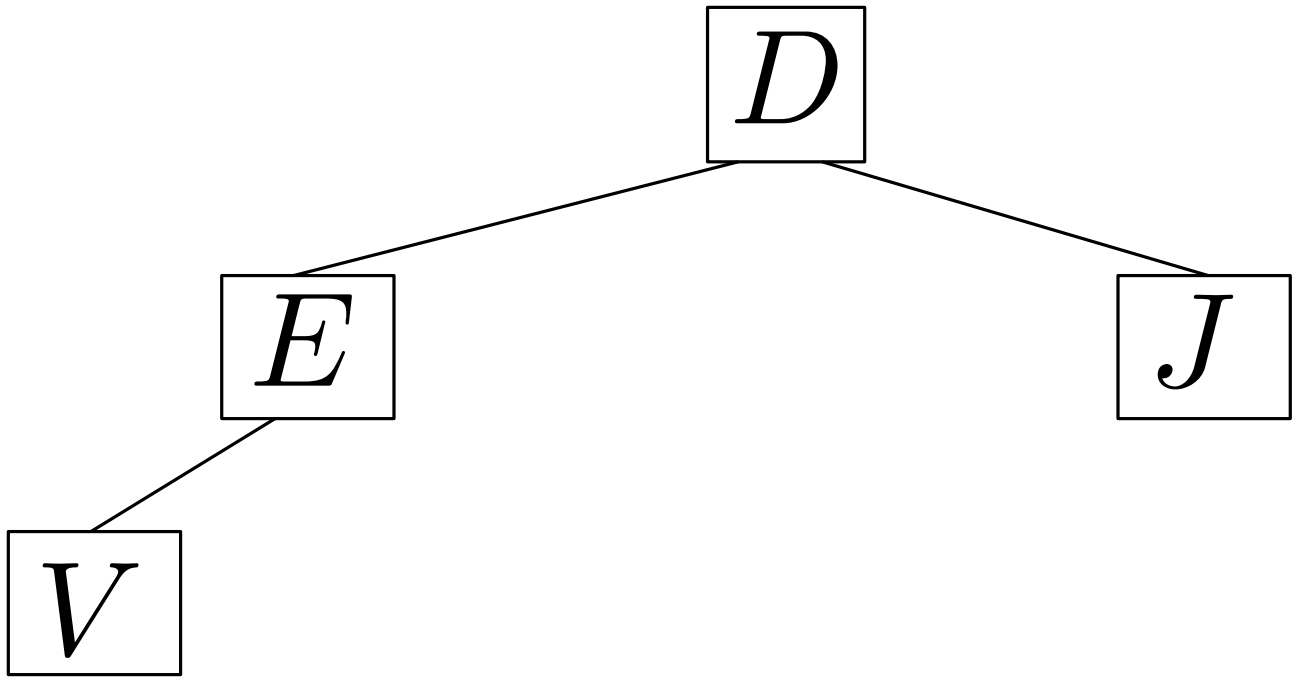
$$H \;\; O \;\; R \;\; C$$

Heap Property: Children come later than their parent (alphabetically)



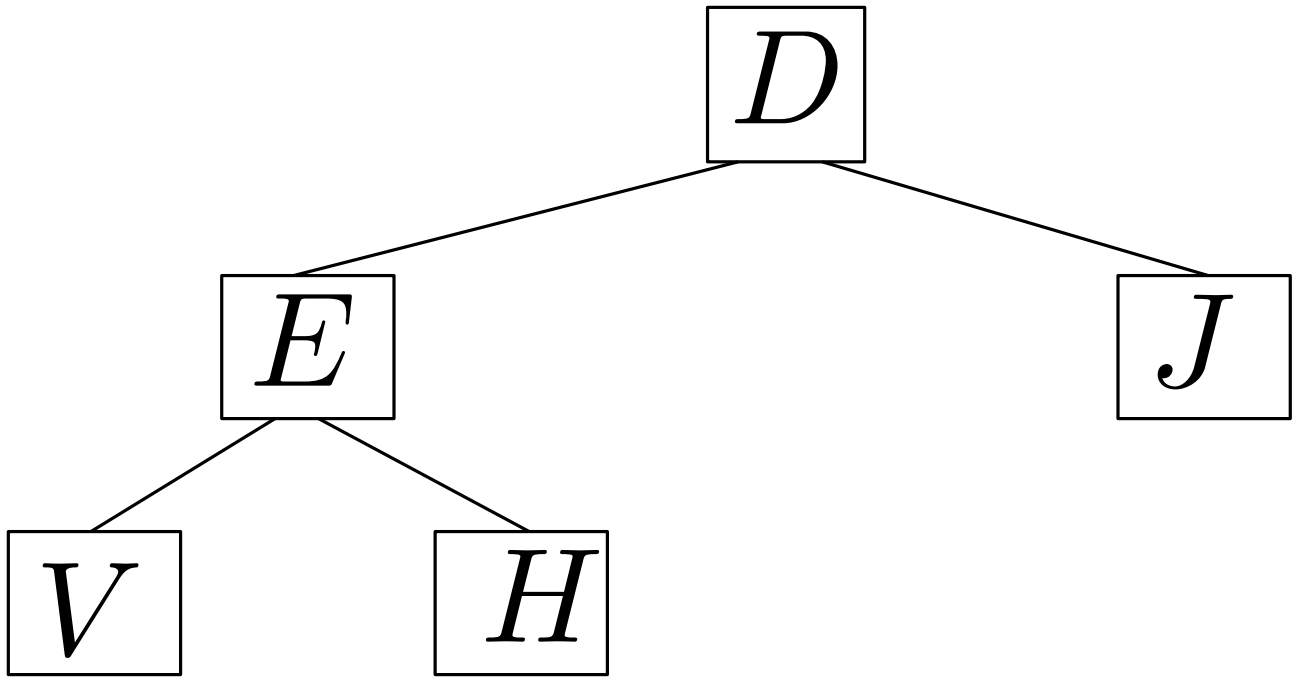
$$H \;\; O \;\; R \;\; C$$

Heap Property: Children come later than their parent (alphabetically)



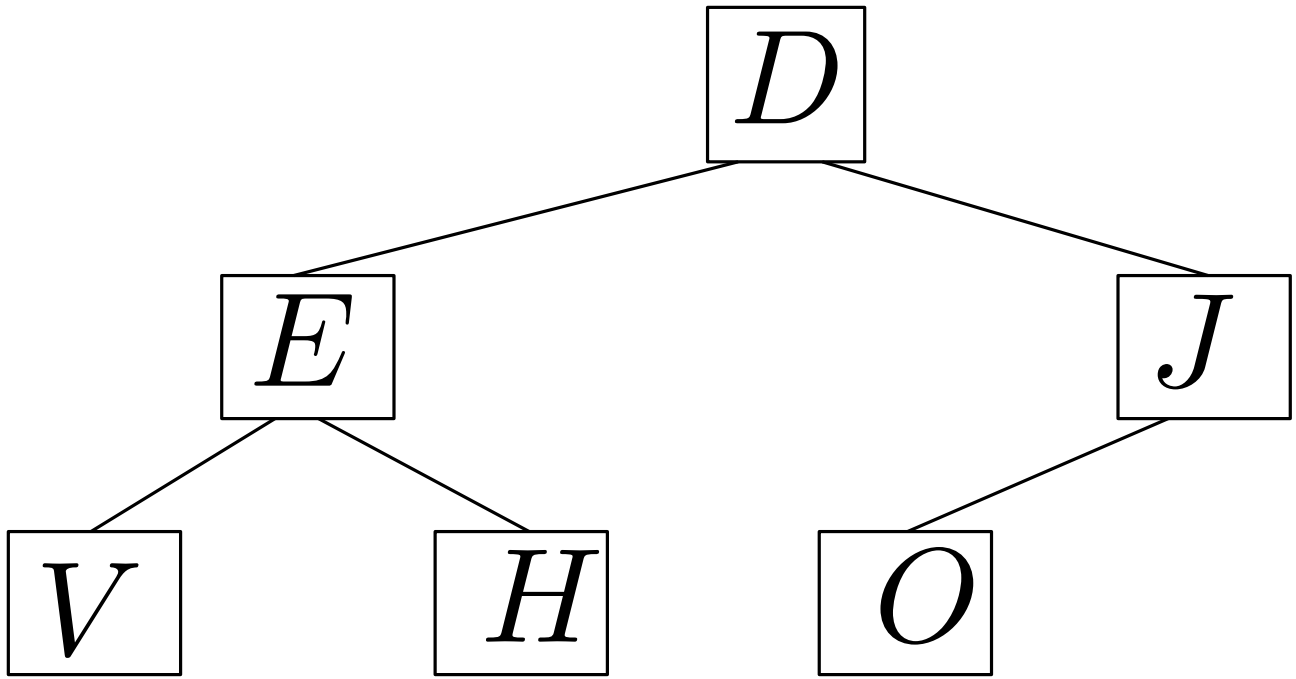
$$H \;\; O \;\; R \;\; C$$

Heap Property: Children come later than their parent (alphabetically)



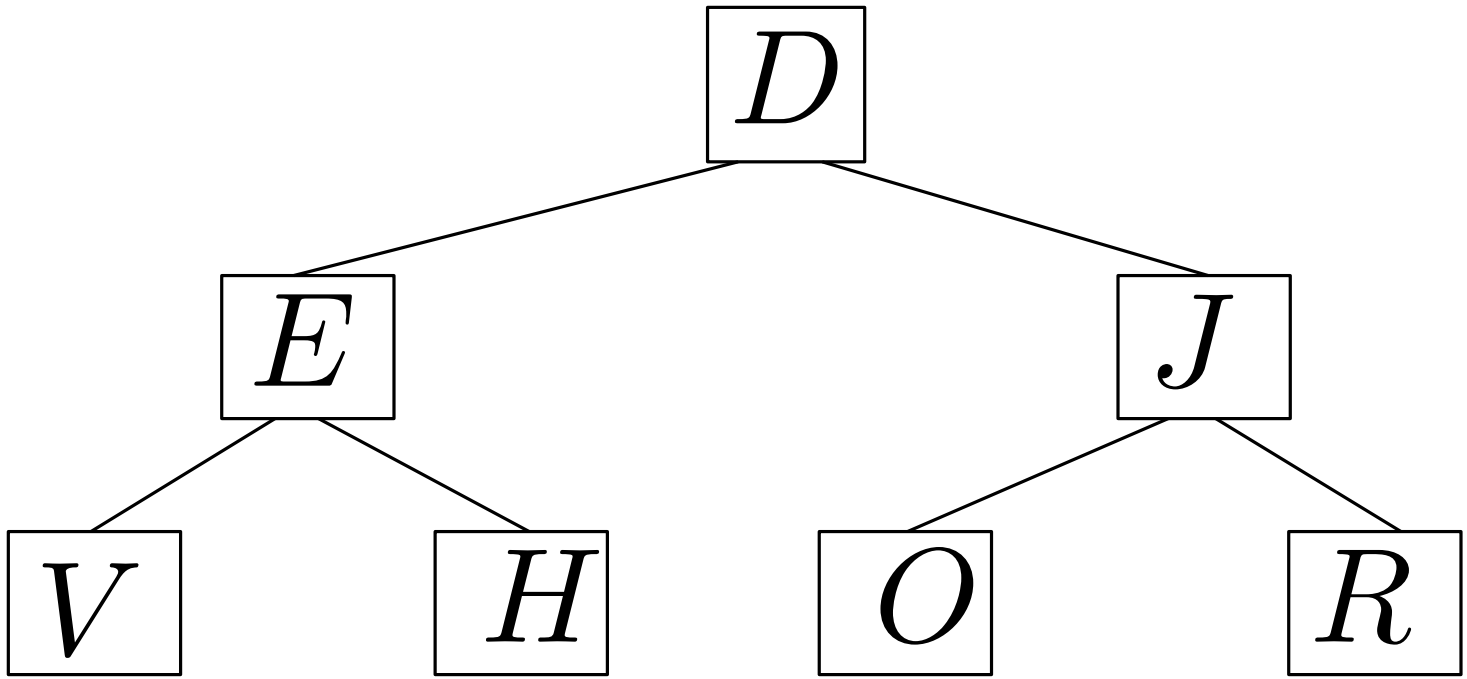
$$O \quad R \quad C$$

Heap Property: Children come later than their parent (alphabetically)



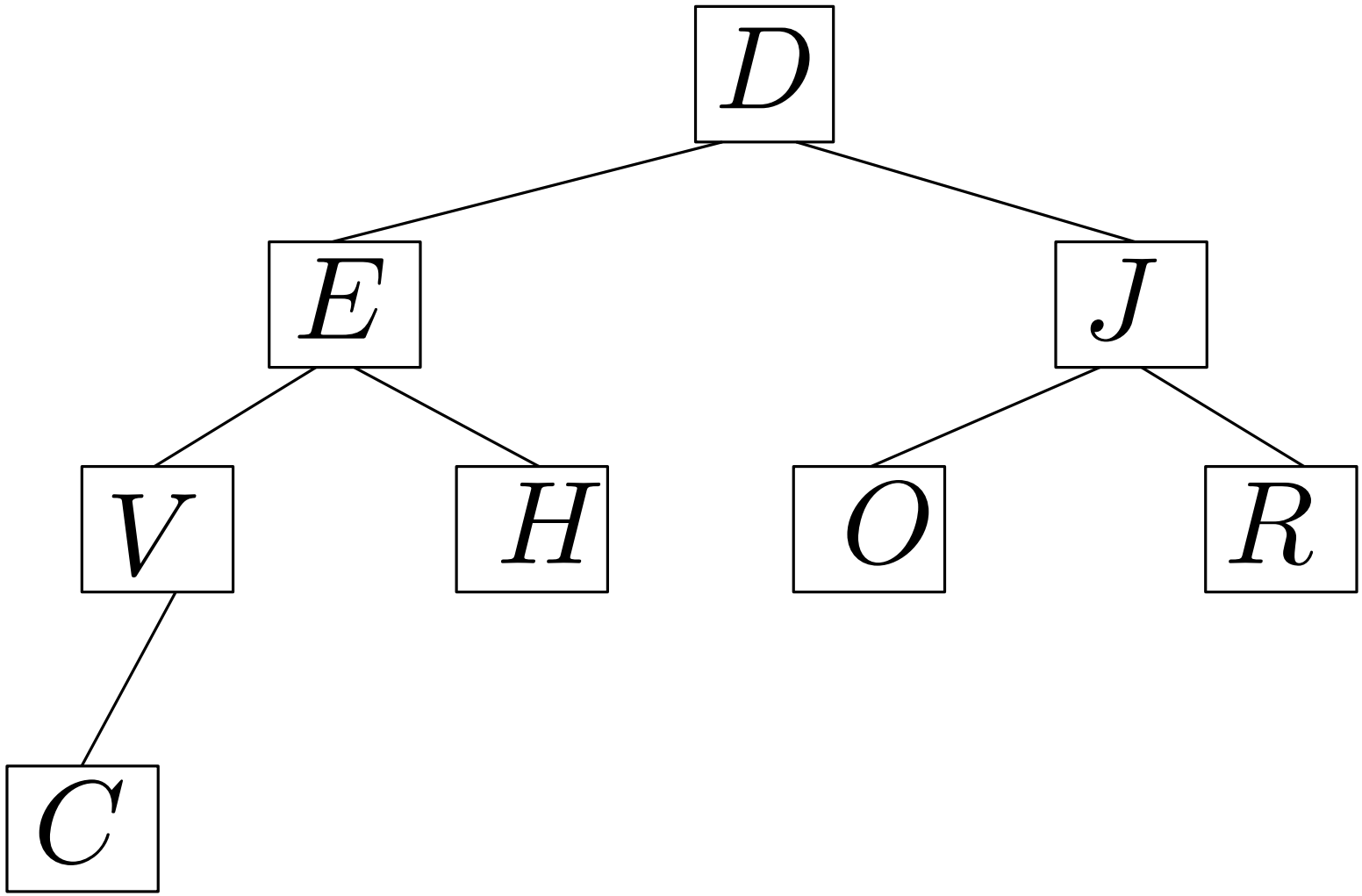
$R \;\; C$

Heap Property: Children come later than their parent (alphabetically)

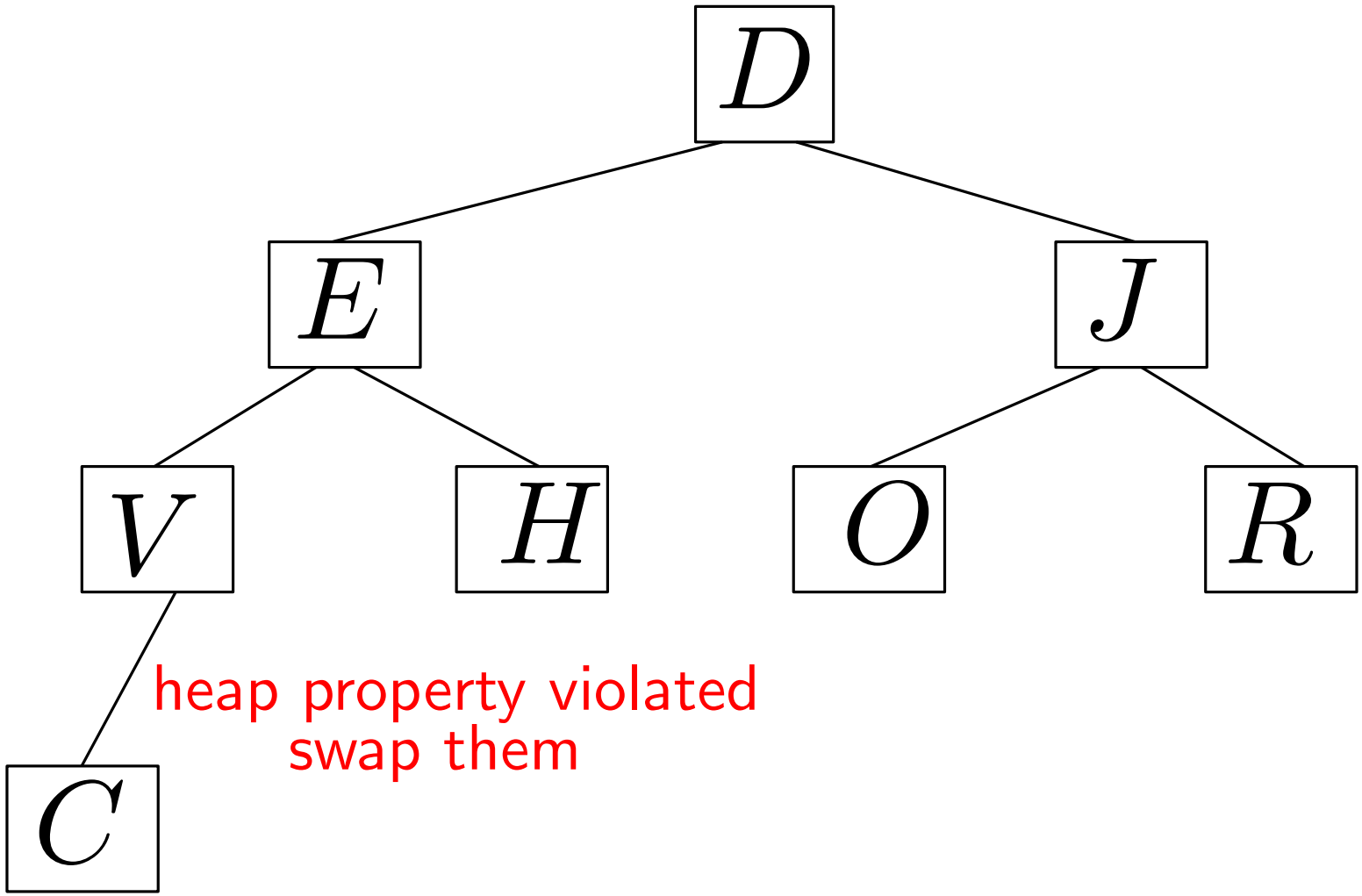


C

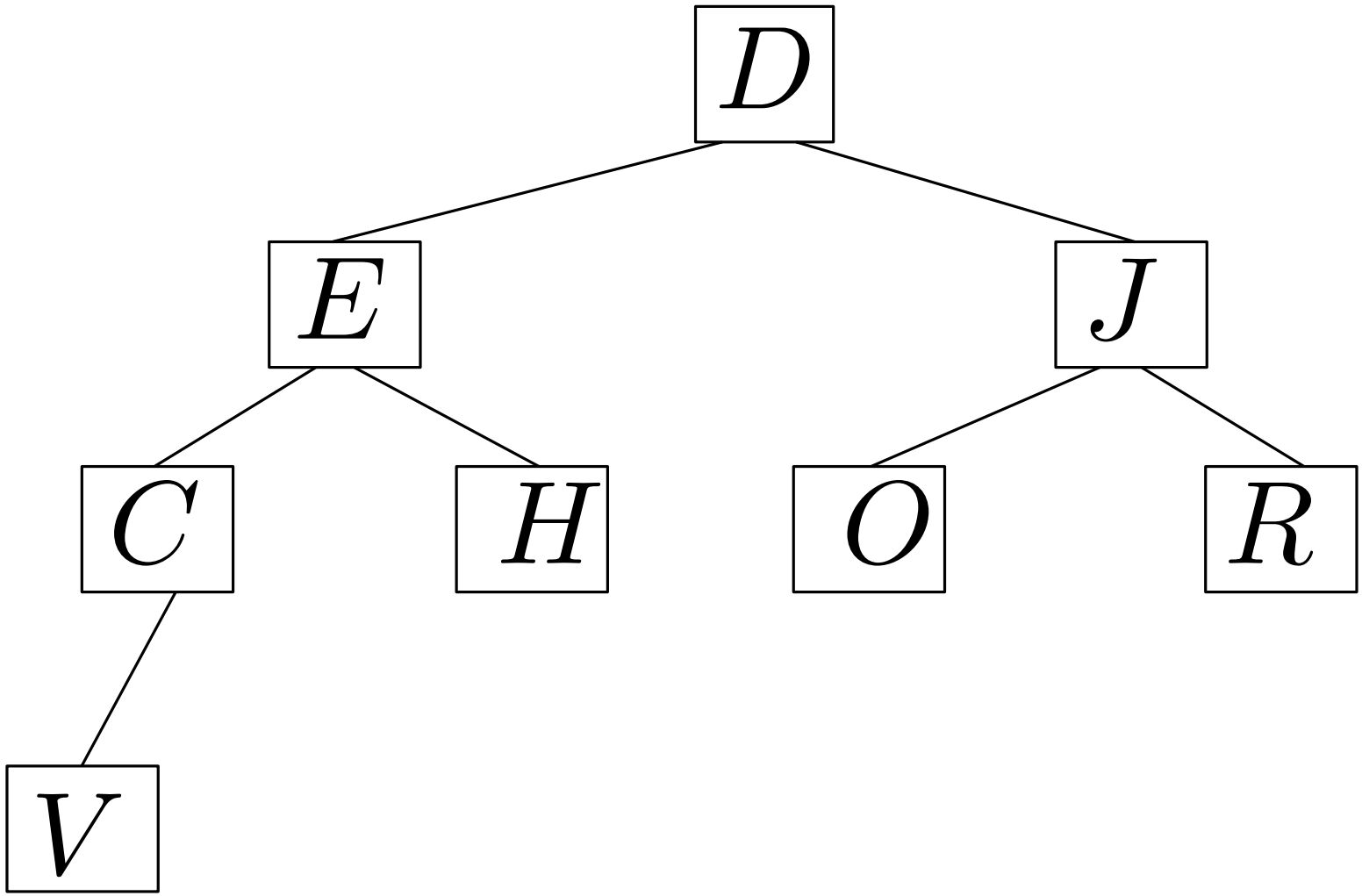
Heap Property: Children come later than their parent (alphabetically)



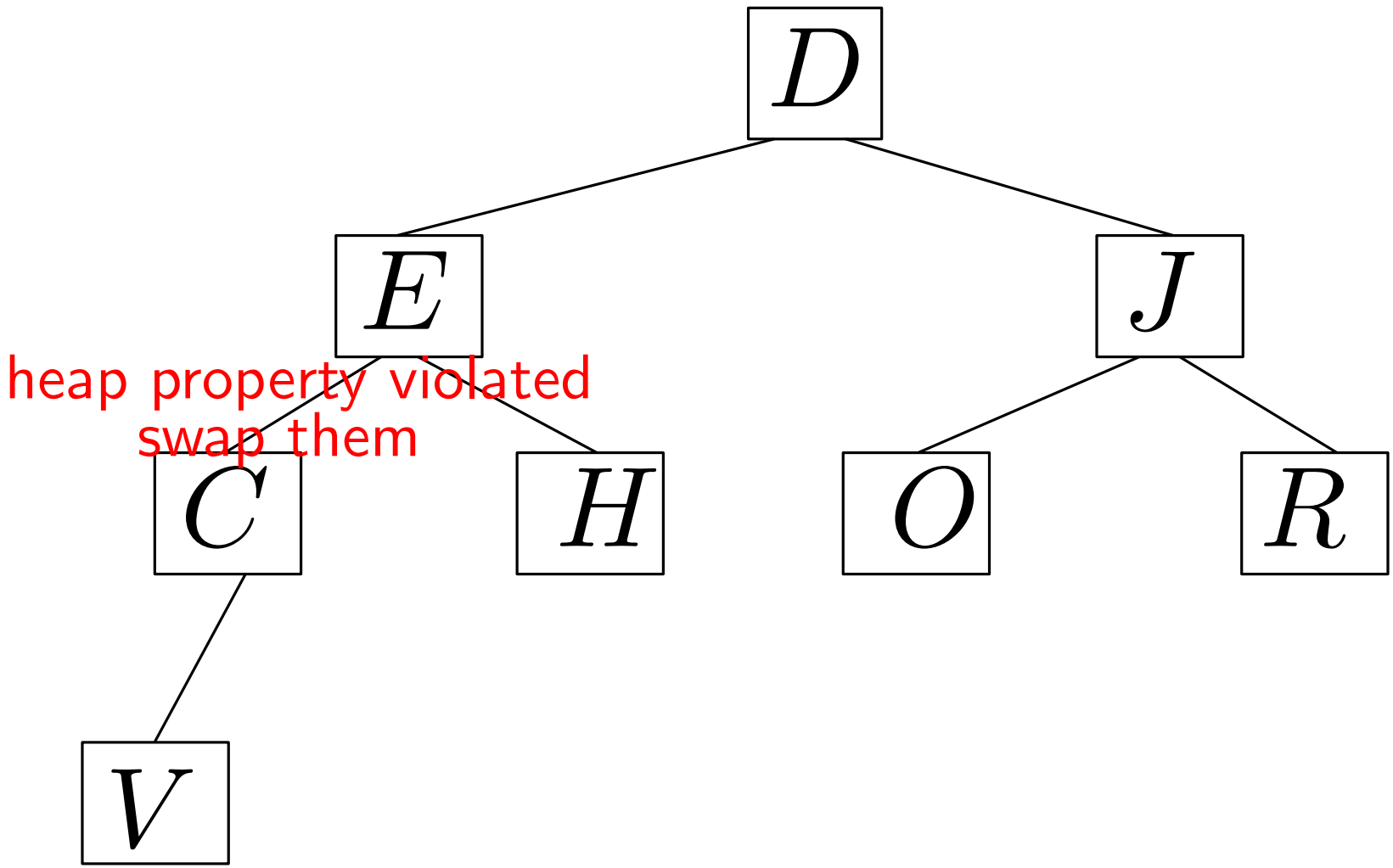
Heap Property: Children come later than their parent (alphabetically)



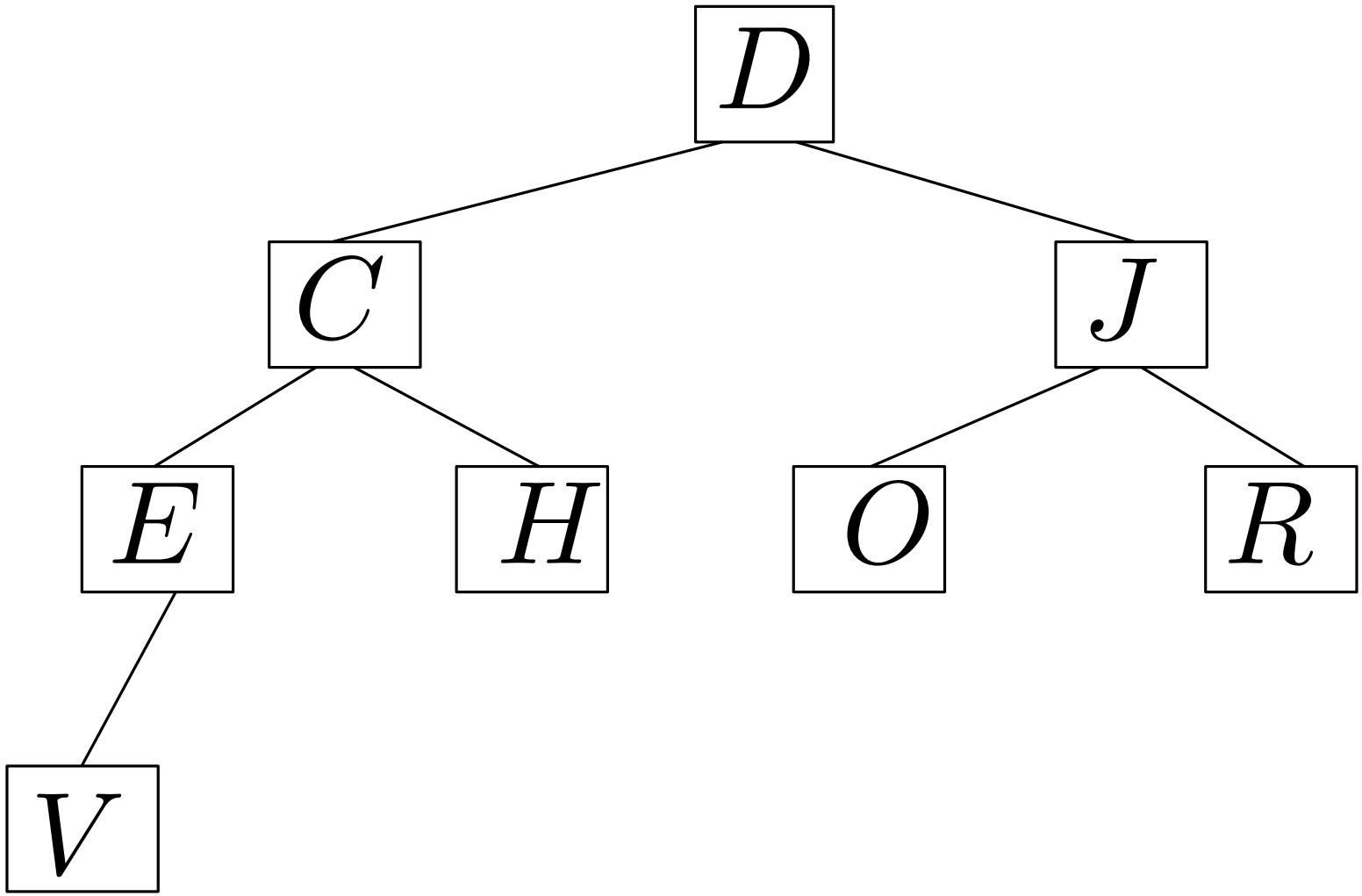
Heap Property: Children come later than their parent (alphabetically)



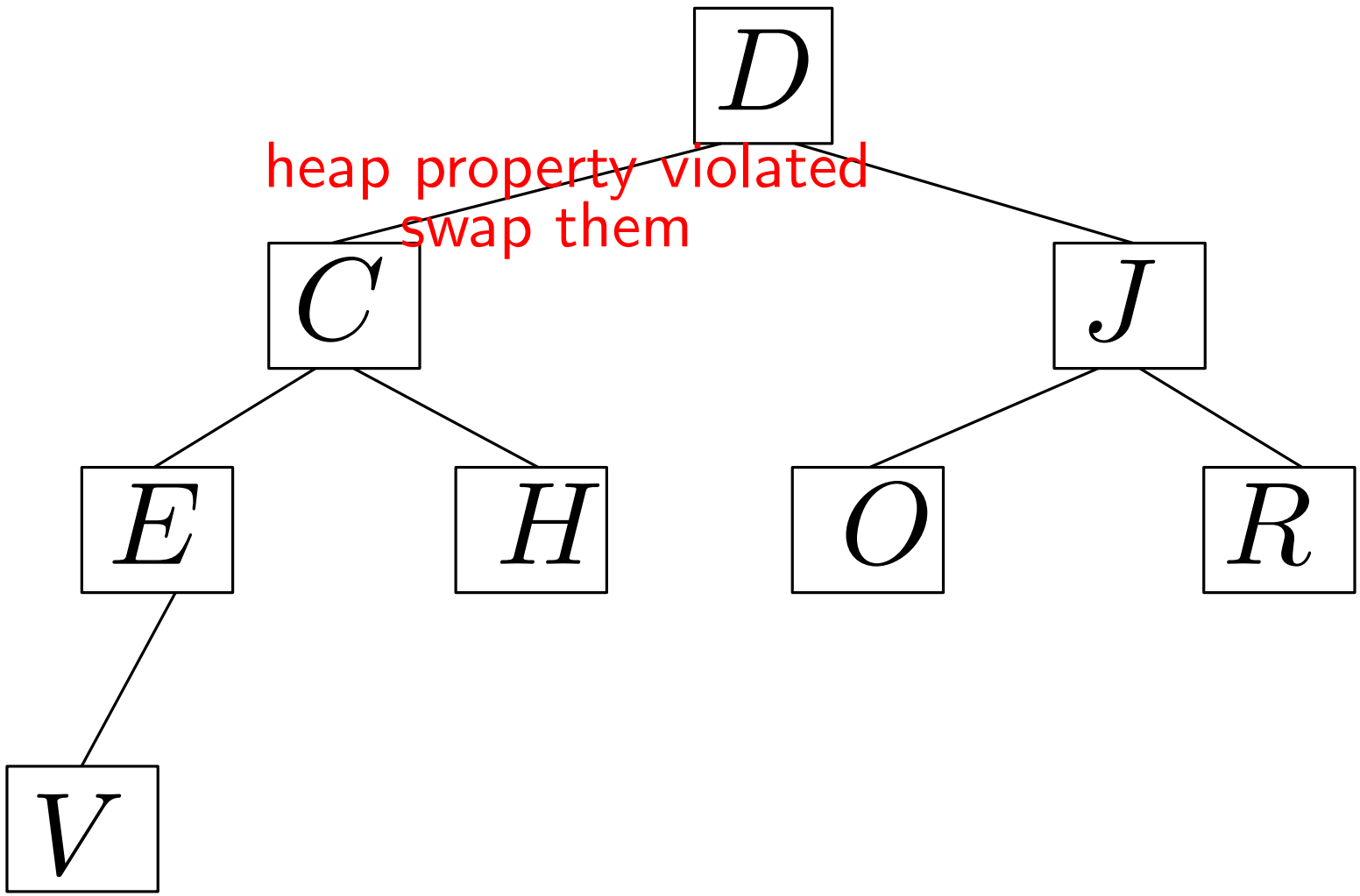
Heap Property: Children come later than their parent (alphabetically)



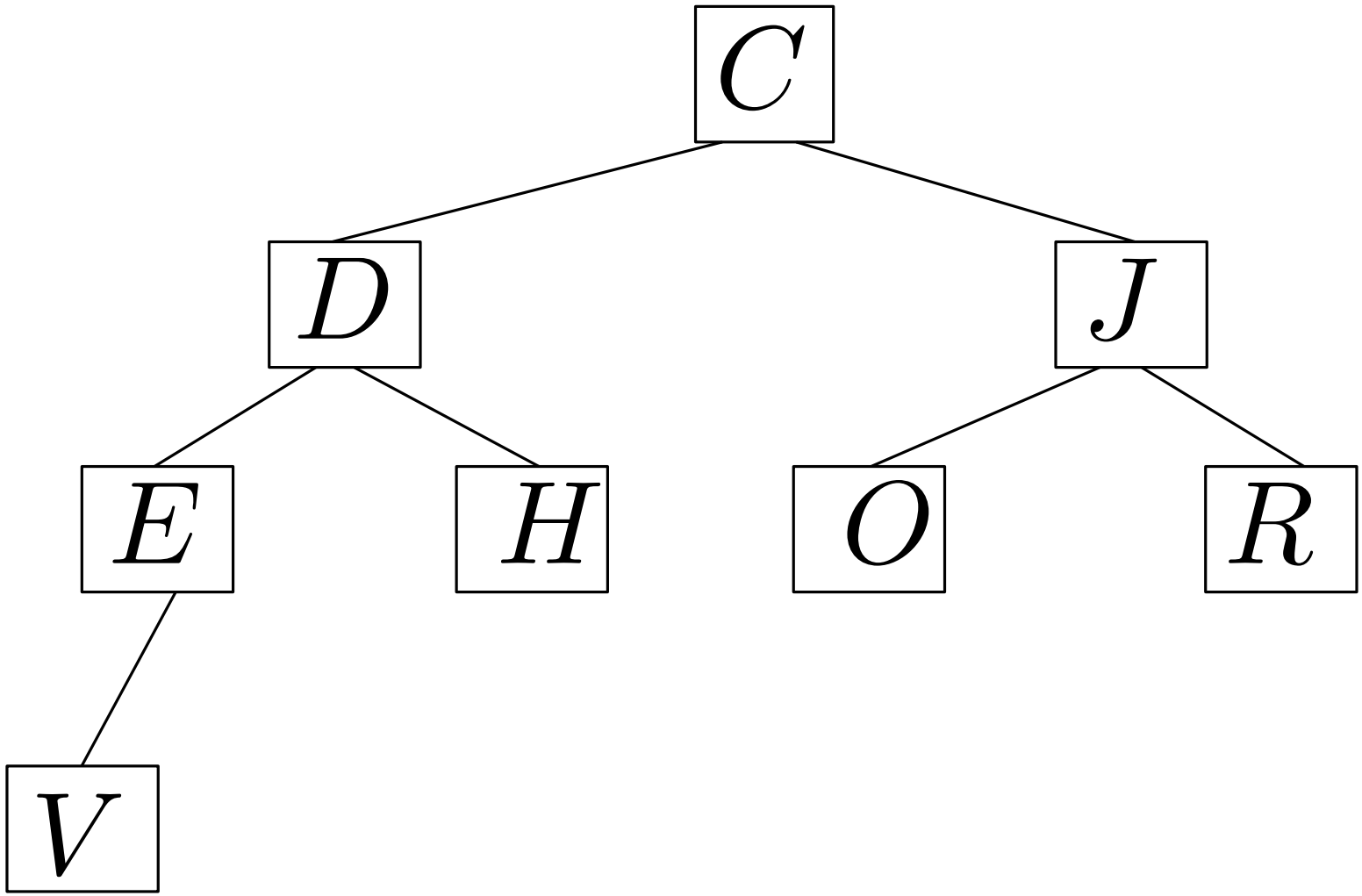
Heap Property: Children come later than their parent (alphabetically)



Heap Property: Children come later than their parent (alphabetically)

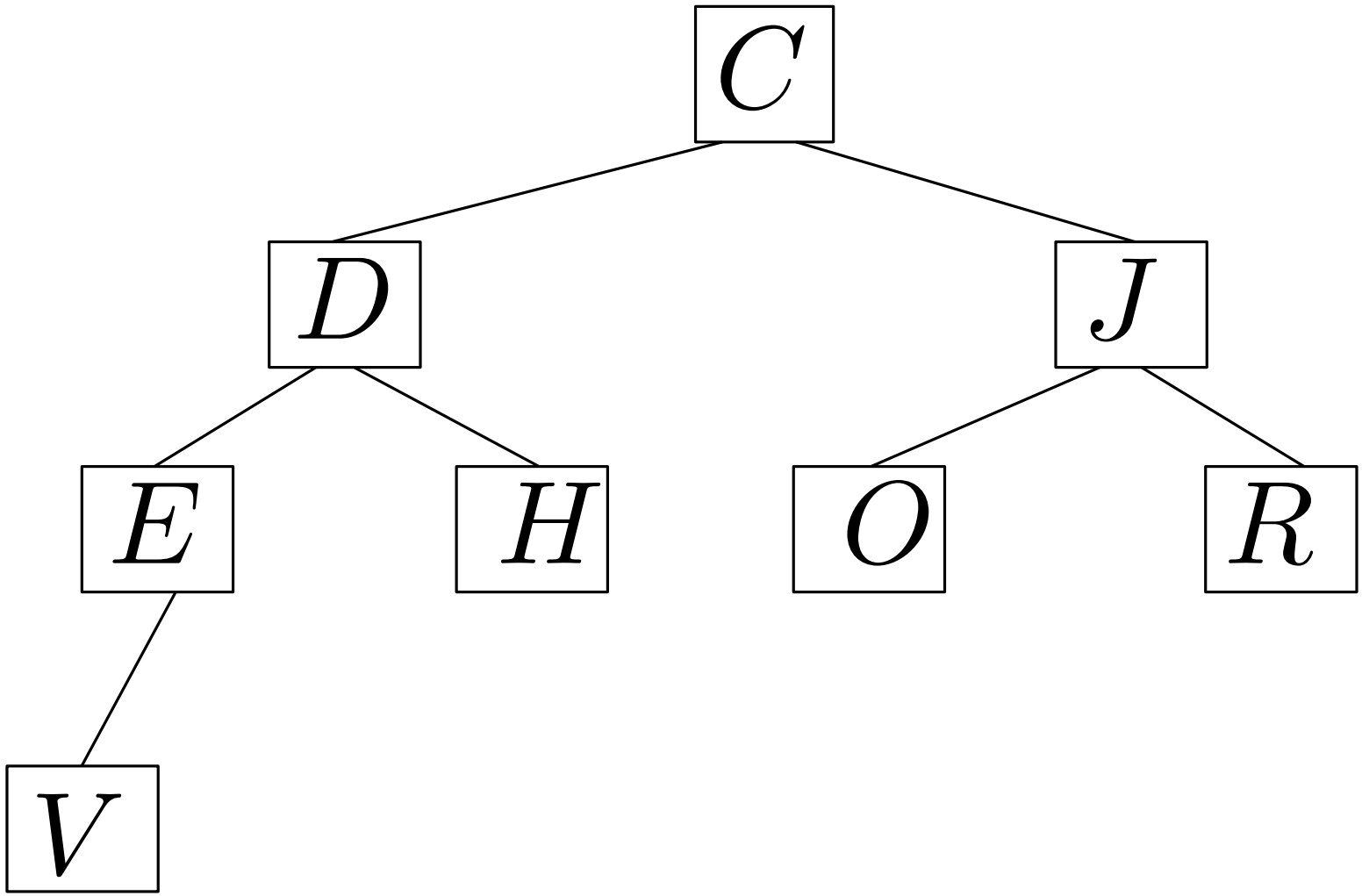


Heap Property: Children come later than their parent (alphabetically)

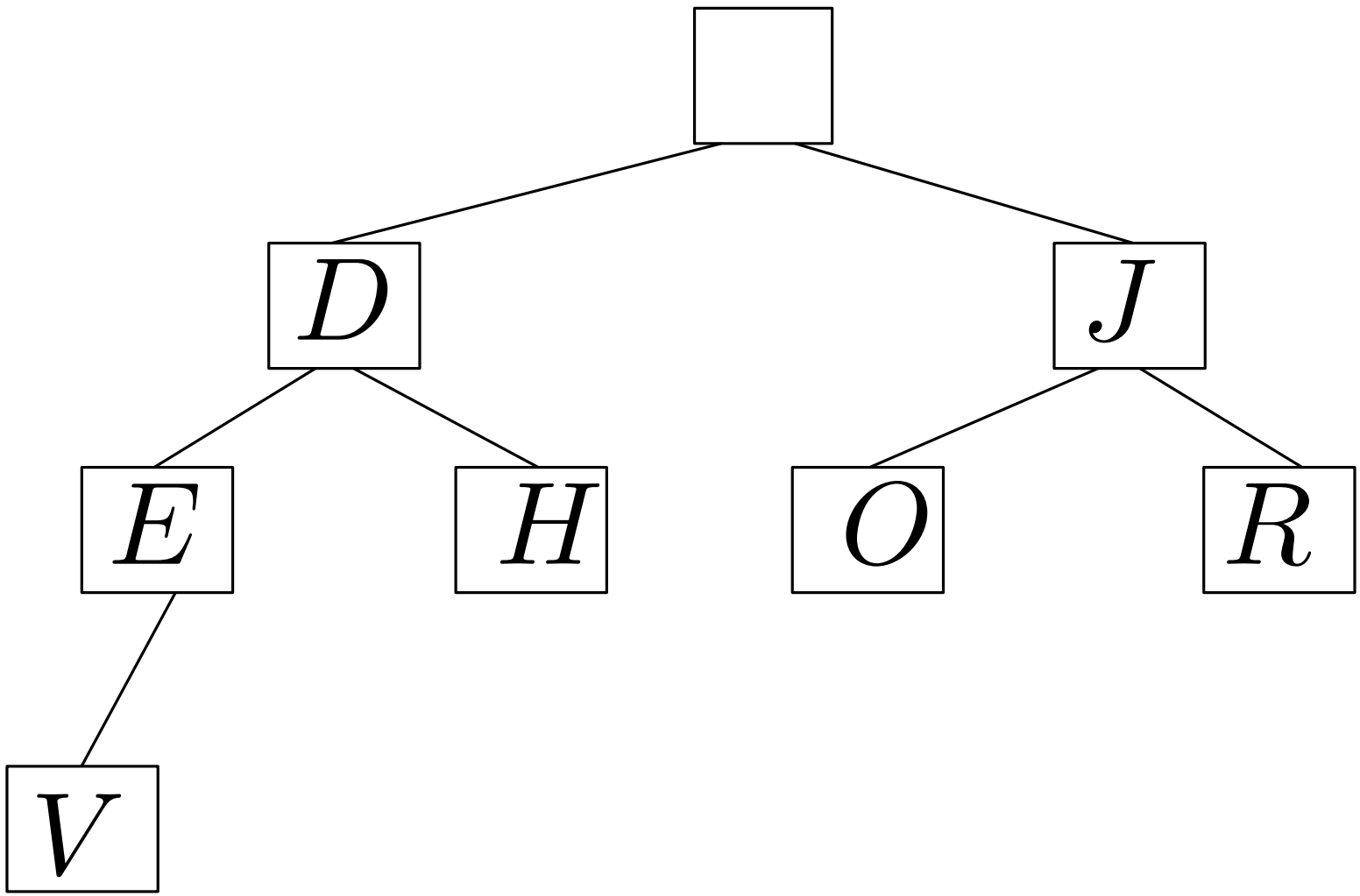


Phase II: Deconstruct heap into sorted array

Heap Property: Children come later than their parent (alphabetically)

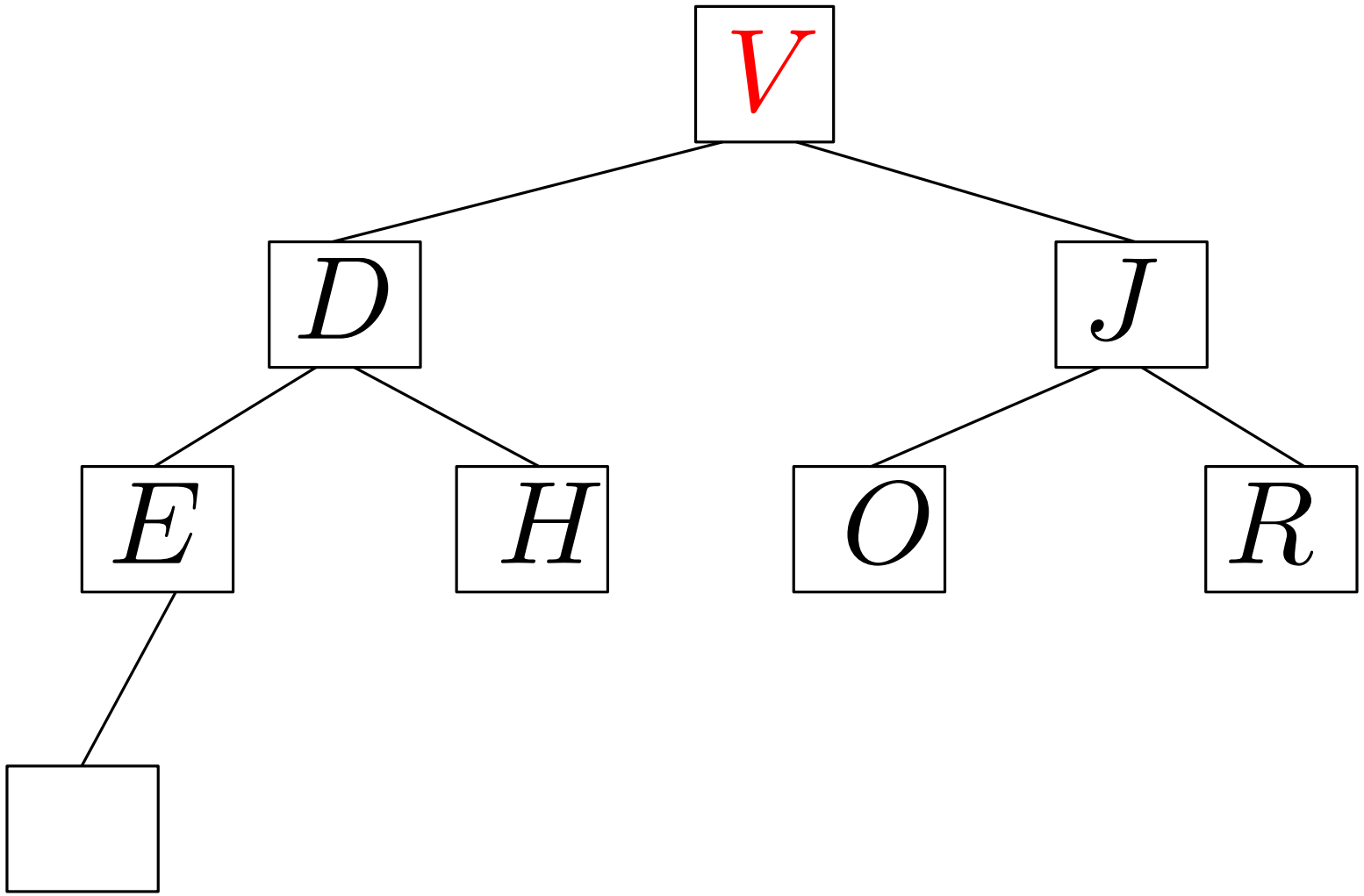


Heap Property: Children come later than their parent (alphabetically)



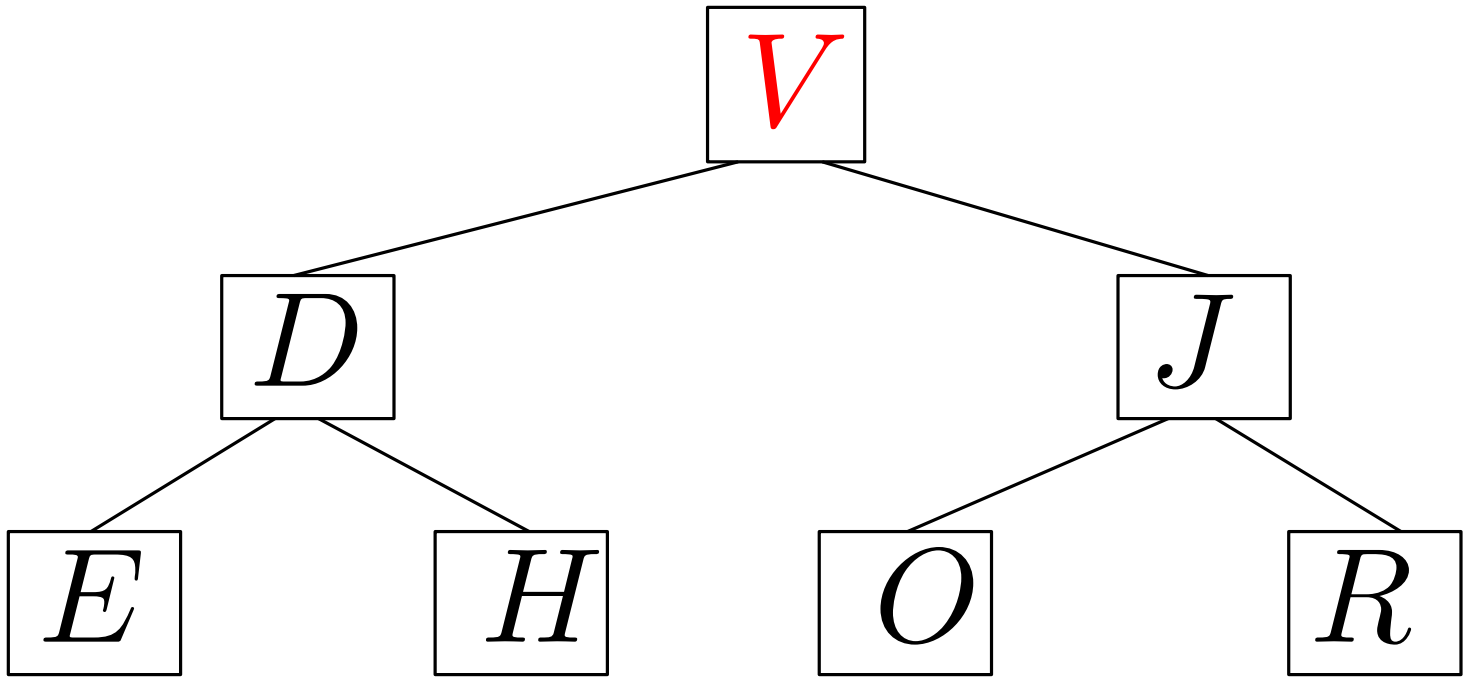
C

Heap Property: Children come later than their parent (alphabetically)



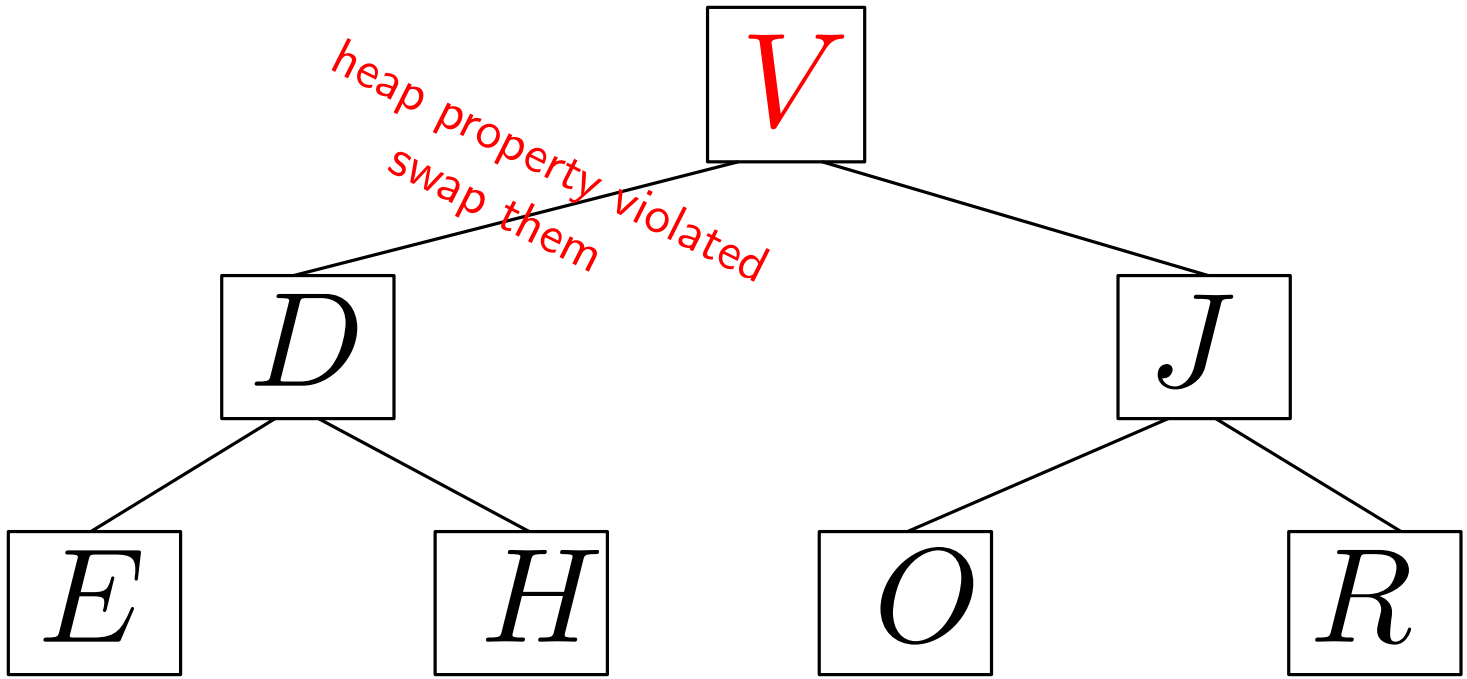
C

Heap Property: Children come later than their parent (alphabetically)



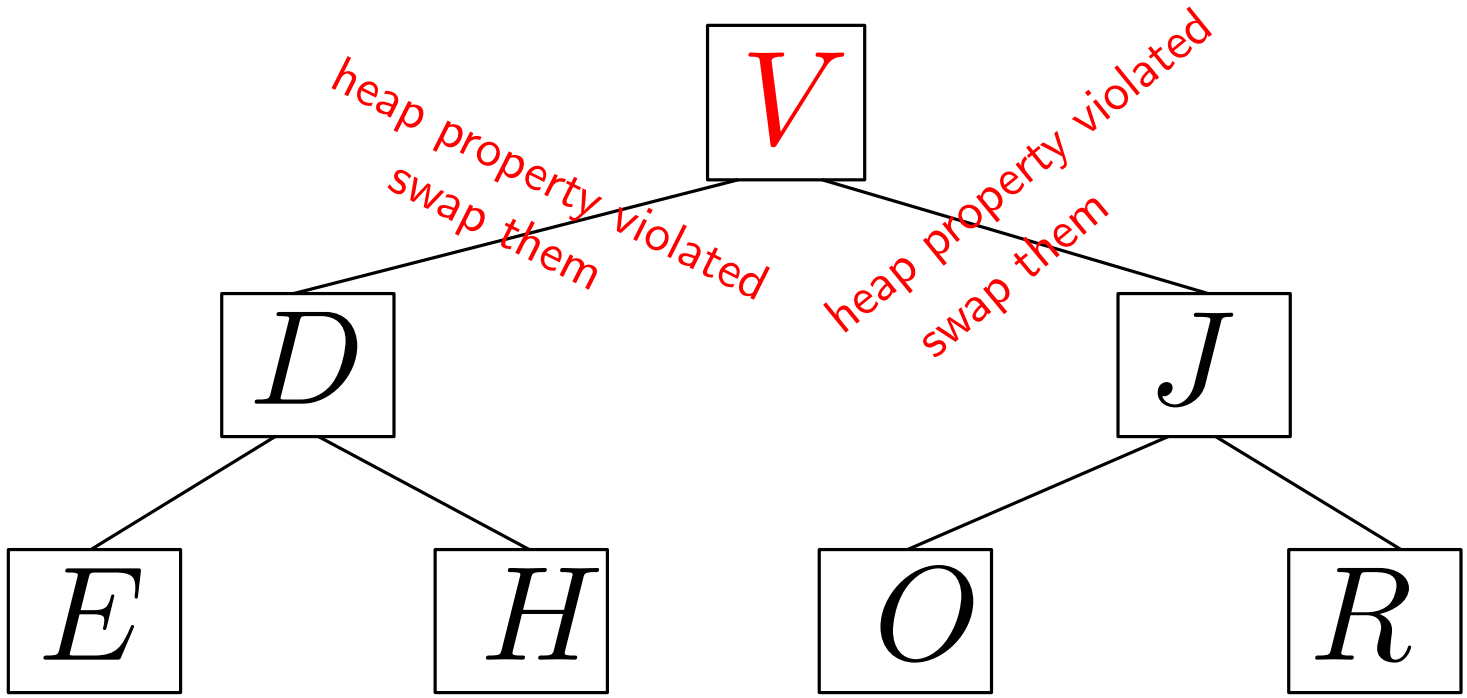
C

Heap Property: Children come later than their parent (alphabetically)



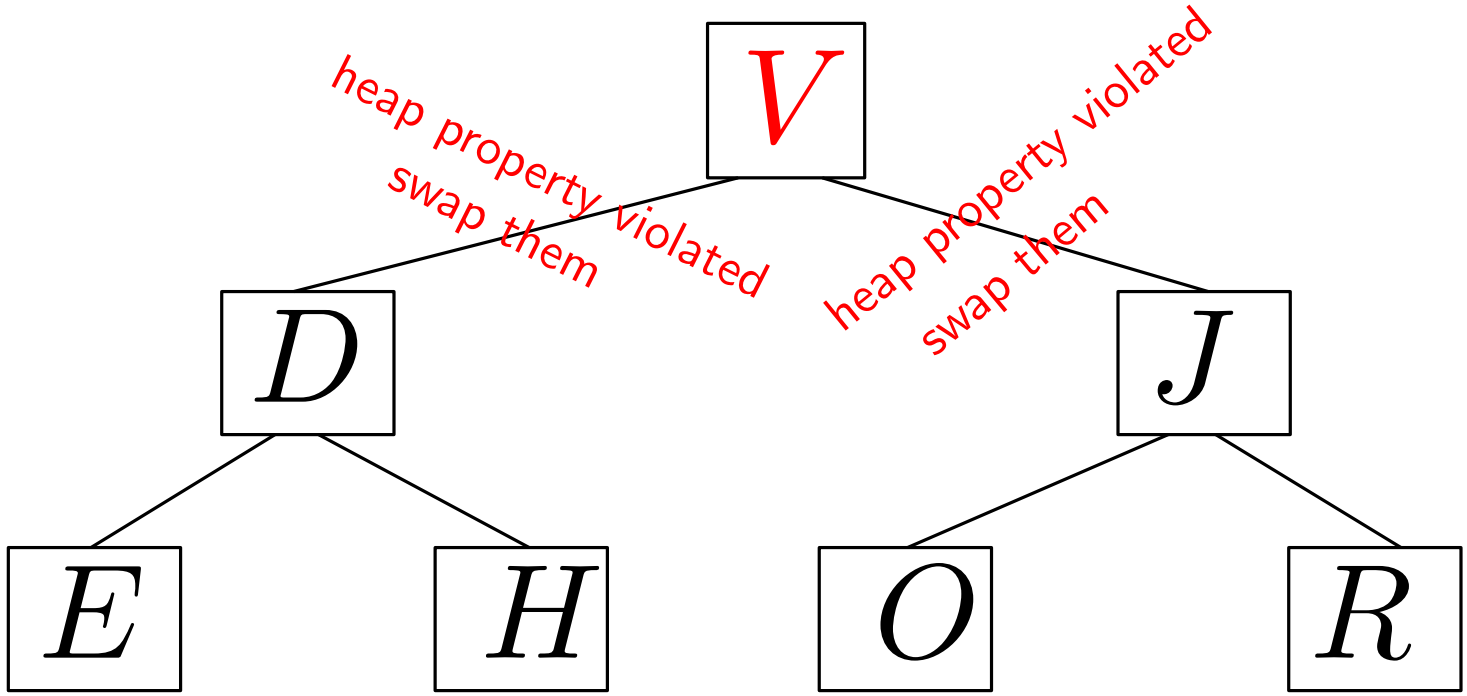
C

Heap Property: Children come later than their parent (alphabetically)



C

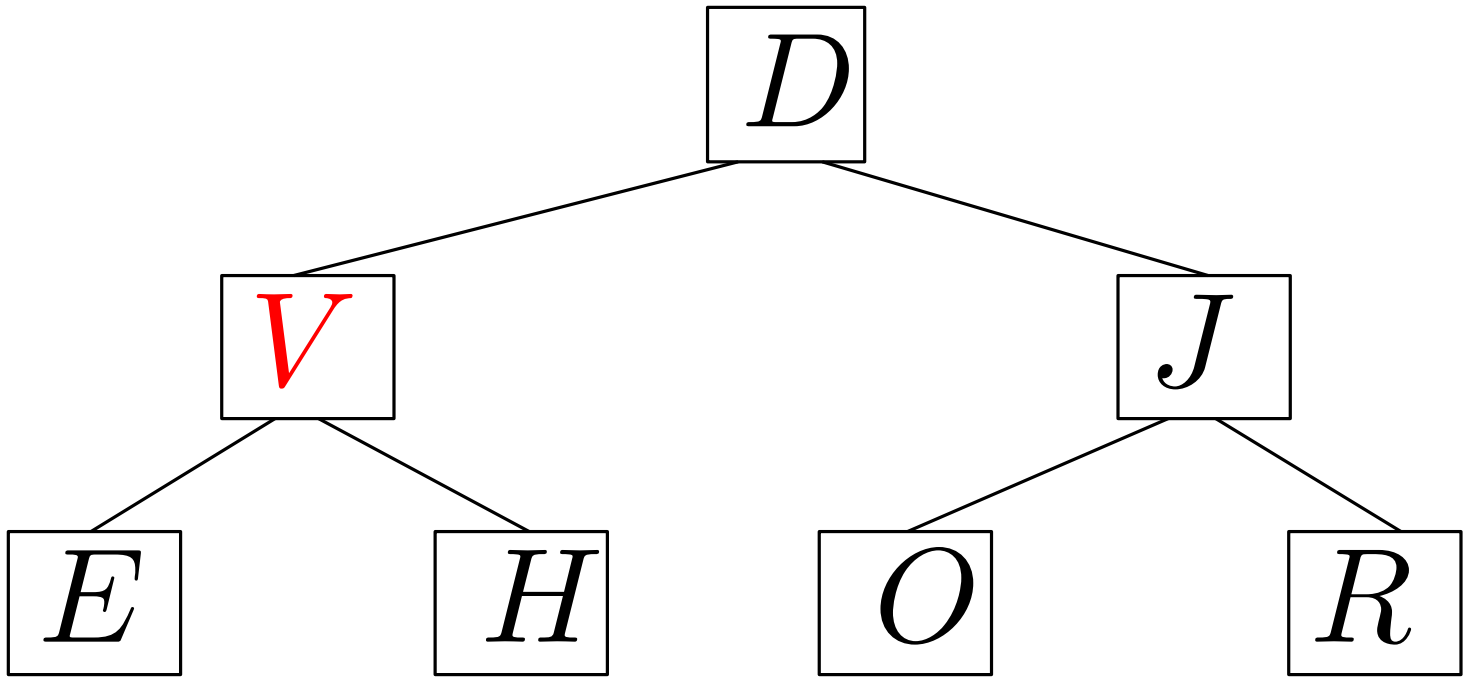
Heap Property: Children come later than their parent (alphabetically)



If root is greater than both children, swap it with the smaller child.

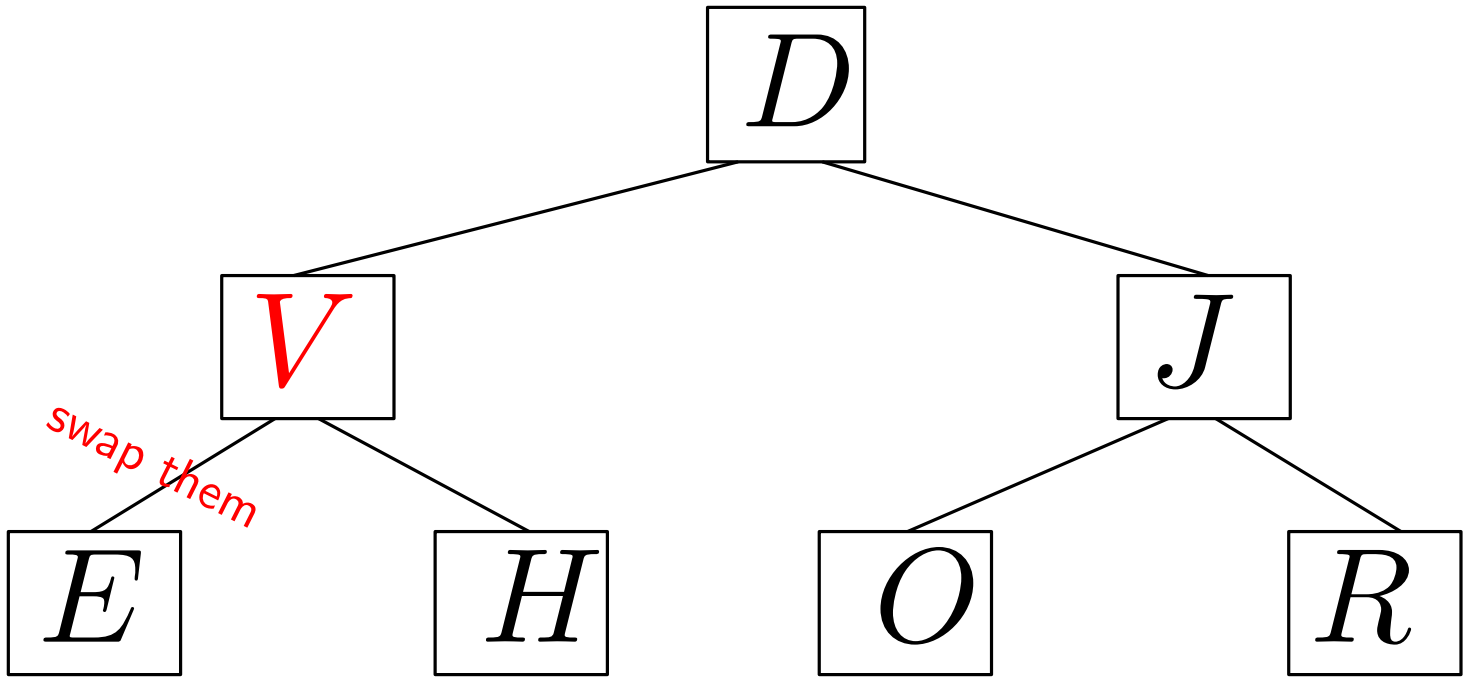
C

Heap Property: Children come later than their parent (alphabetically)



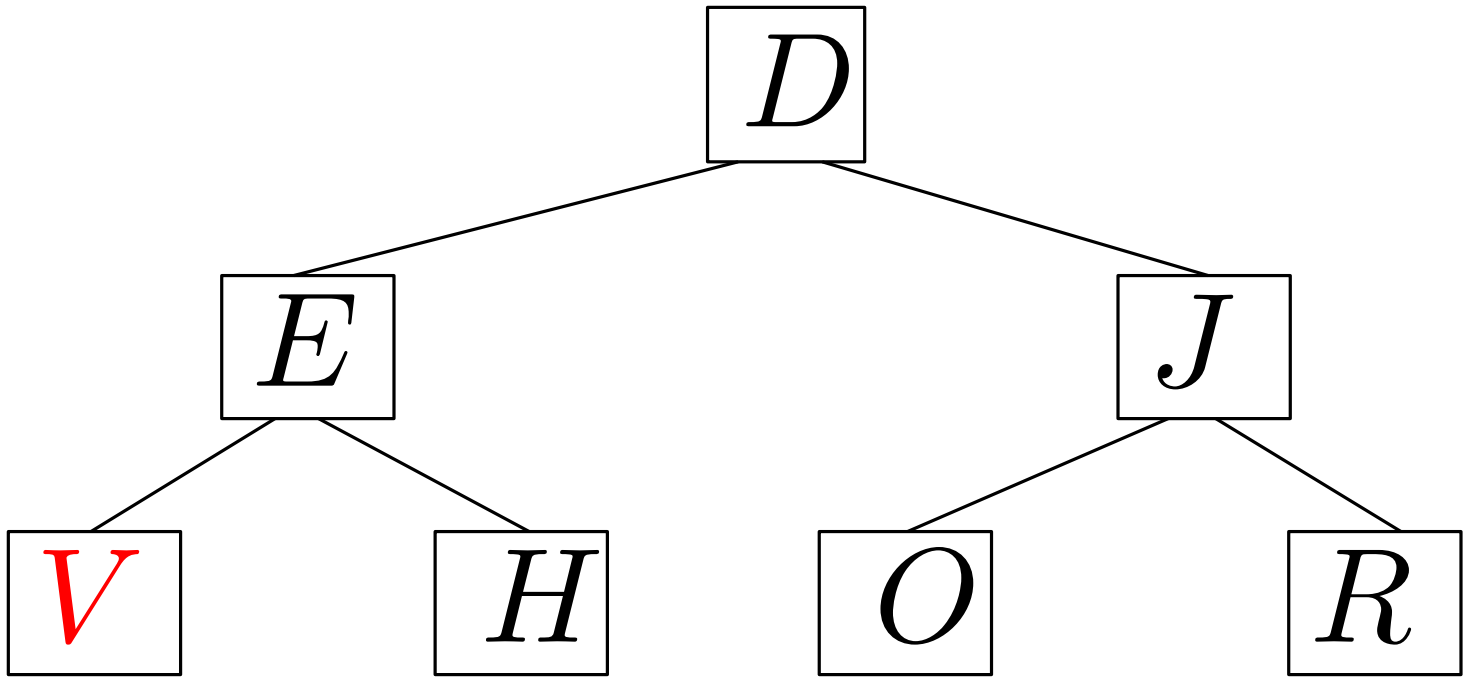
C

Heap Property: Children come later than their parent (alphabetically)



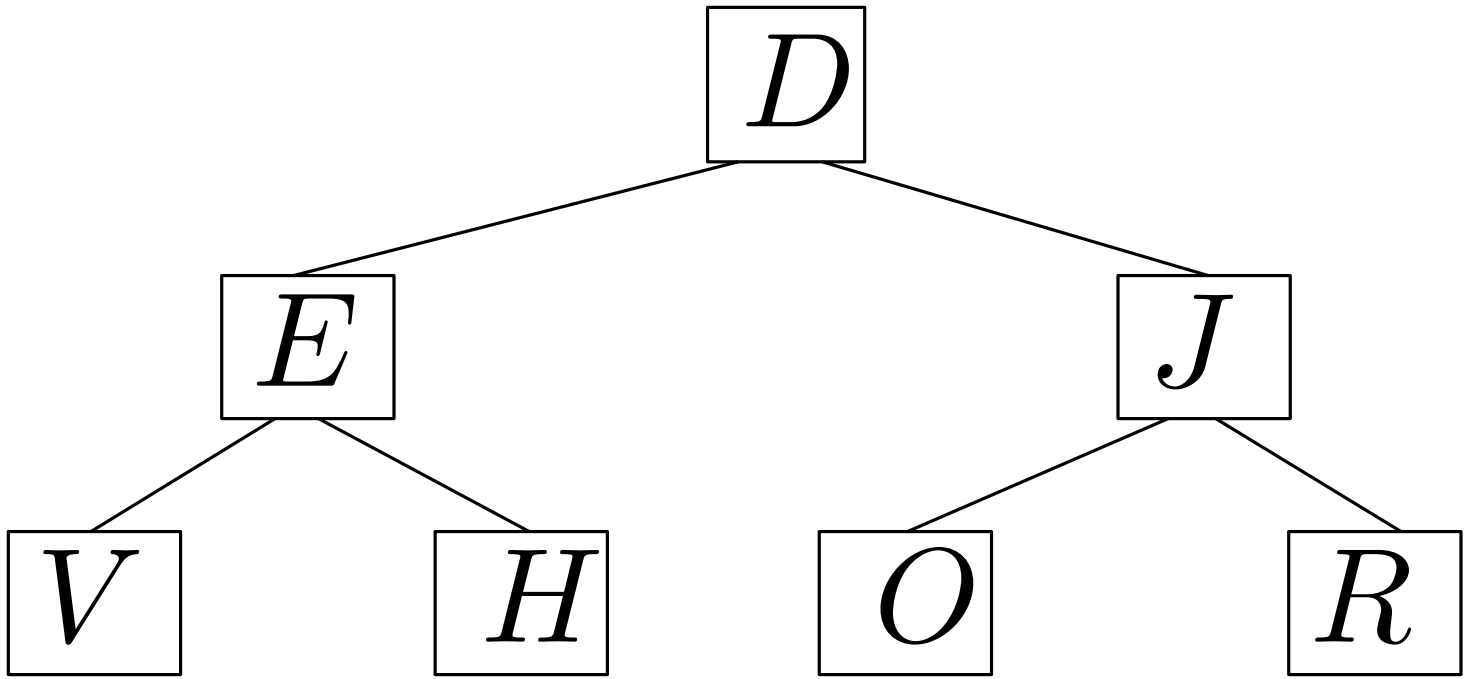
C

Heap Property: Children come later than their parent (alphabetically)



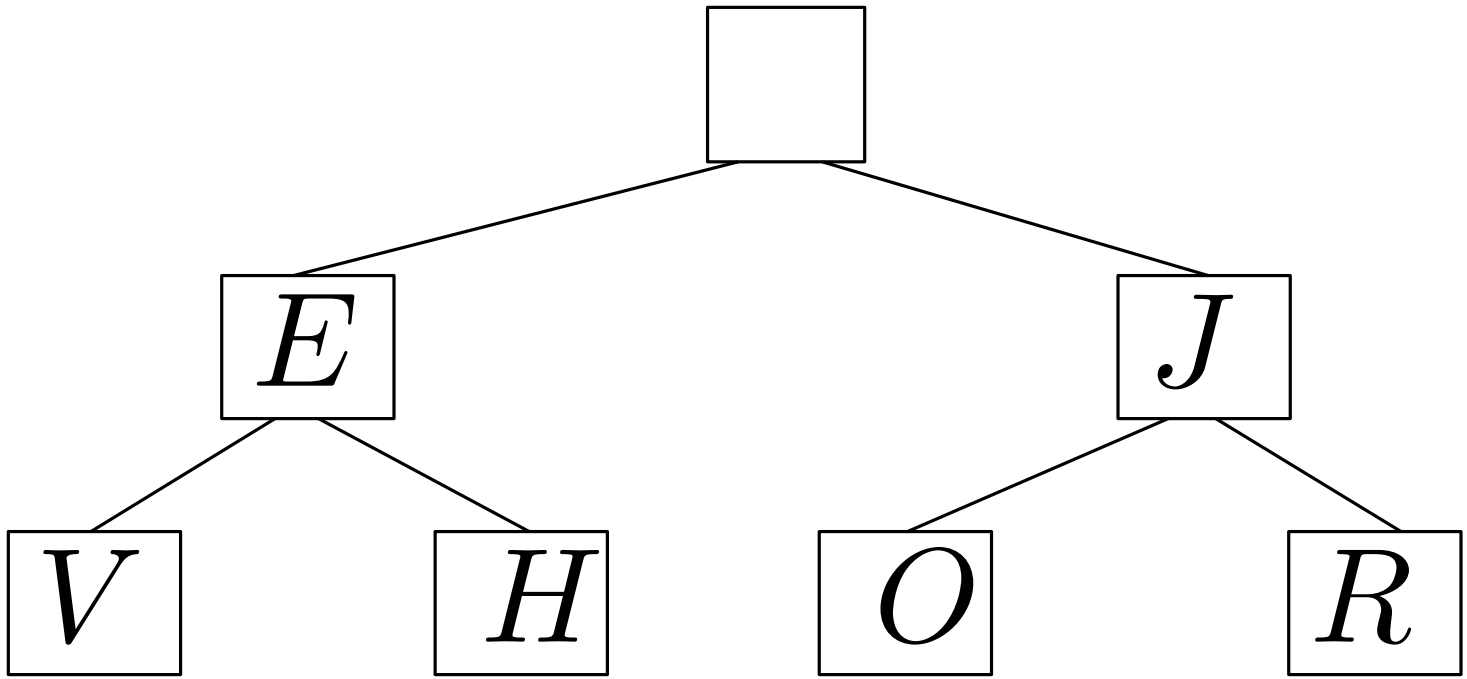
C

Heap Property: Children come later than their parent (alphabetically)



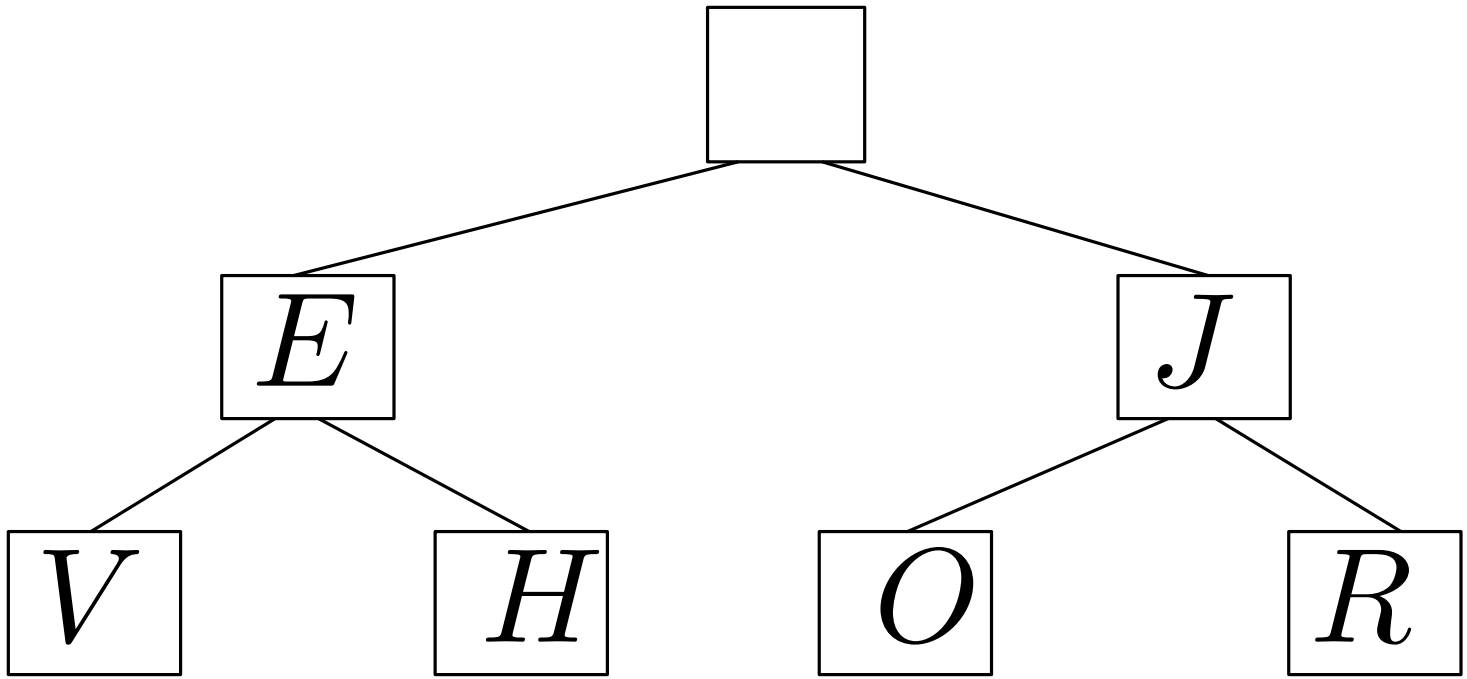
C

Heap Property: Children come later than their parent (alphabetically)



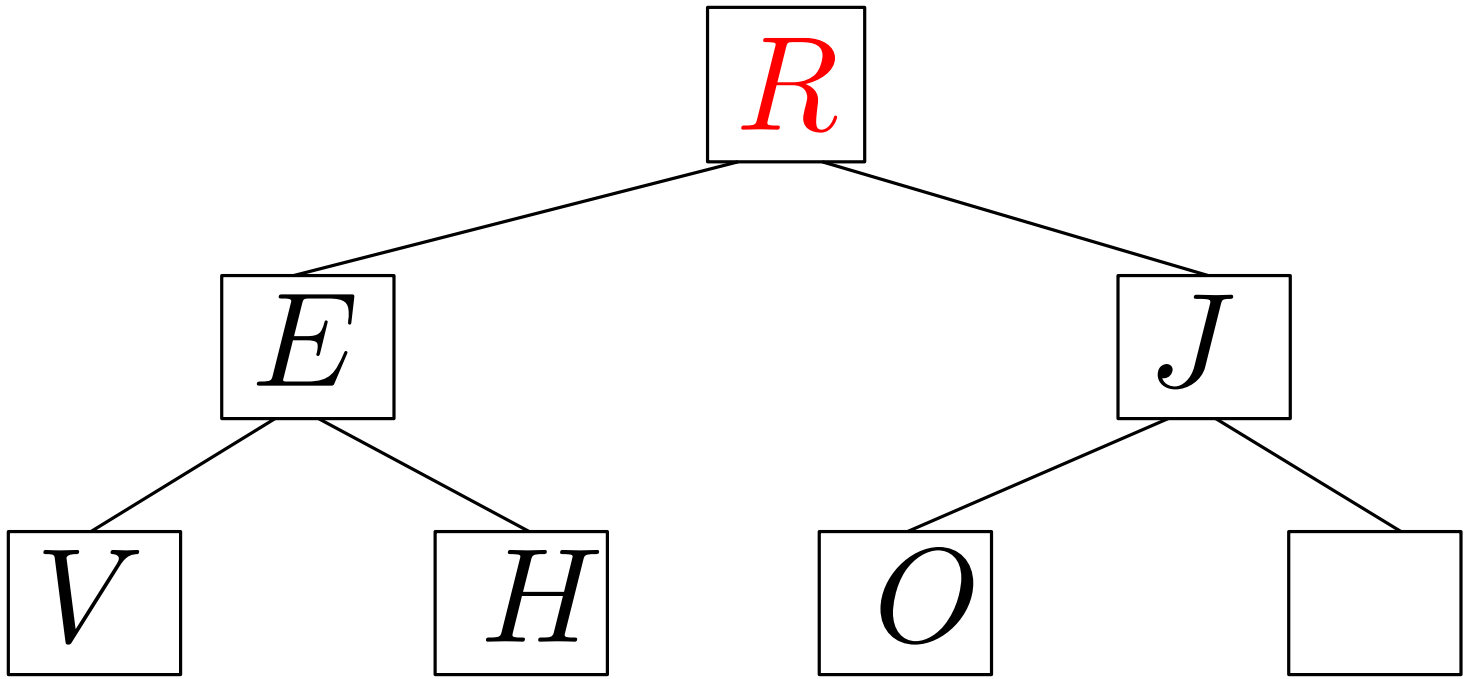
$C \quad D$

Heap Property: Children come later than their parent (alphabetically)



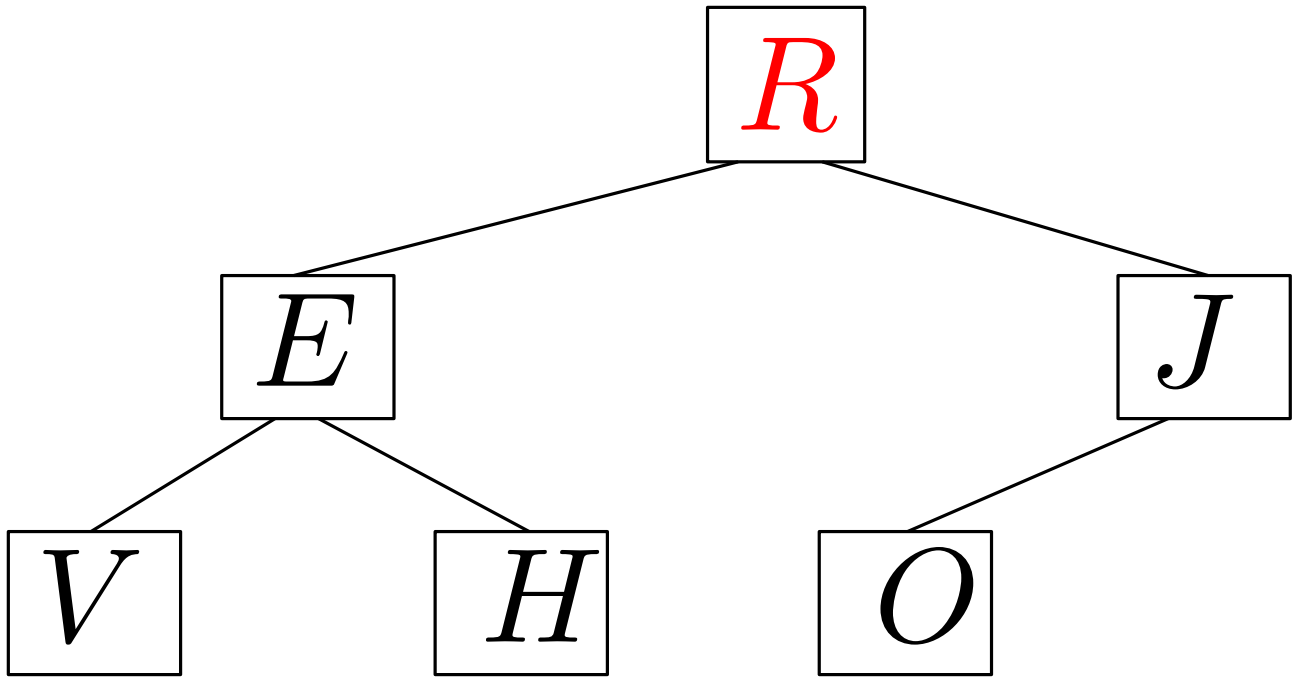
$C \quad D$

Heap Property: Children come later than their parent (alphabetically)



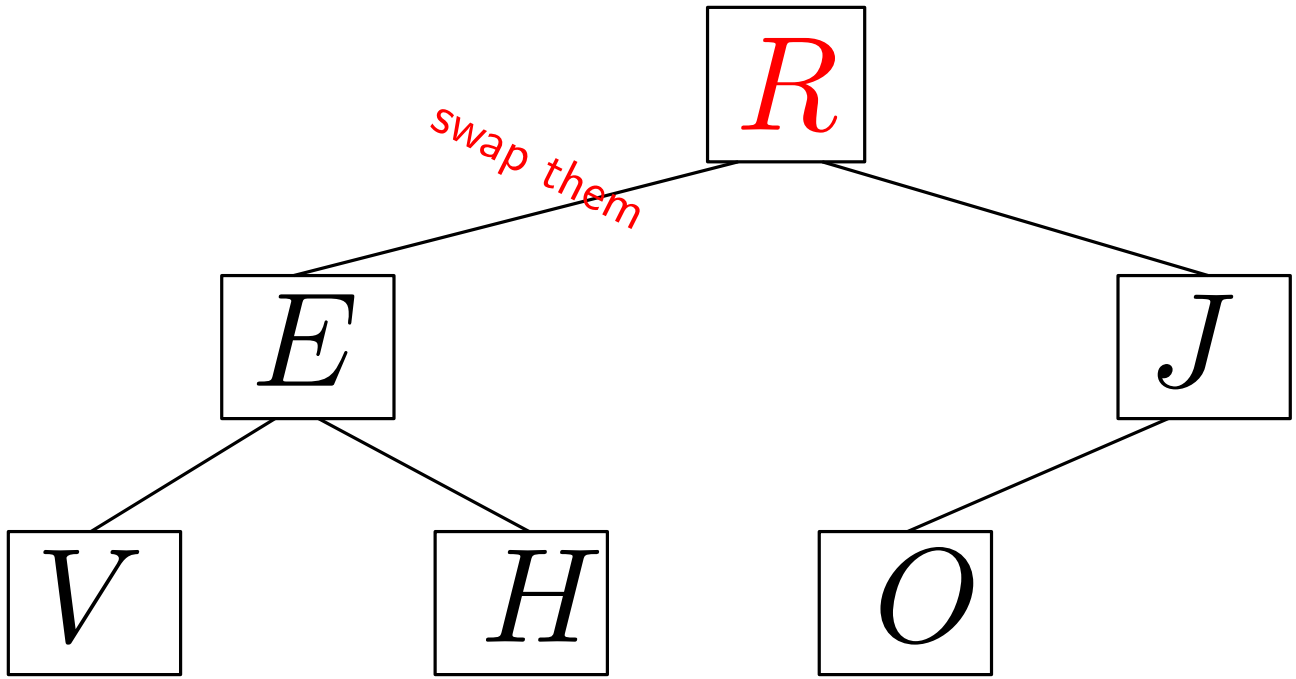
$C \quad D$

Heap Property: Children come later than their parent (alphabetically)



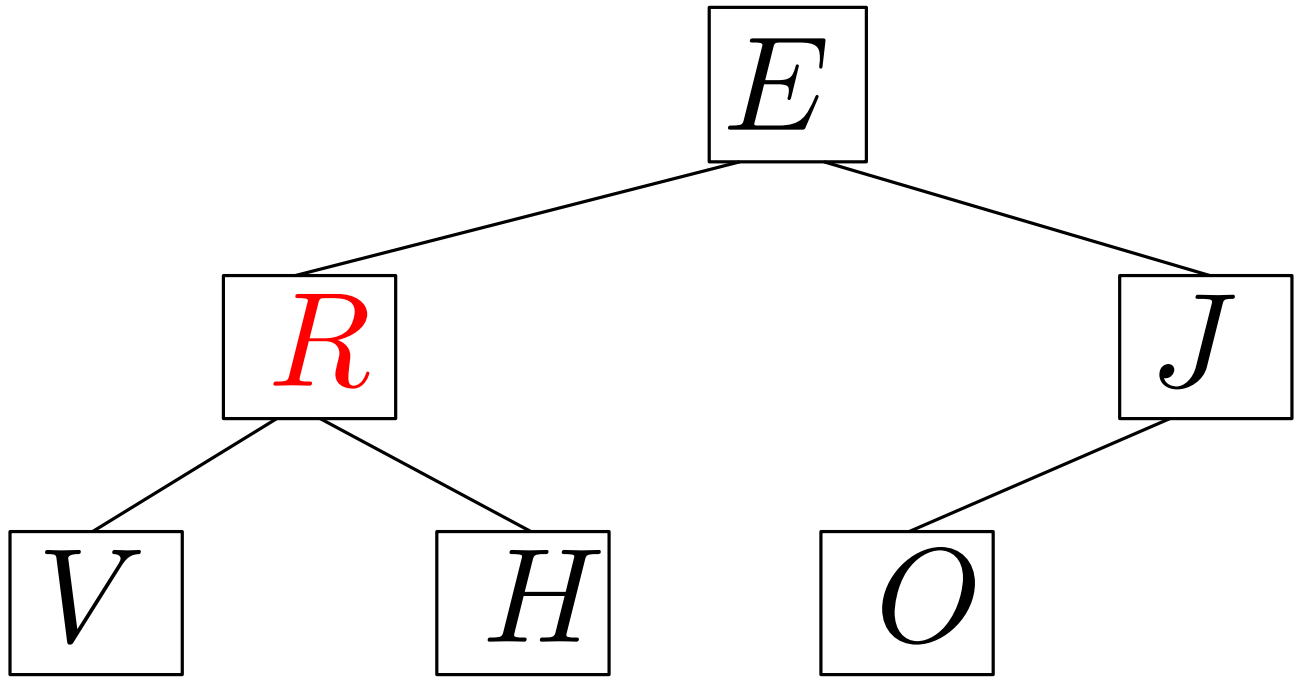
$C \quad D$

Heap Property: Children come later than their parent (alphabetically)



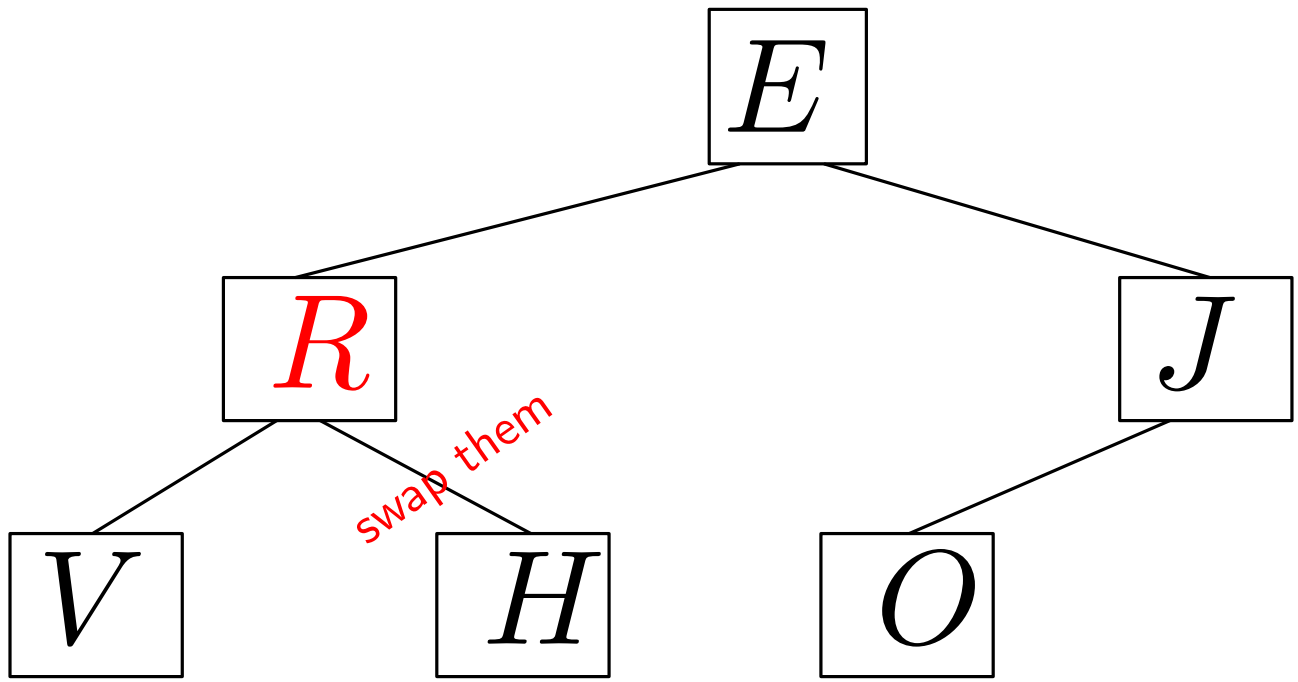
$C \quad D$

Heap Property: Children come later than their parent (alphabetically)



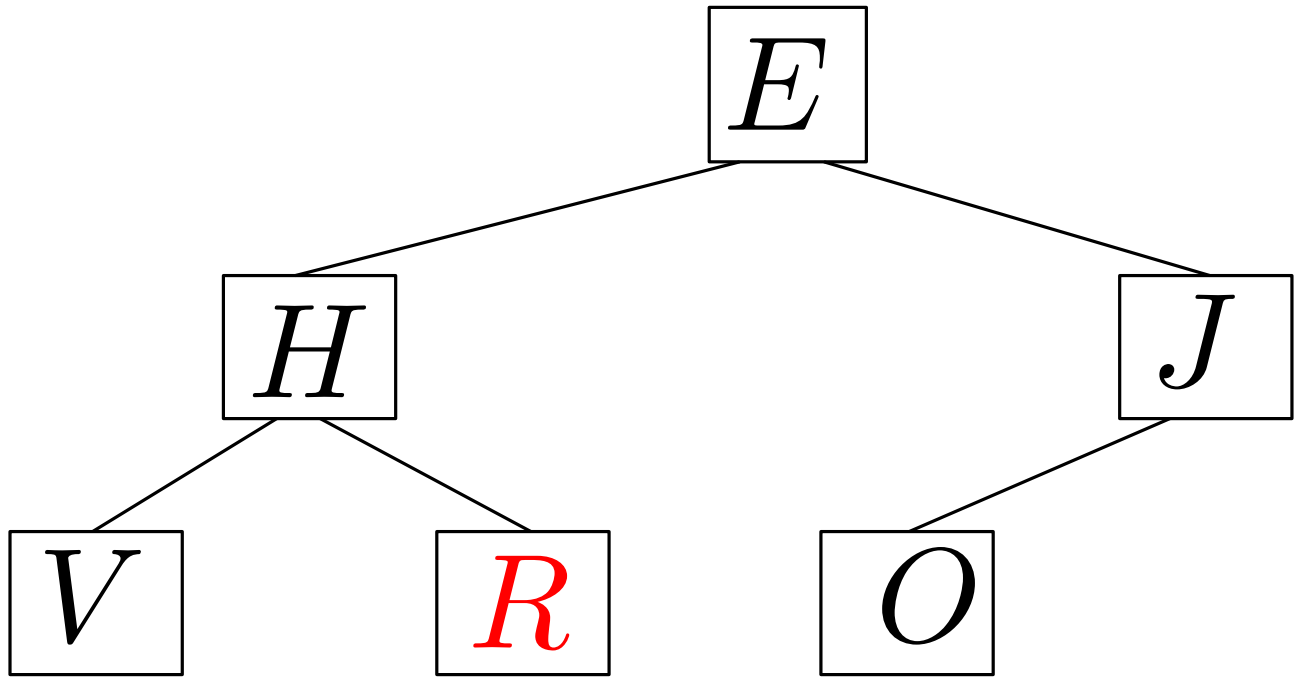
$C \quad D$

Heap Property: Children come later than their parent (alphabetically)



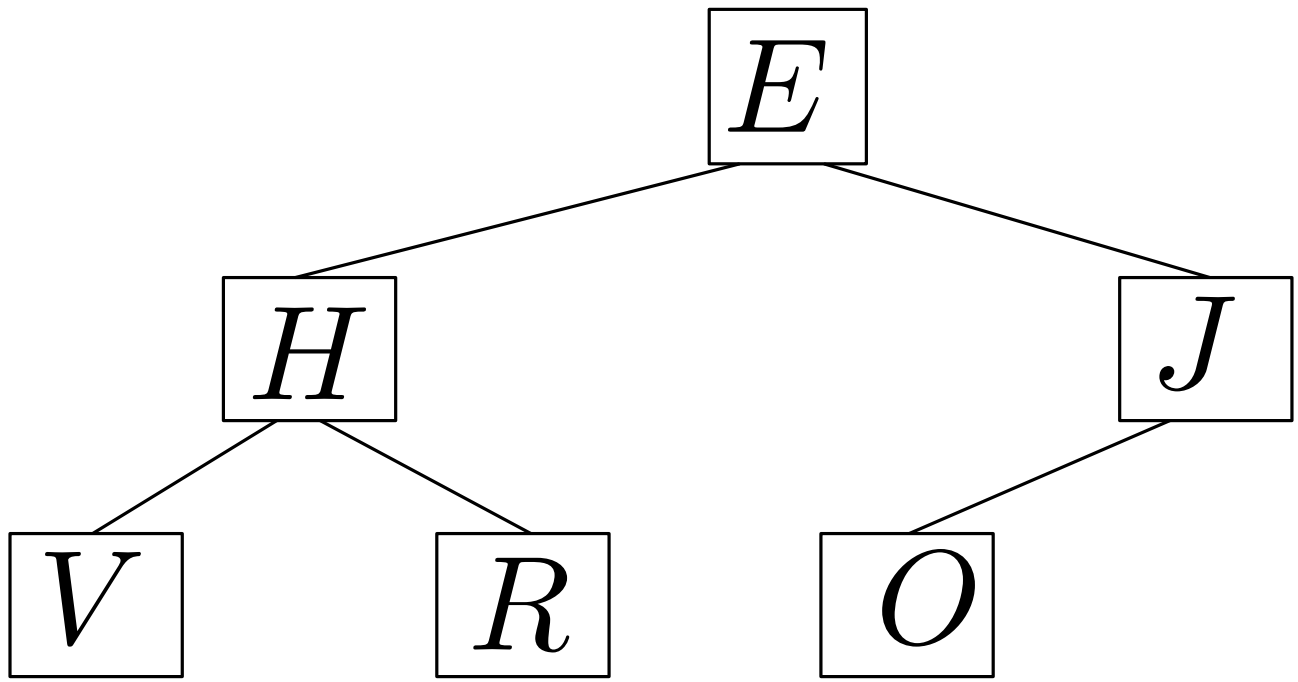
$C \quad D$

Heap Property: Children come later than their parent (alphabetically)



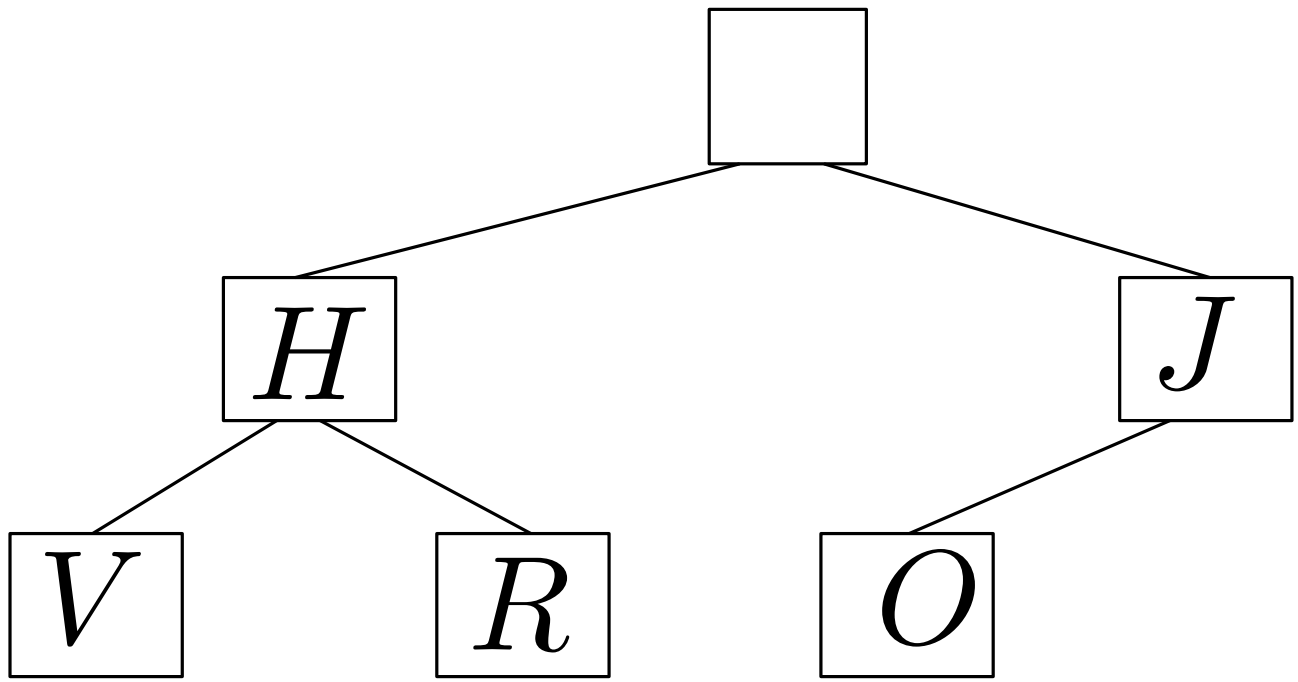
$C \quad D$

Heap Property: Children come later than their parent (alphabetically)



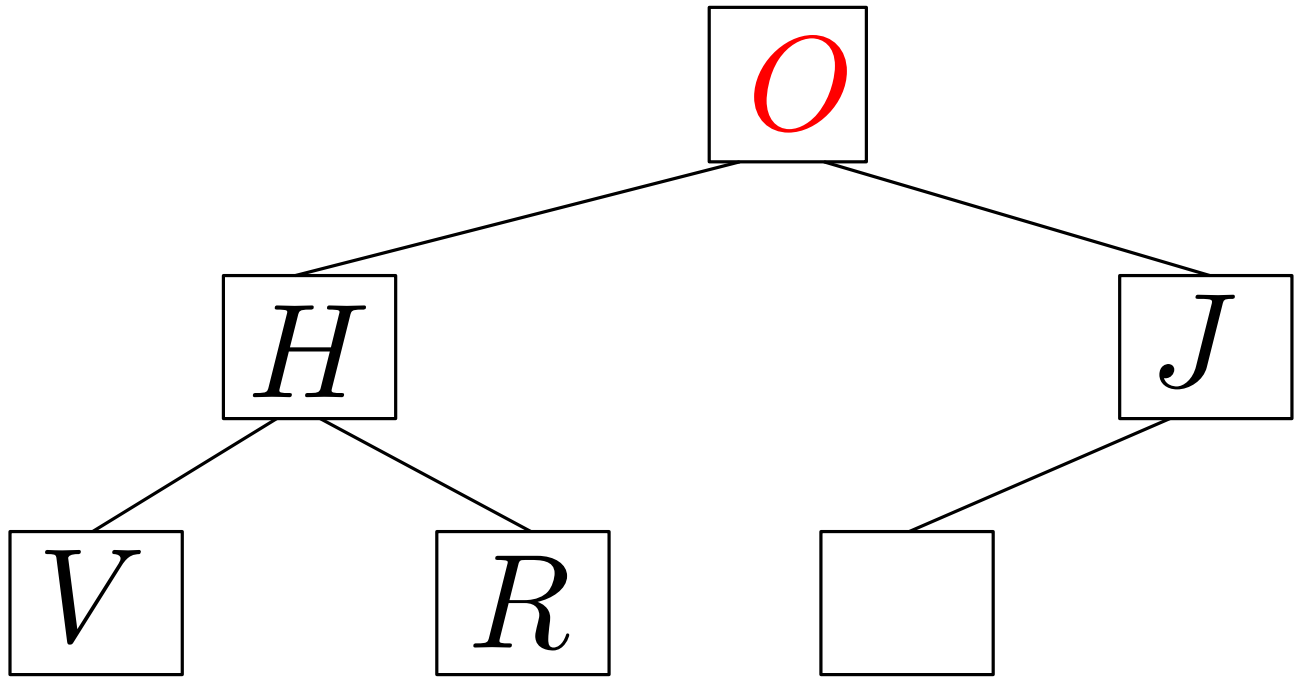
$C \quad D$

Heap Property: Children come later than their parent (alphabetically)



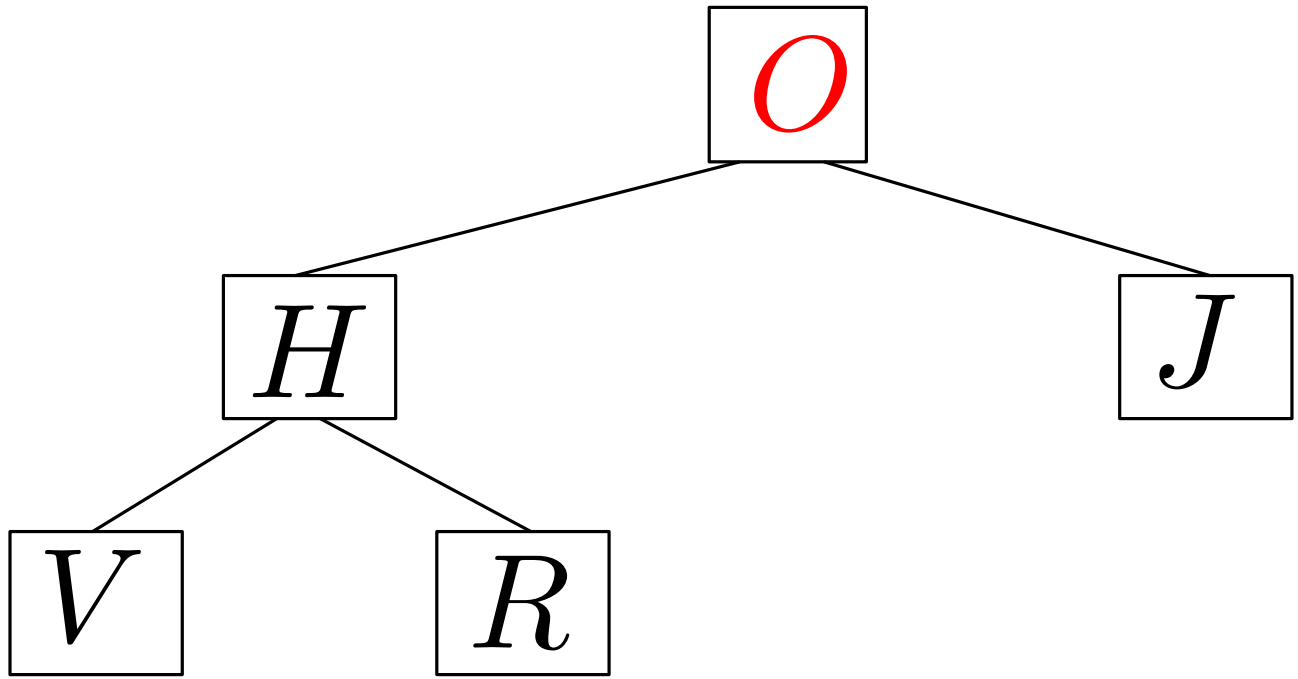
$C \quad D \quad E$

Heap Property: Children come later than their parent (alphabetically)



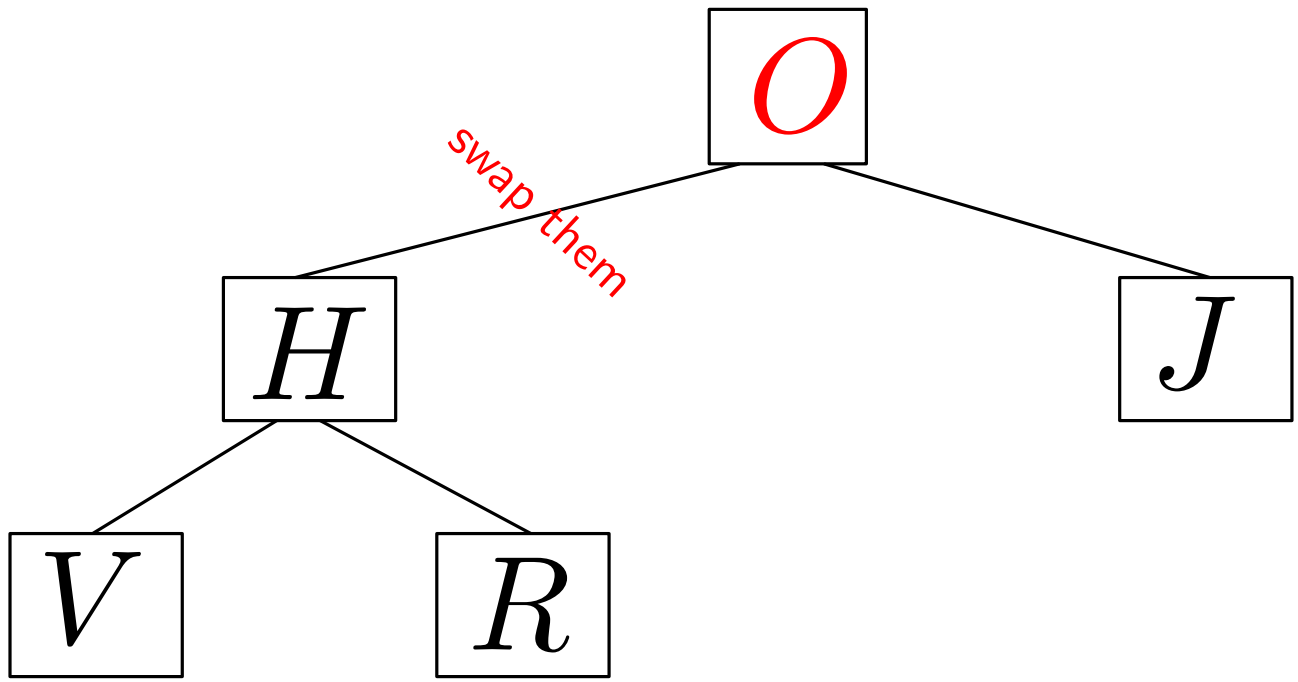
$C \quad D \quad E$

Heap Property: Children come later than their parent (alphabetically)



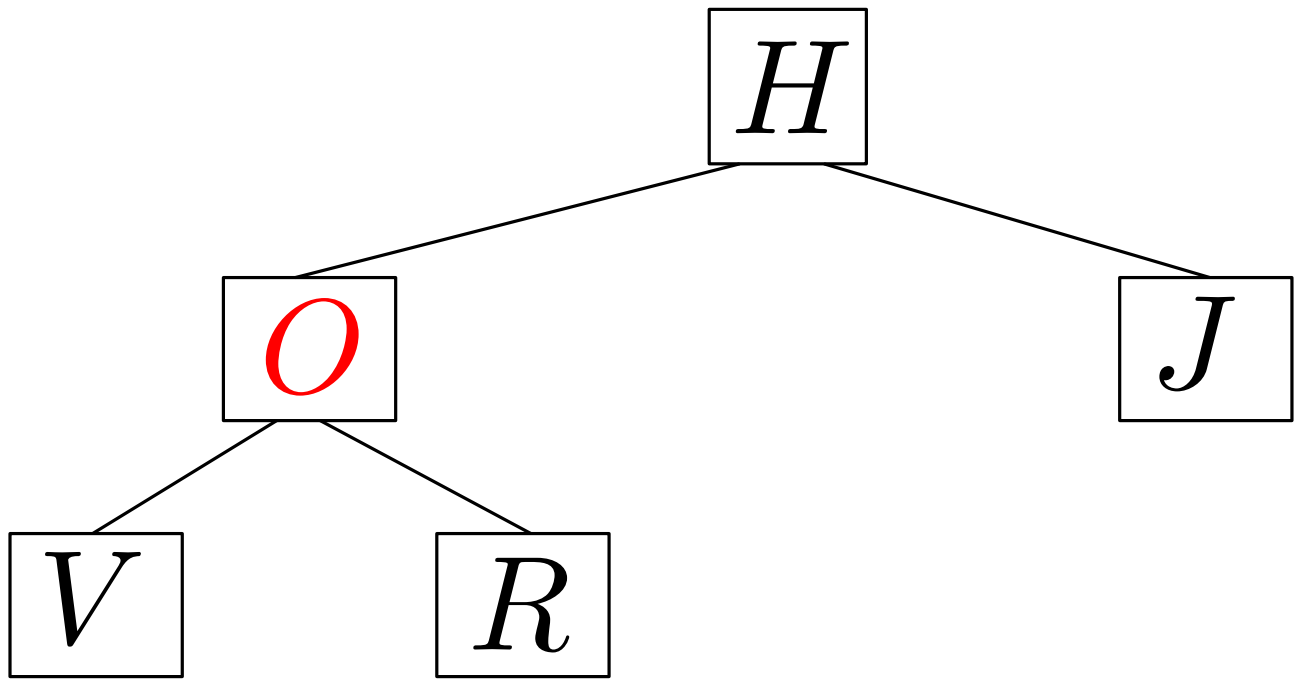
$C \quad D \quad E$

Heap Property: Children come later than their parent (alphabetically)



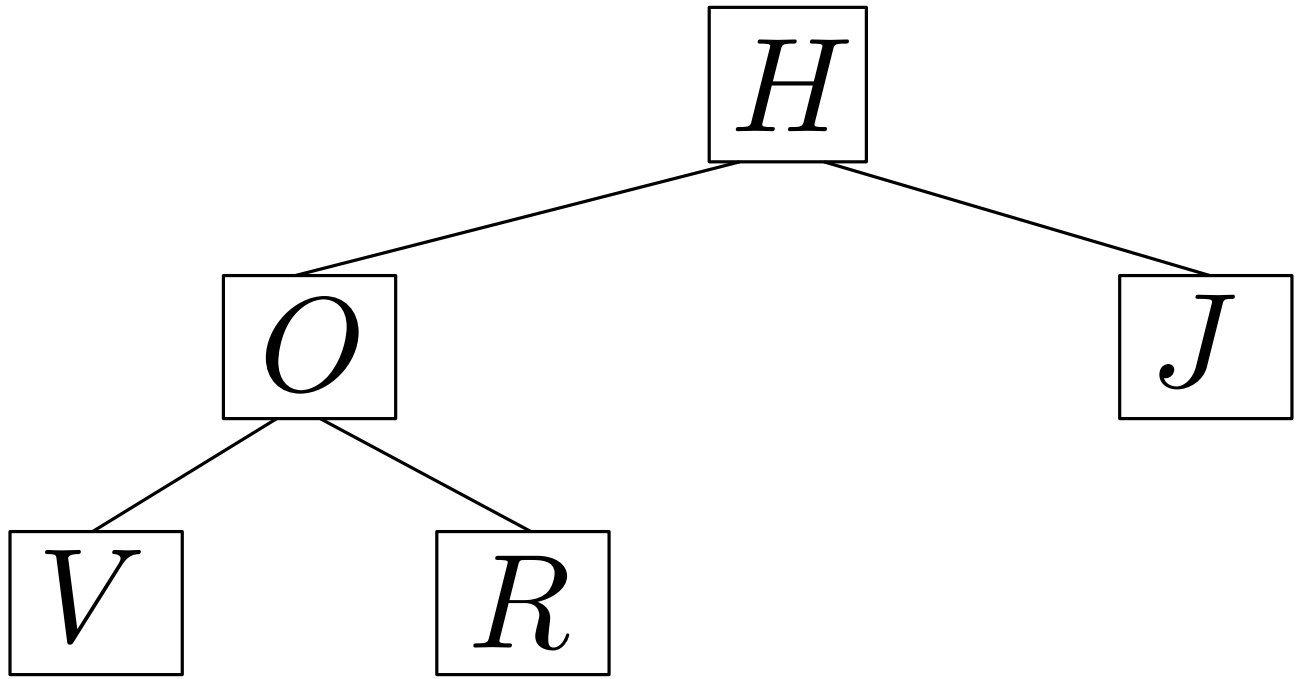
$C \quad D \quad E$

Heap Property: Children come later than their parent (alphabetically)



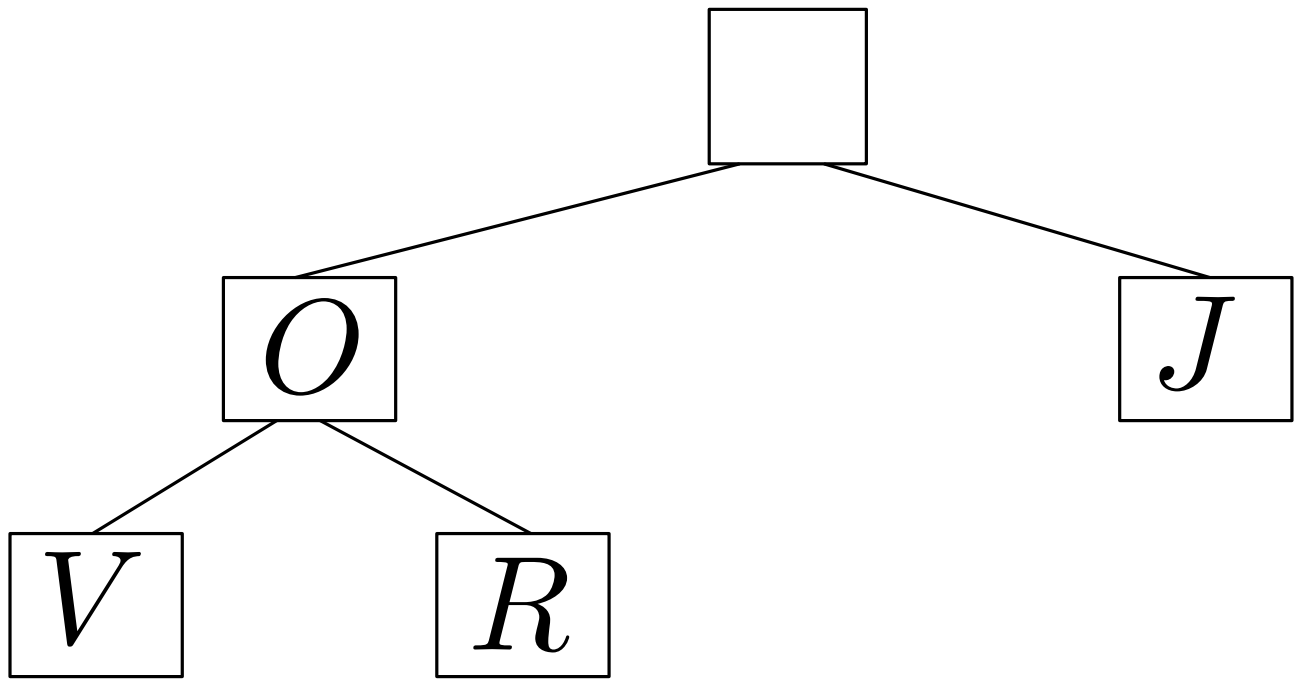
$C \quad D \quad E$

Heap Property: Children come later than their parent (alphabetically)



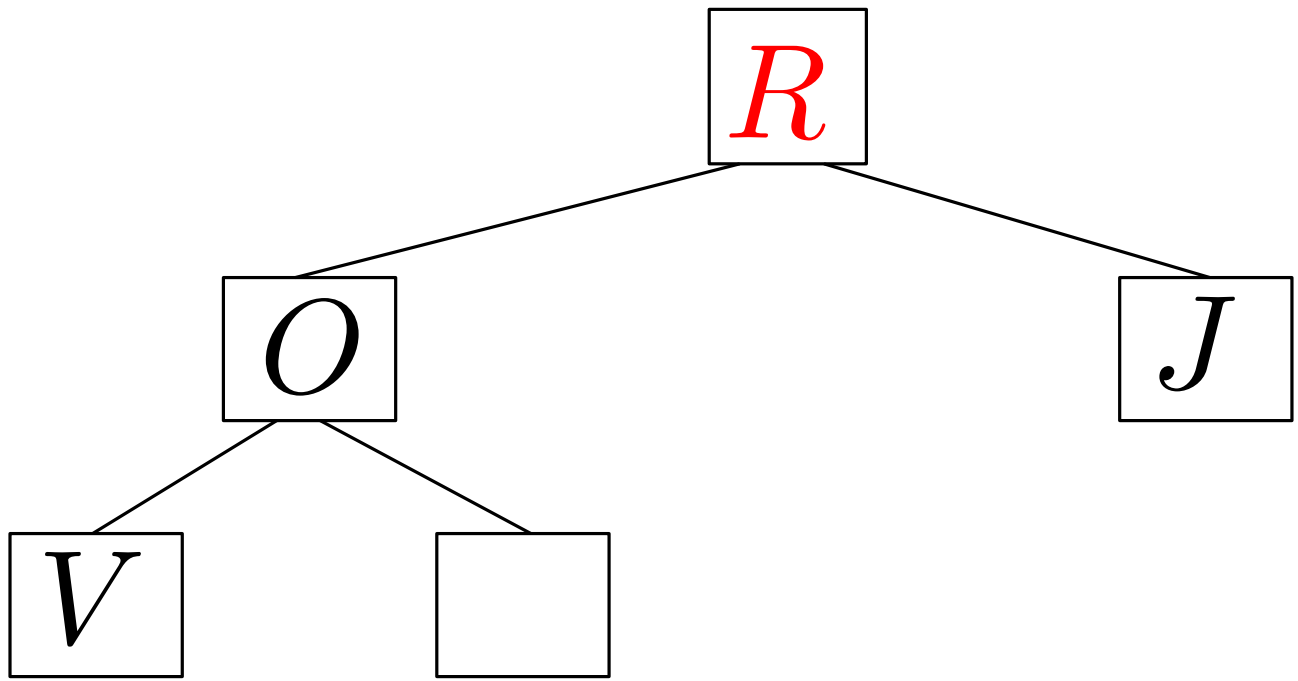
$C \quad D \quad E$

Heap Property: Children come later than their parent (alphabetically)



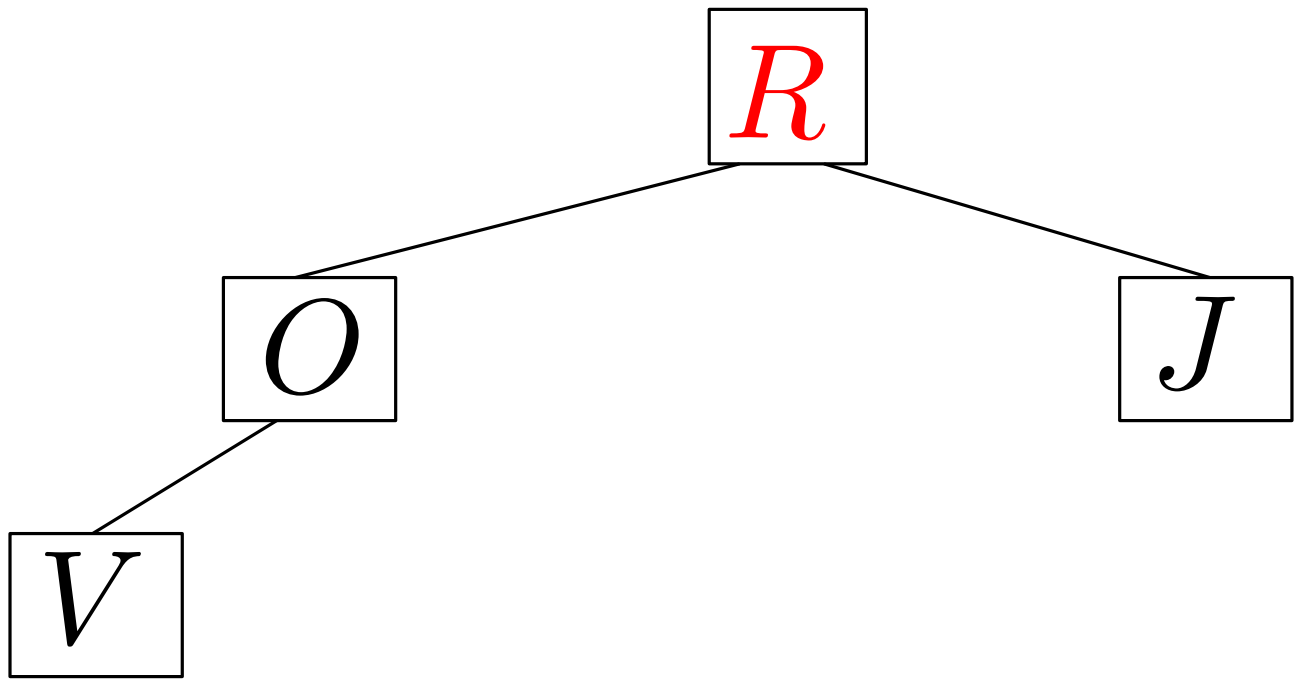
$C \quad D \quad E \quad H$

Heap Property: Children come later than their parent (alphabetically)



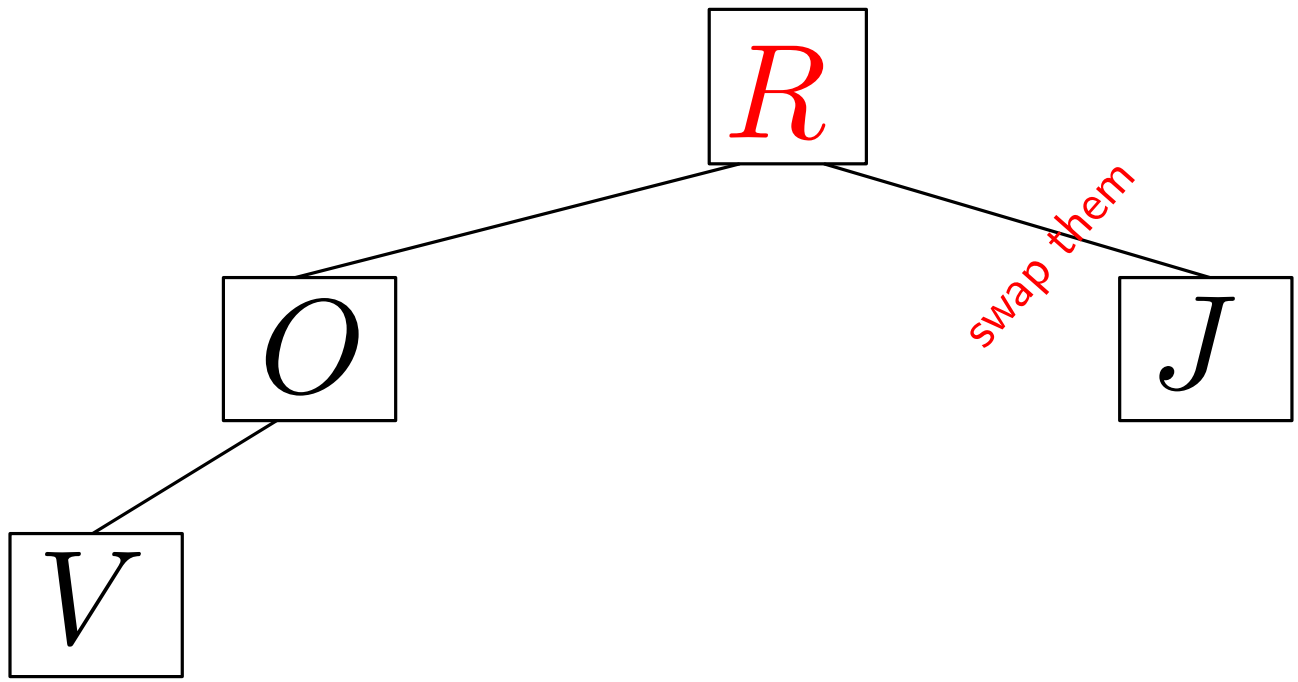
$C \quad D \quad E \quad H$

Heap Property: Children come later than their parent (alphabetically)



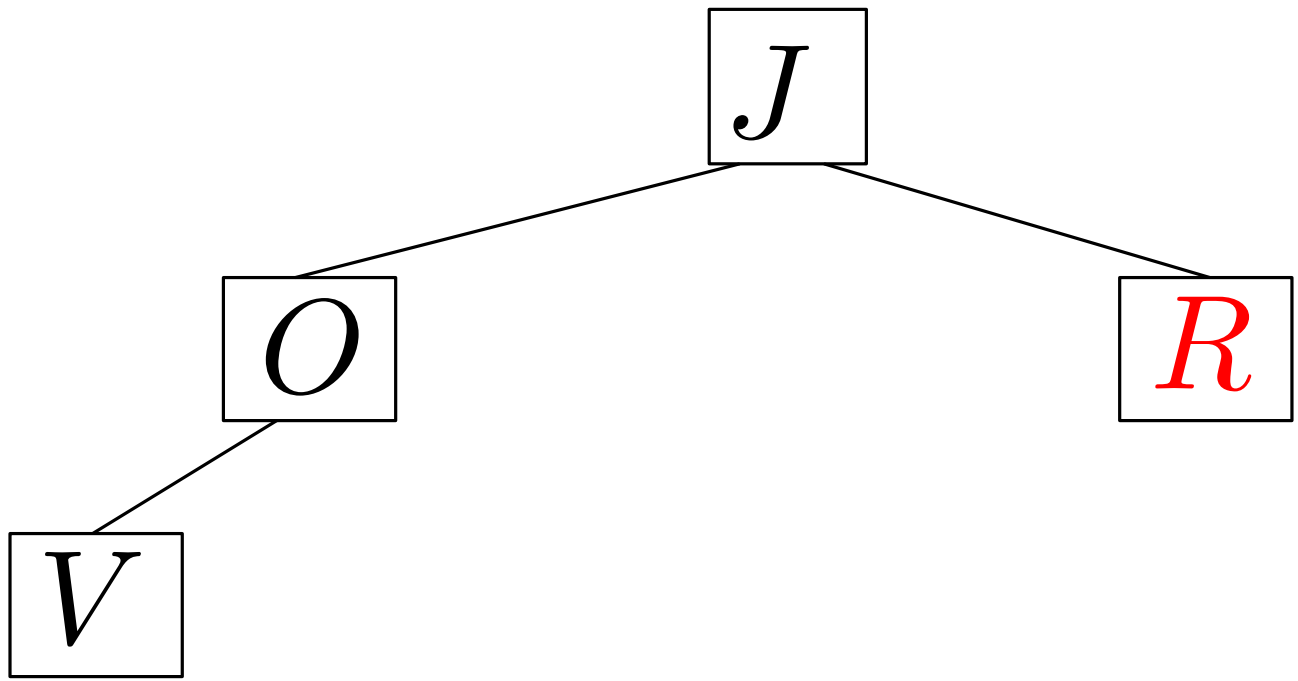
$C \quad D \quad E \quad H$

Heap Property: Children come later than their parent (alphabetically)



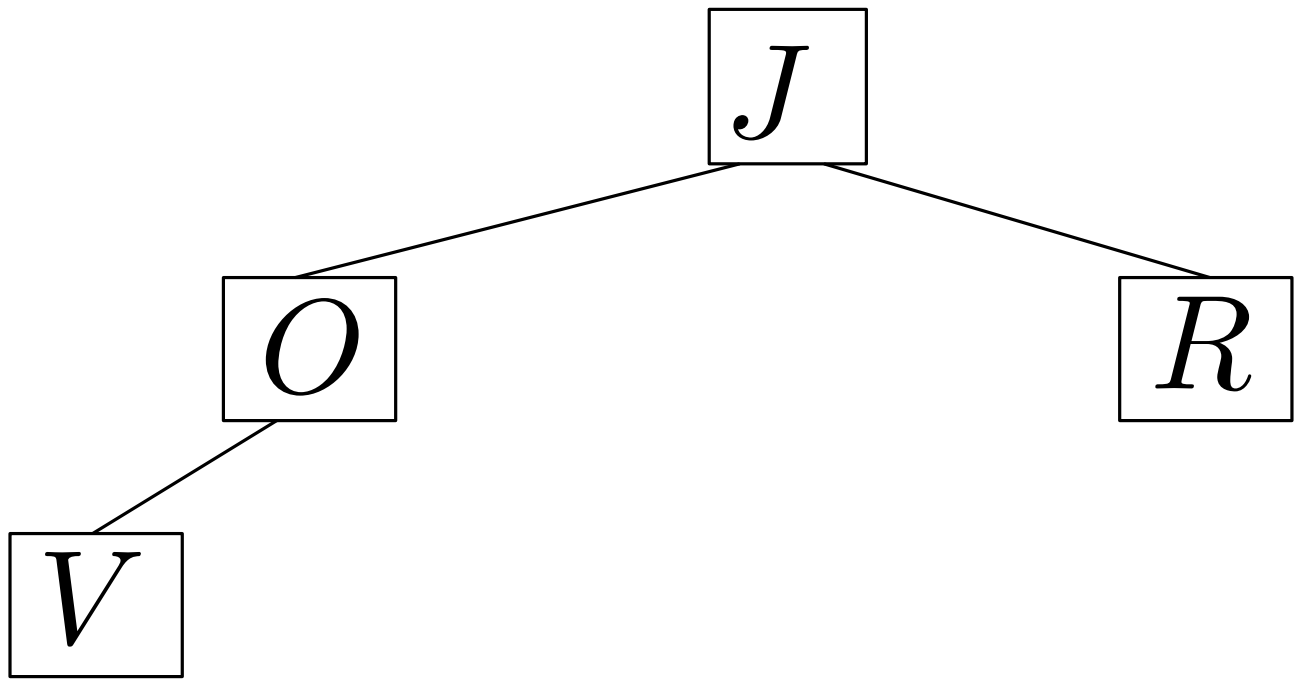
$C \quad D \quad E \quad H$

Heap Property: Children come later than their parent (alphabetically)



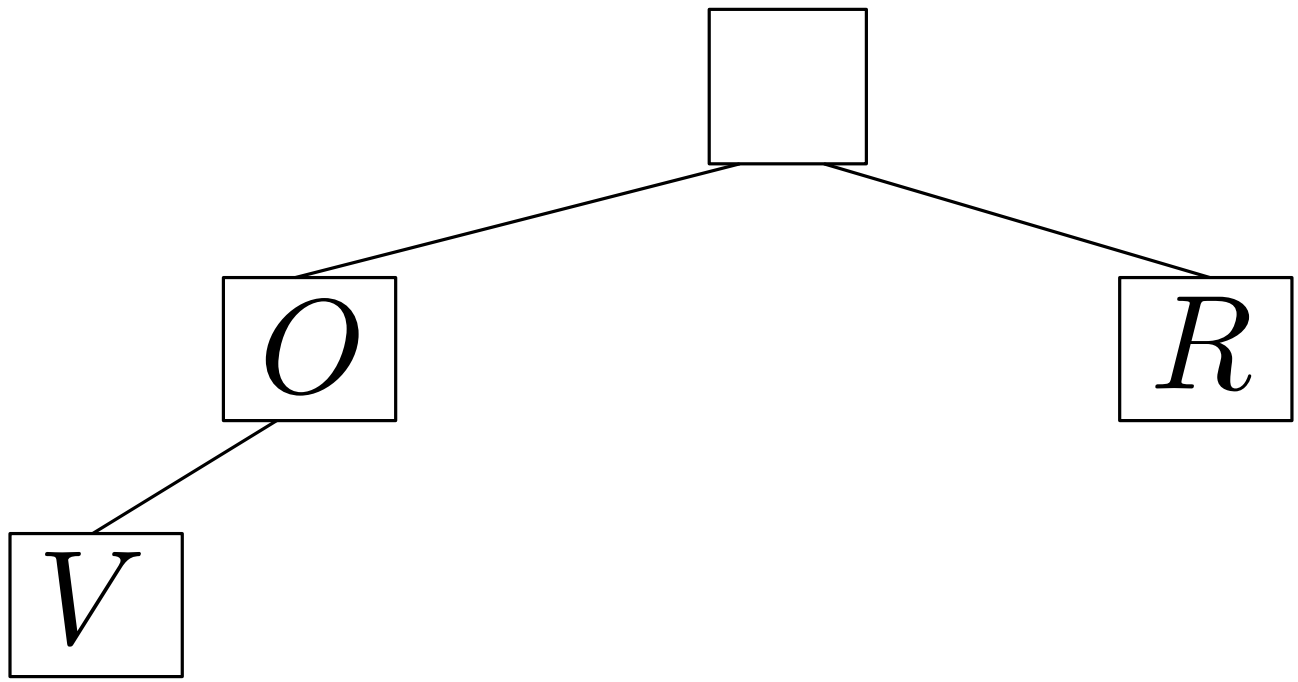
$C \quad D \quad E \quad H$

Heap Property: Children come later than their parent (alphabetically)



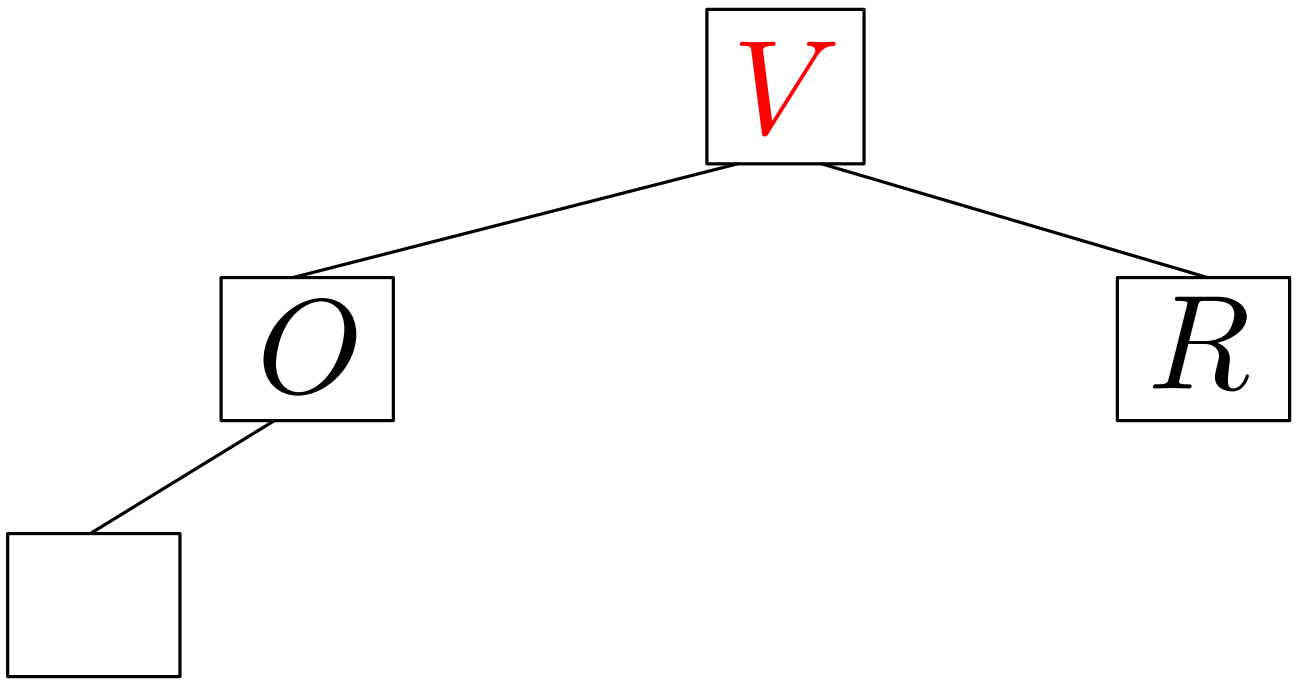
$C \quad D \quad E \quad H$

Heap Property: Children come later than their parent (alphabetically)



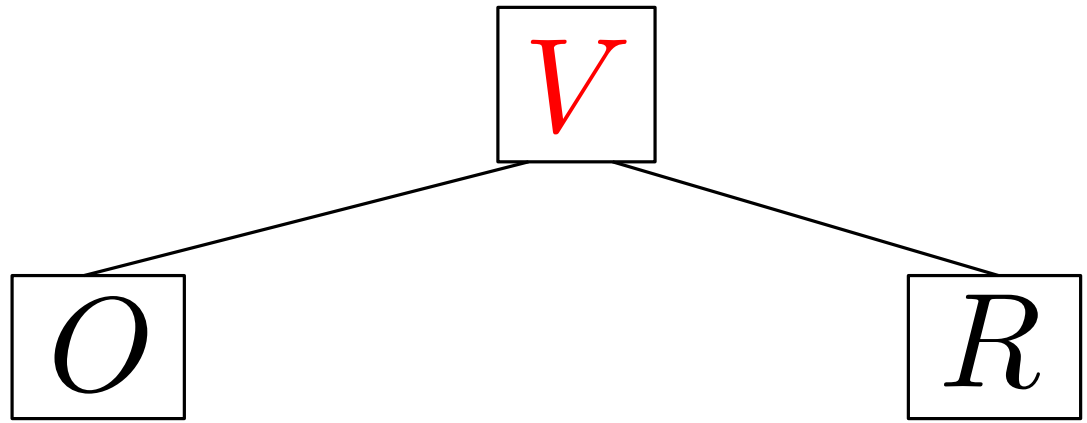
$C \quad D \quad E \quad H \quad J$

Heap Property: Children come later than their parent (alphabetically)



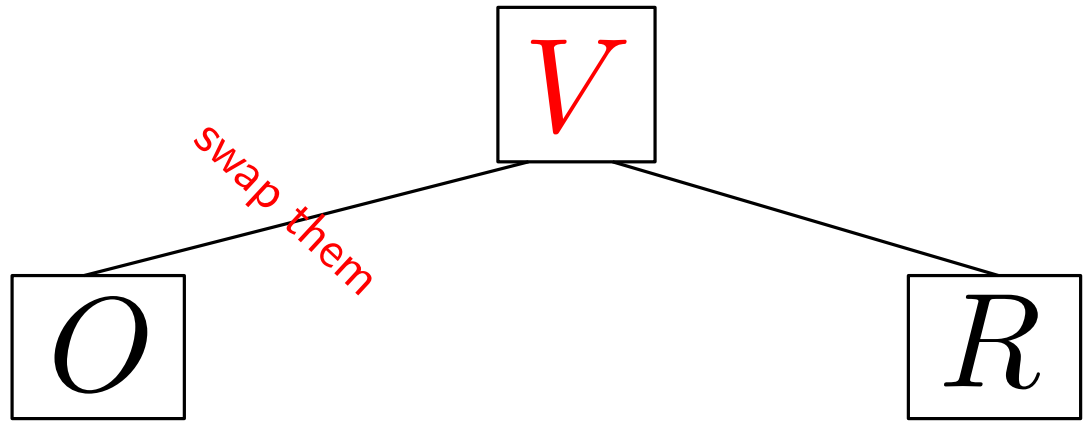
$C \quad D \quad E \quad H \quad J$

Heap Property: Children come later than their parent (alphabetically)



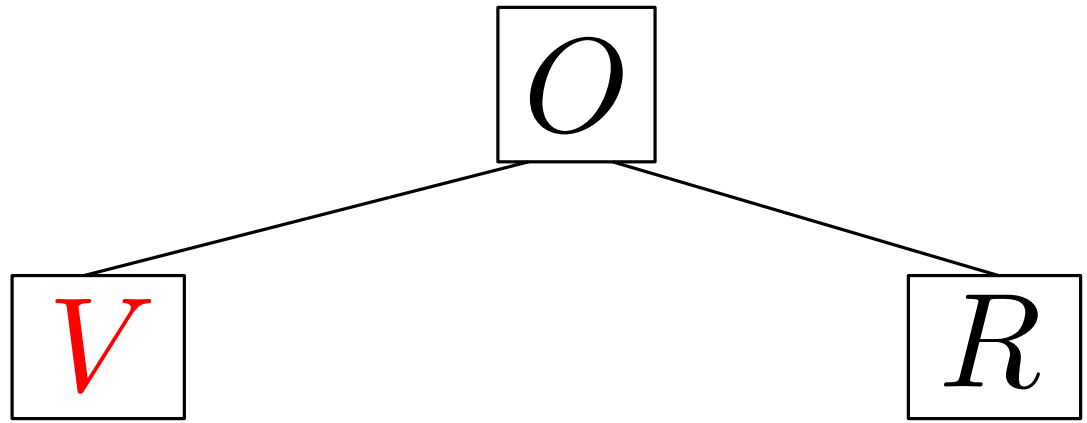
$C \quad D \quad E \quad H \quad J$

Heap Property: Children come later than their parent (alphabetically)



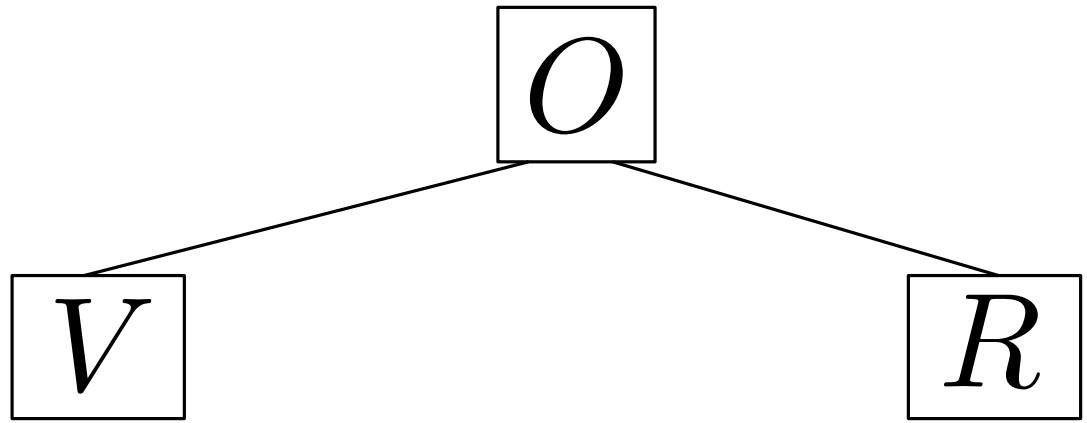
$C \quad D \quad E \quad H \quad J$

Heap Property: Children come later than their parent (alphabetically)



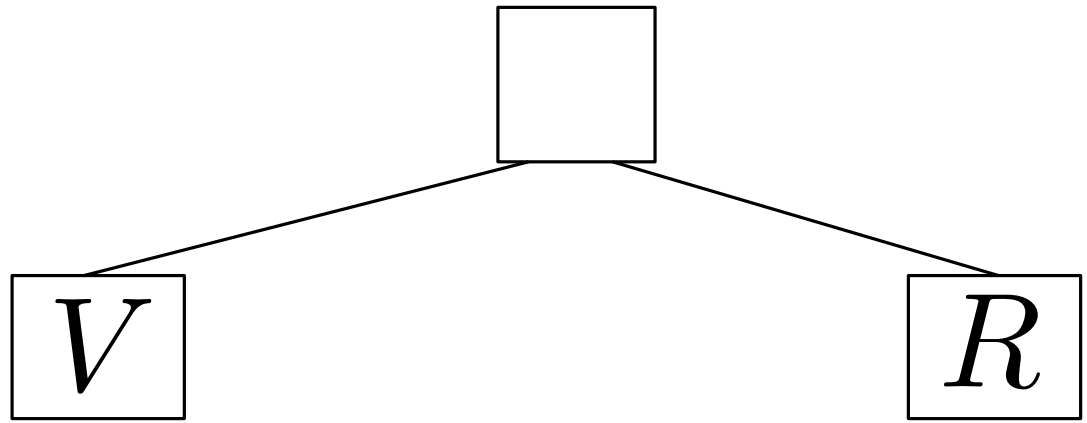
$C \quad D \quad E \quad H \quad J$

Heap Property: Children come later than their parent (alphabetically)



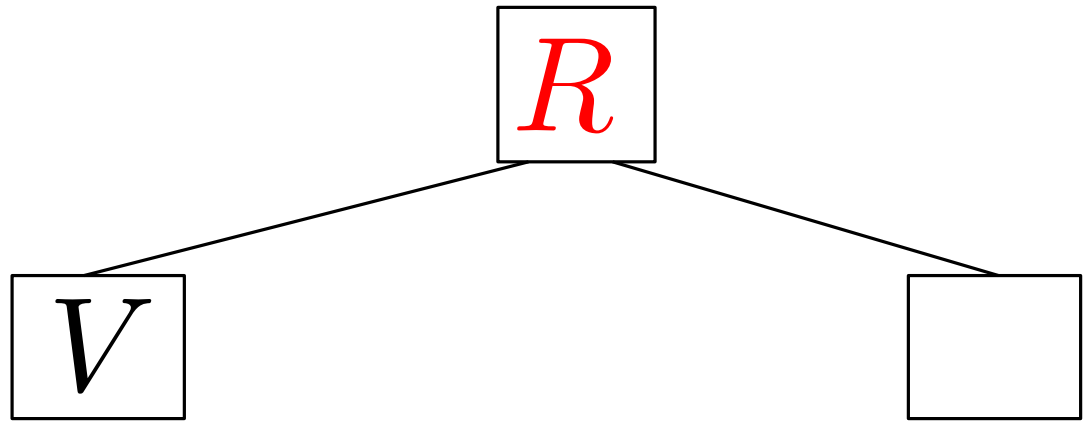
$C \quad D \quad E \quad H \quad J$

Heap Property: Children come later than their parent (alphabetically)



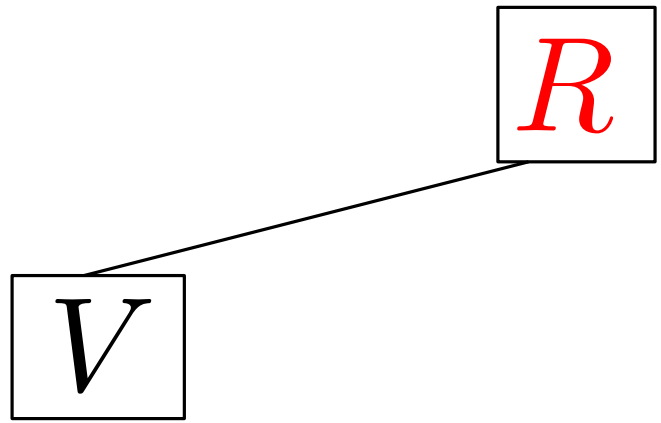
$C \quad D \quad E \quad H \quad J \quad O$

Heap Property: Children come later than their parent (alphabetically)



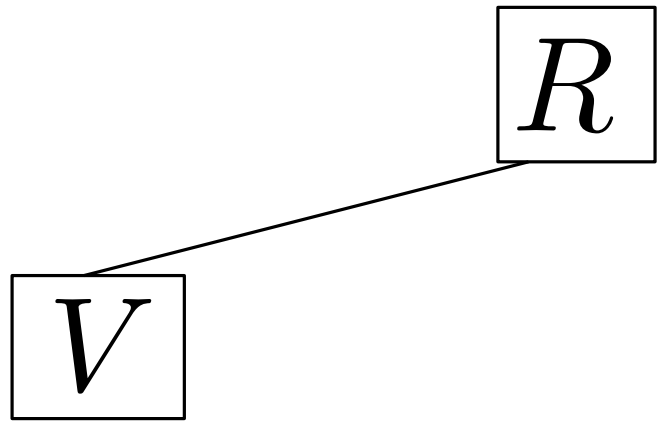
$C \quad D \quad E \quad H \quad J \quad O$

Heap Property: Children come later than their parent (alphabetically)



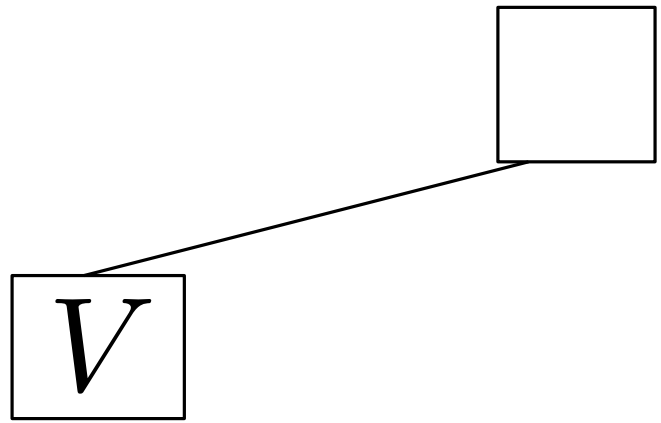
$C \quad D \quad E \quad H \quad J \quad O$

Heap Property: Children come later than their parent (alphabetically)



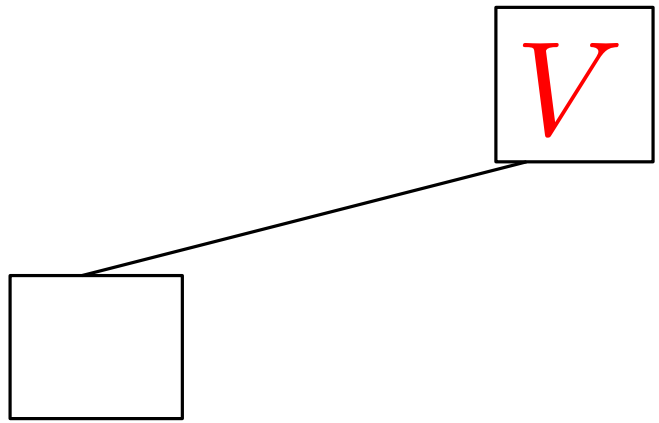
$C \quad D \quad E \quad H \quad J \quad O$

Heap Property: Children come later than their parent (alphabetically)



$C \quad D \quad E \quad H \quad J \quad O \quad R$

Heap Property: Children come later than their parent (alphabetically)



$C \quad D \quad E \quad H \quad J \quad O \quad R$

Heap Property: Children come later than their parent (alphabetically)

V

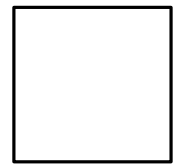
$C \quad D \quad E \quad H \quad J \quad O \quad R$

Heap Property: Children come later than their parent (alphabetically)

$$\boxed{V}$$

$$C \quad D \quad E \quad H \quad J \quad O \quad R$$

Heap Property: Children come later than their parent (alphabetically)

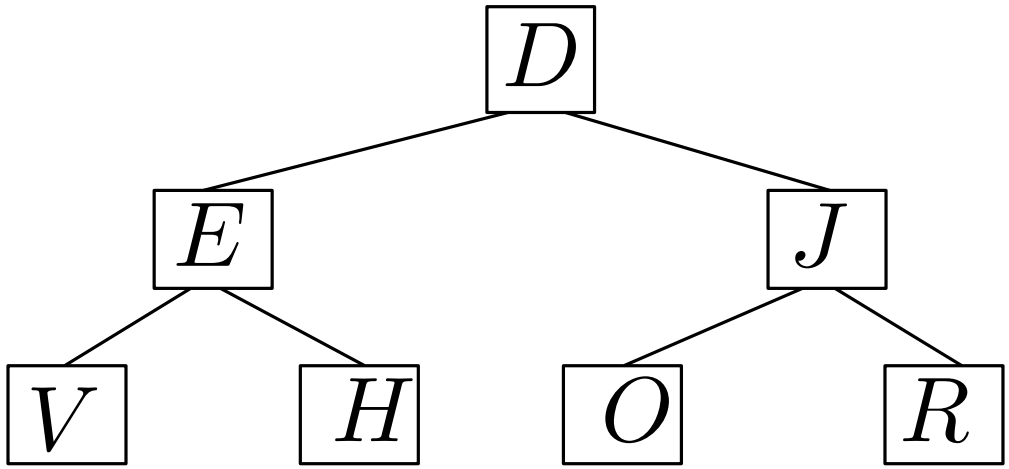


$$C \quad D \quad E \quad H \quad J \quad O \quad R \quad V$$

Heap Property: Children come later than their parent (alphabetically)

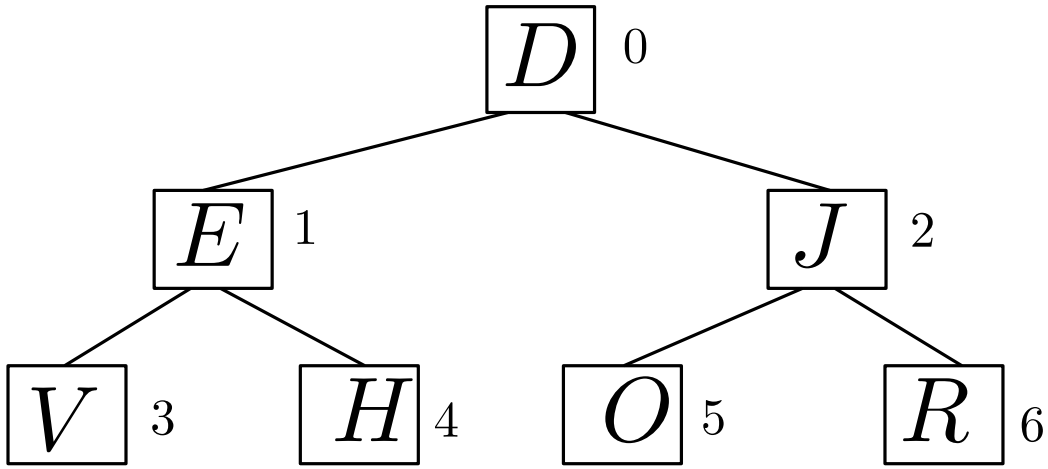
$$C \quad D \quad E \quad H \quad J \quad O \quad R \quad V$$

HeapSort: towards Java Code



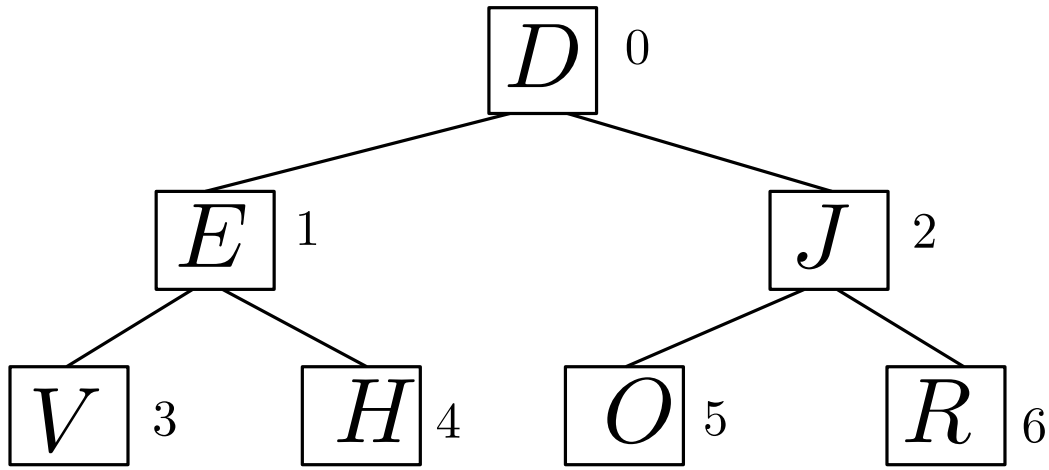
C

HeapSort: towards Java Code

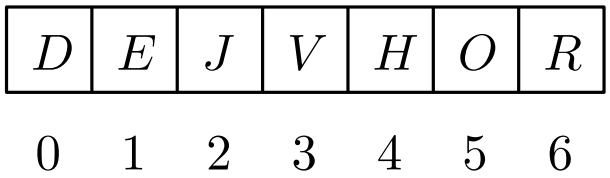


C

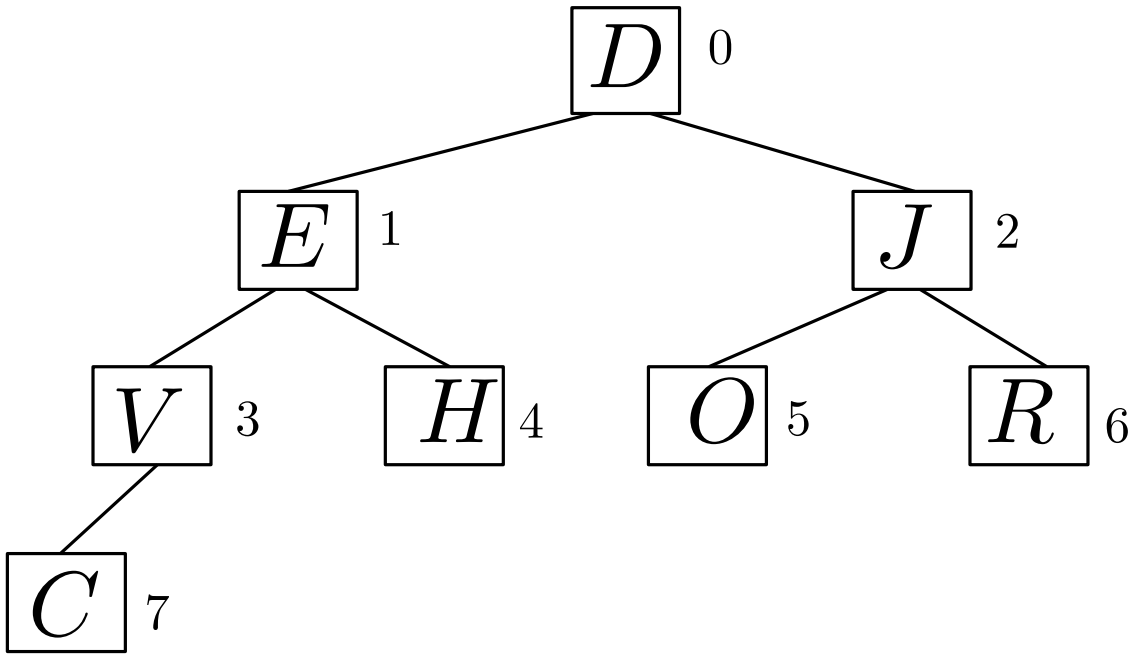
HeapSort: towards Java Code



C

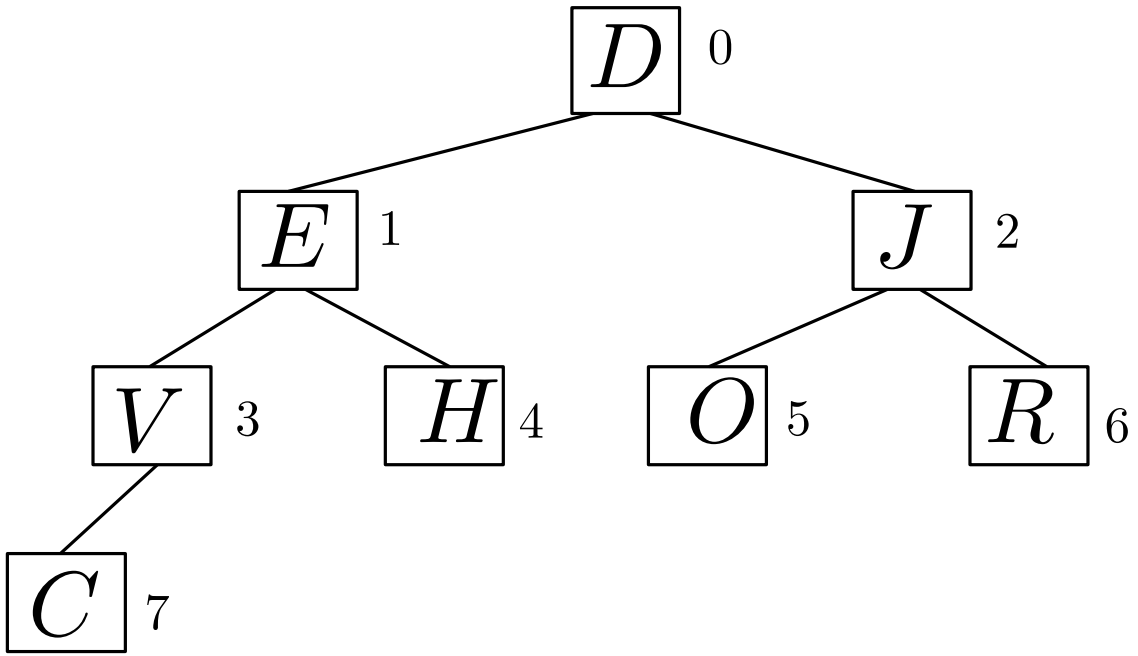


HeapSort: towards Java Code



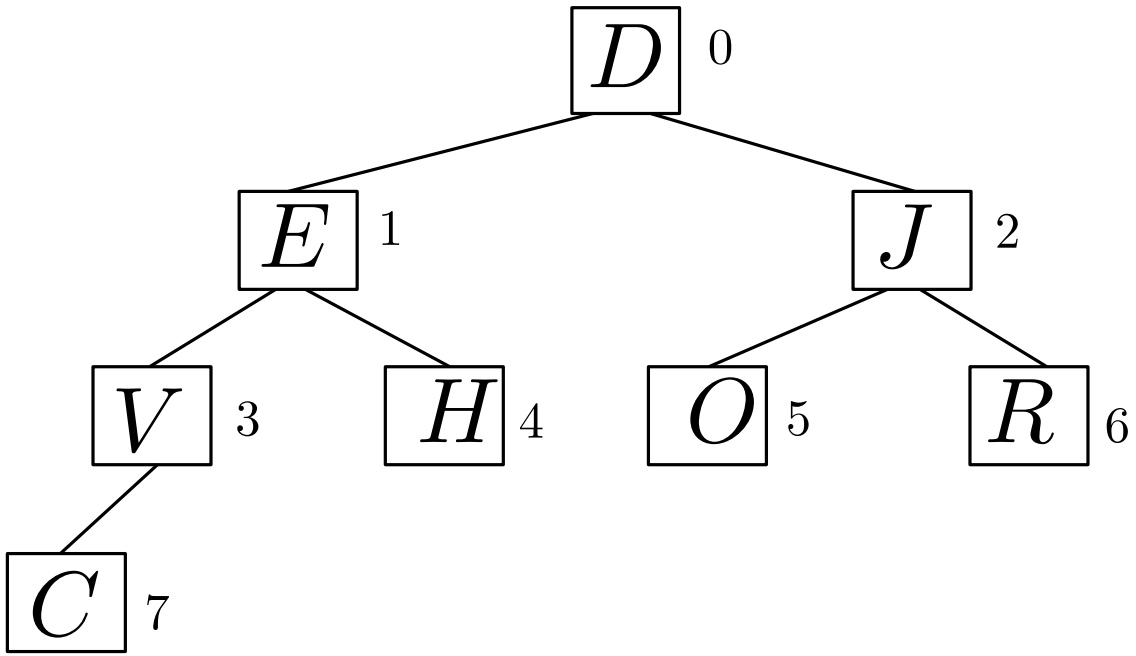
D	E	J	V	H	O	R
0	1	2	3	4	5	6

HeapSort: towards Java Code



D	E	J	V	H	O	R	C
0	1	2	3	4	5	6	7

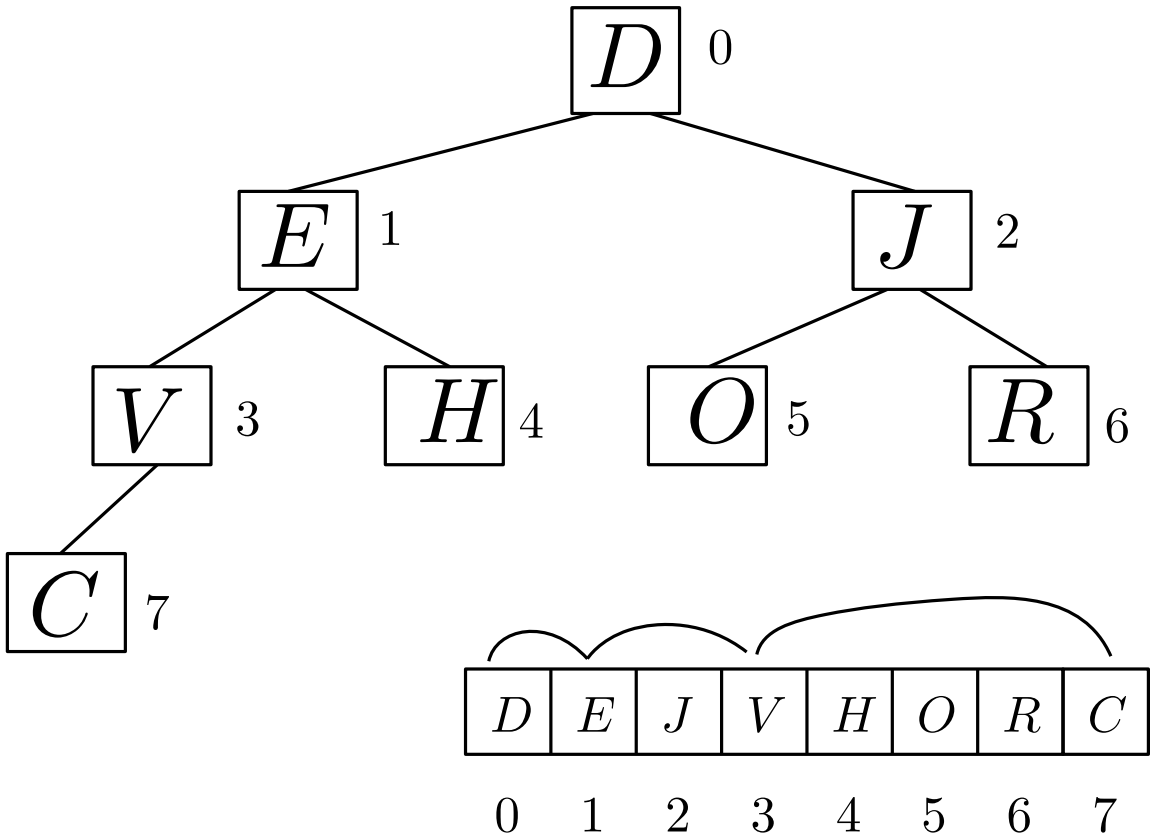
HeapSort: towards Java Code



D	E	J	V	H	O	R	C
0	1	2	3	4	5	6	7

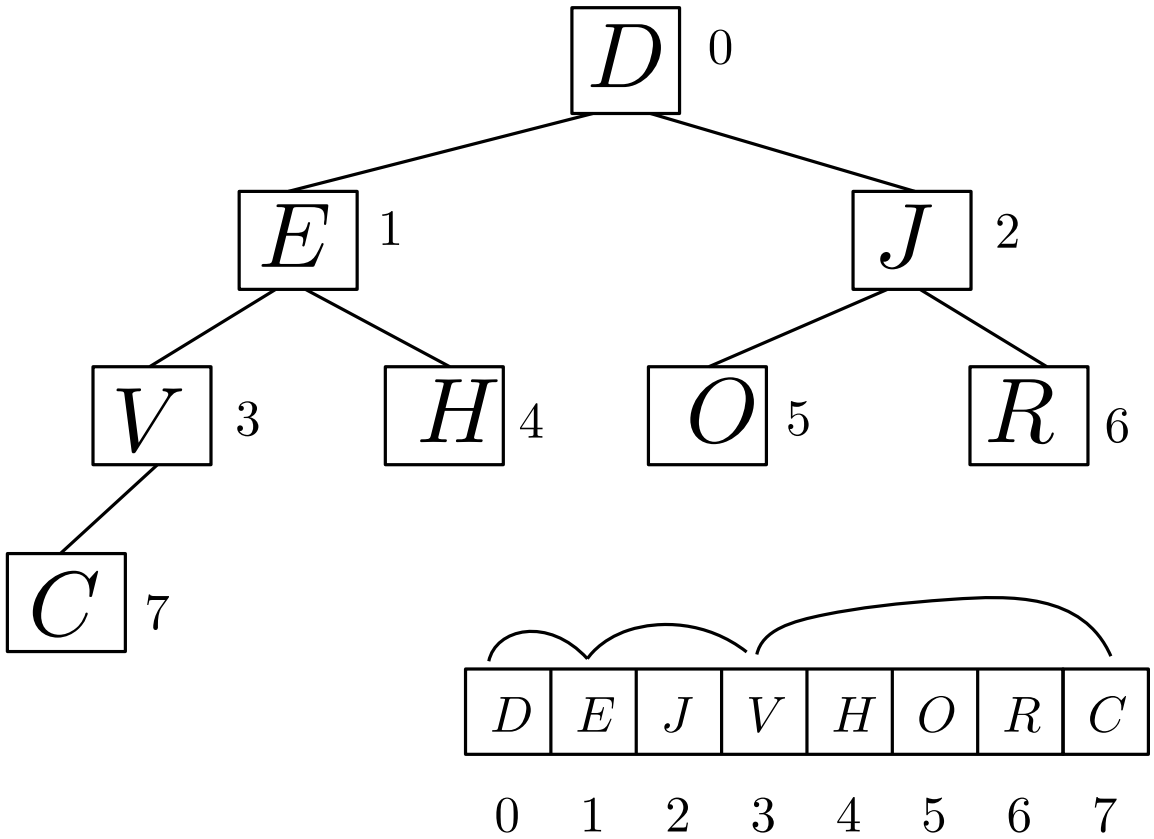
We find the "parent" of cell i by the formula $\left\lfloor \frac{i-1}{2} \right\rfloor$

HeapSort: towards Java Code



We find the "parent" of cell i by the formula $\left\lfloor \frac{i-1}{2} \right\rfloor$

HeapSort: towards Java Code



We find the "parent" of cell i by the formula $\left\lfloor \frac{i-1}{2} \right\rfloor$

Make i hop along this path until it is at its correct position

An arrary A has the *heap property* if $A[i] > A[\mathrm{parent}(i)]$ for all indices $i \geq 1$. Here, $\mathrm{parent}(i) := \left\lfloor \frac{i-1}{2} \right\rfloor$.

An arrary A has the *heap property* if $A[i] > A[\mathrm{parent}(i)]$ for all indices $i \geq 1$. Here, $\mathrm{parent}(i) := \left\lfloor \frac{i-1}{2} \right\rfloor$.

Suppose `array` is an array of length n, and `array[0..i-1]` already has the heap property.
The following code moves the element `array[i]` to its correct position.

```
int node = i;
while (i > 0 && array[i] < array[(i-1)/2] {
    int temp = array[i];
    array[i] = array[(i-1)/2];
    array[(i-1)/2] = temp;
}
```

After this code has been executed, the heap property holds for `array[0..i]`.

An arrary A has the *heap property* if $A[i] > A[\text{parent}(i)]$ for all indices $i \geq 1$. Here, $\text{parent}(i) := \left\lfloor \frac{i-1}{2} \right\rfloor$.

```
// array is an unsorted array
for (int i = 1; i < n; i++) {
    int node = i;
    while (i > 0 && array[i] < array[(i-1)/2] {
        int temp = array[i];
        array[i] = array[(i-1)/2];
        array[(i-1)/2] = temp;
    }
}
```

After this code has been executed, the whole array has the heap property.

An arrary A has the *heap property* if $A[i] > A[\text{parent}(i)]$ for all indices $i \geq 1$. Here, $\text{parent}(i) := \left\lfloor \frac{i-1}{2} \right\rfloor$.

compareTo when using String

```
// array is an unsorted array
for (int i = 1; i < n; i++) {
    int node = i;
    while (i > 0 && array[i] < array[(i-1)/2] {
        int temp = array[i];
        array[i] = array[(i-1)/2];
        array[(i-1)/2] = temp;
    }
}
```

After this code has been executed, the whole array has the heap property.