Algoritmi e Strutture Dati

Alberi di Ricerca

Alberi Bilanciati I (Alberi AVL)

Alberi bilanciati di ricerca

- **Gli *alberi binari di ricerca* sono semplici da gestire (inserimenti e cancellazioni facili da implementare) ma hanno prestazioni poco prevedibili e potenzialmente basse**
- **Gli *alberi perfettamente bilanciati* hanno prestazioni ottimali (*log N* garantito) ma inserimenti e cancellazioni complesse (ri-bilanciamenti)**
- **Alberi AVL (*Adelson-Velskii e Landis*): *classe di alberi bilanciati* (non ottimale come la precedente). Hanno buone prestazioni e gestione relativamente semplice.**

Alberi AVL: definizione

Definizione: **Un albero binario di ricerca è un *Albero AVL* se per ogni nodo *x*:**

- **l'*altezza* del *sottoalbero sinistro di x* e quella del *sottoalbero destro di x* differiscono al più di uno, e**

NOTA: In ogni albero AVL entrambi i sottoalberi **sinistro e destro di un qualsiasi nodo sono *alberi AVL*.**

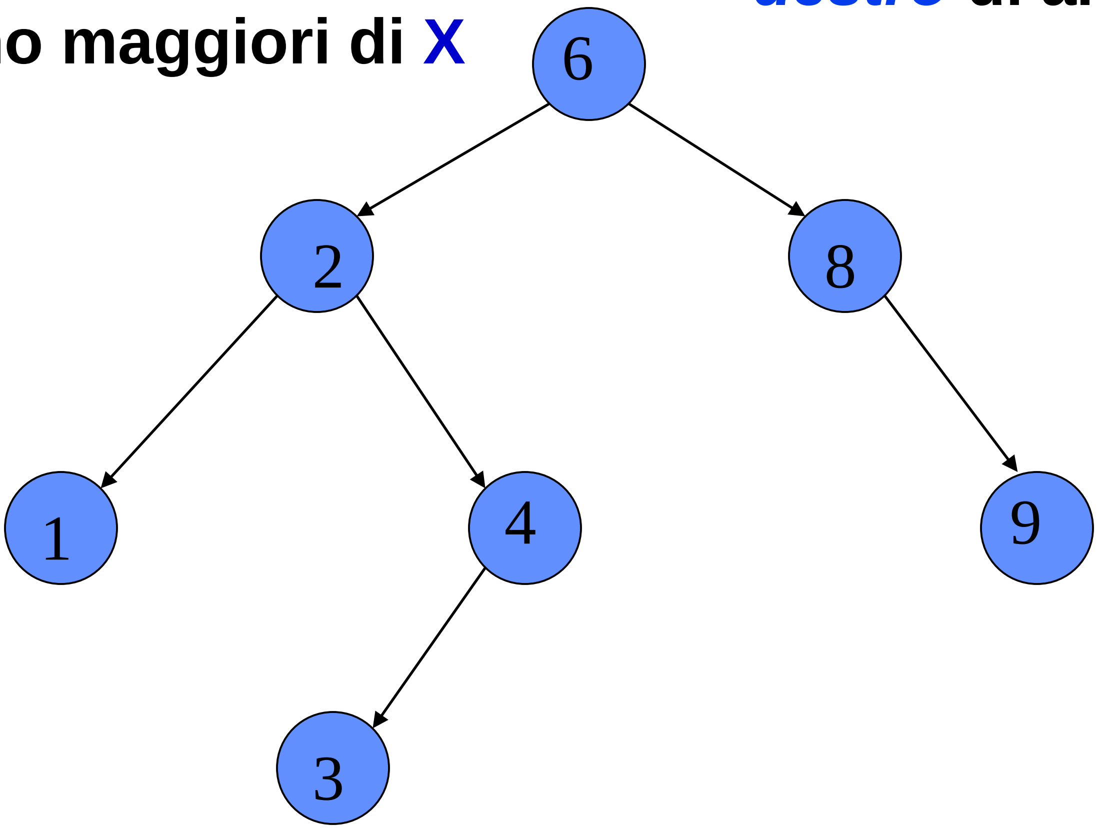
Alberi AVL: alberi binari bilanciati

Proprietà* degli *ABR

Per ogni nodo X, i nodi del sottoalbero sinistro sono minori del nodo X, e I nodi del sottoalbero destro sono maggiori di X

Properietà AVL

Ad ogni nodo *X*, l'*altezza* del *sottoalbero sinistro differisce* da quella del *destro* di al massimo 1.

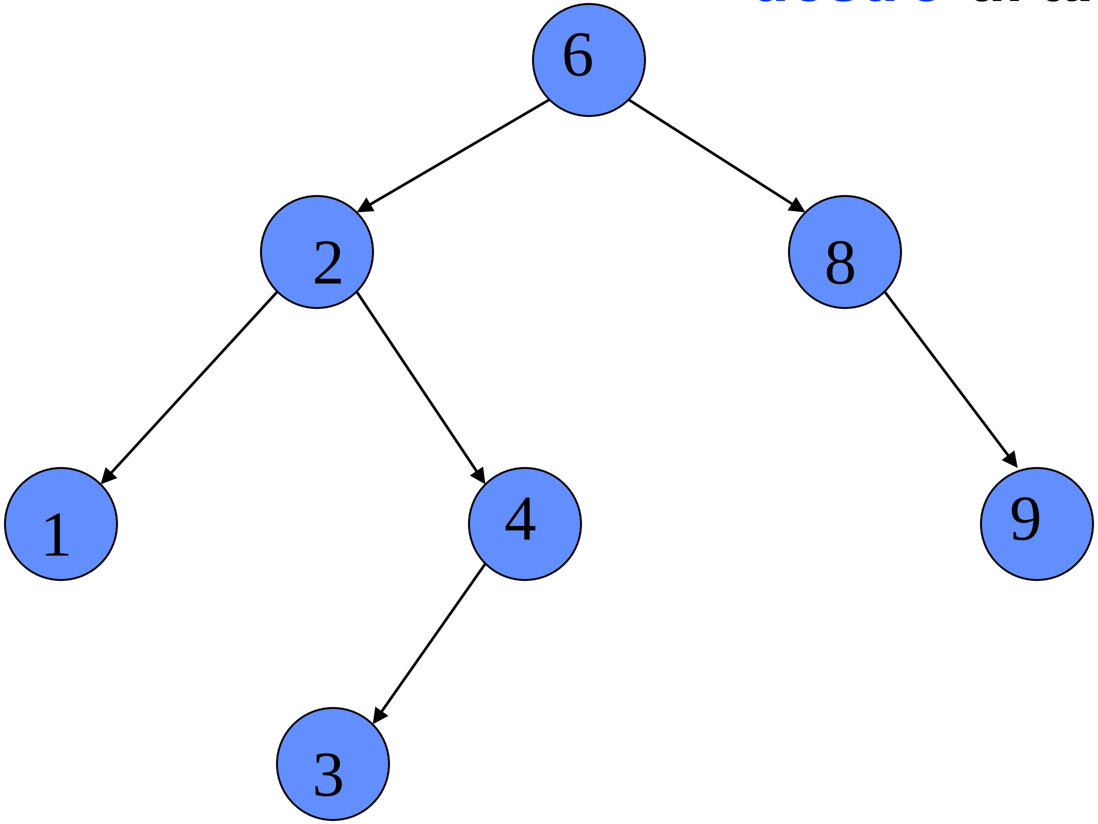


Alberi AVL: alberi binari bilanciati

Altezza(*X*) =
max(Altezza(sinistro(*X*)),
Altezza(destro(*X*))) + 1

Properietà AVL

Ad ogni nodo *X*, l'*altezza* del *sottoalbero sinistro differisce* da quella del *destro* di al massimo 1.

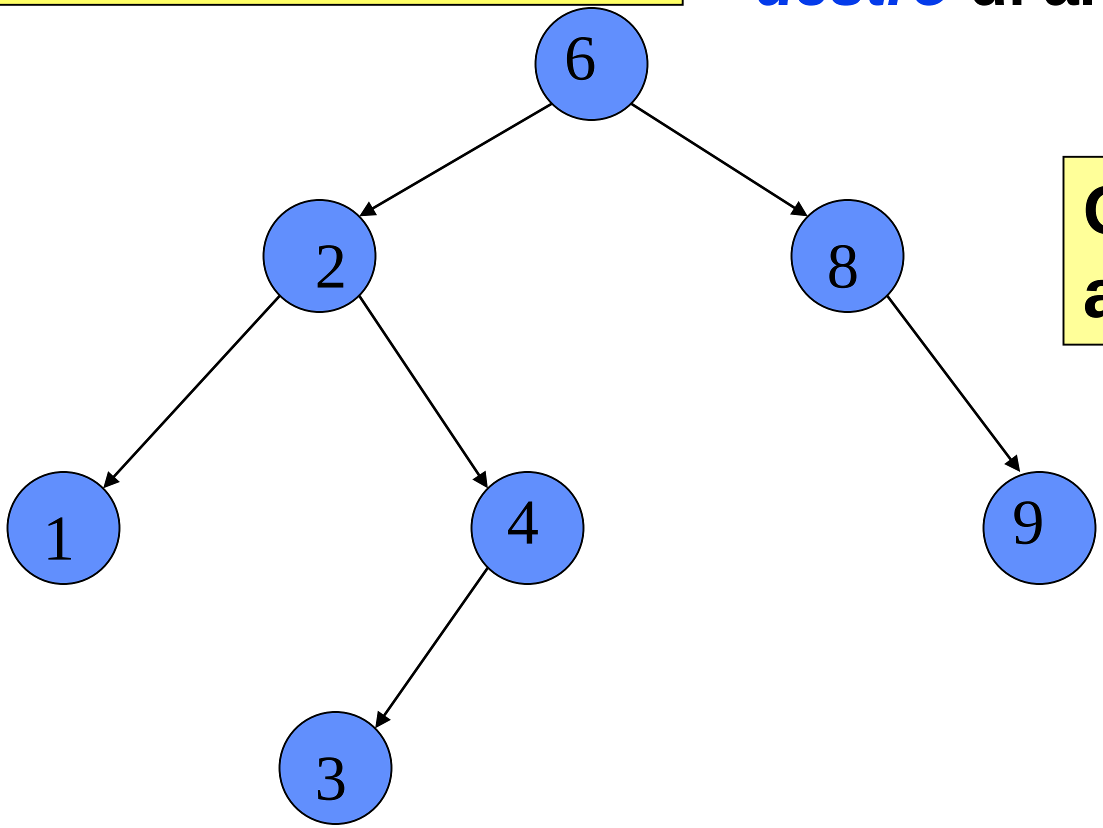


Alberi AVL: alberi binari bilanciati

Altezza(X) =
max(Altezza(sinistro(X)),
Altezza(destro(X))) + 1
Altezza($\varnothing$) = -1

Properietà AVL

Ad ogni nodo *X*, l'*altezza* del *sottoalbero sinistro* *differisce* da quella del *destro* di al massimo 1.



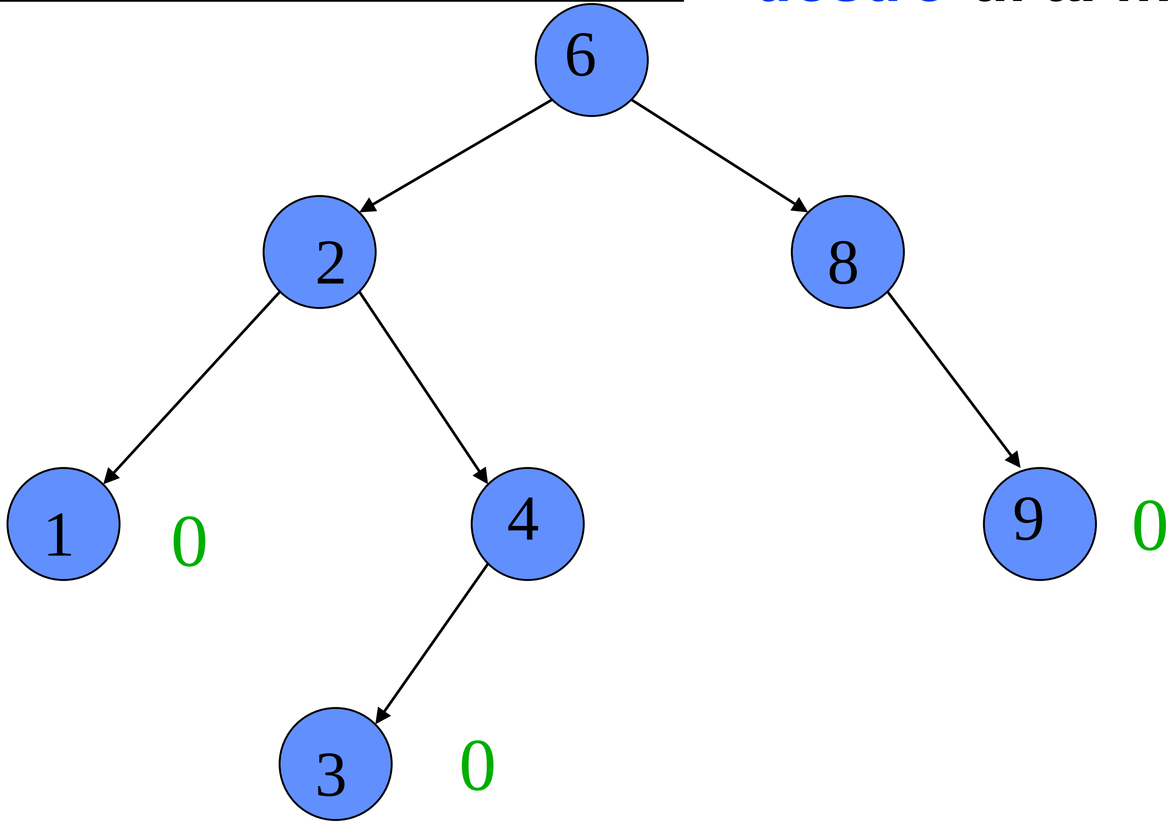
Questo è un albero AVL?

Alberi AVL: alberi binari bilanciati

Altezza(*X*) =
max(Altezza(sinistro(*X*)),
Altezza(destro(*X*))) + 1
Altezza(∅) = -1

Properietà AVL

Ad ogni nodo *X*, l'*altezza* del *sottoalbero sinistro differisce* da quella del *destro* di al massimo 1.

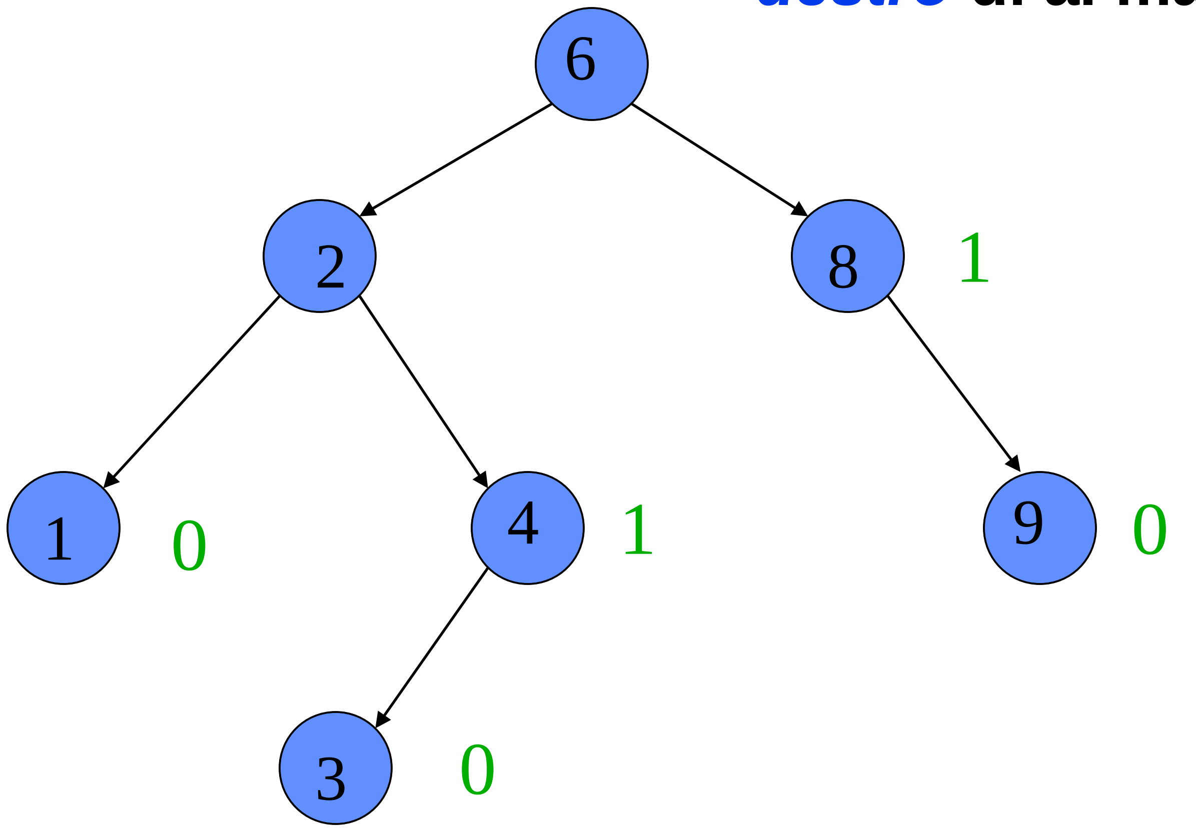


Alberi AVL: alberi binari bilanciati

Altezza(*X*) =
max(Altezza(sinistro(*X*)),
Altezza(destro(*X*))) + 1

Properietà AVL

Ad ogni nodo *X*, l'*altezza* del *sottoalbero sinistro differisce* da quella del *destro* di al massimo 1.

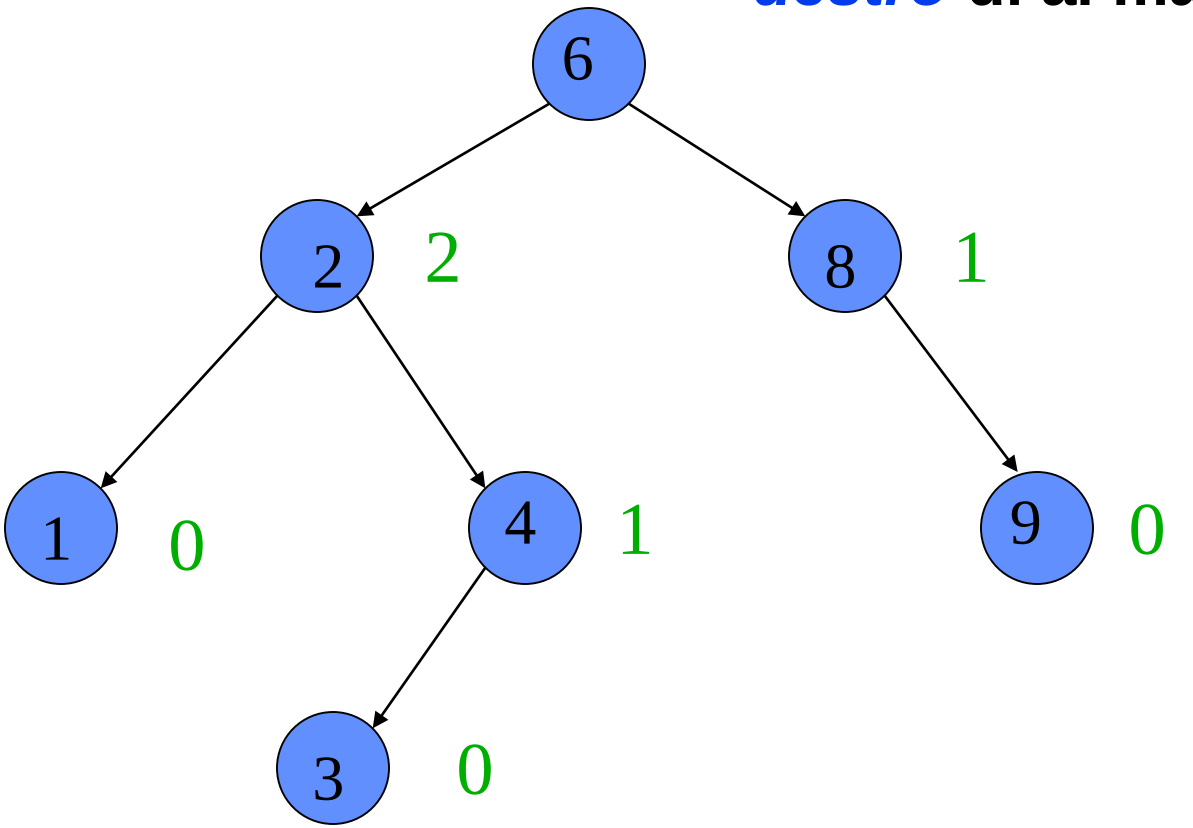


Alberi AVL: alberi binari bilanciati

Altezza(*X*) =
max(Altezza(sinistro(*X*)),
Altezza(destro(*X*))) + 1

Properietà AVL

Ad ogni nodo *X*, l'*altezza* del *sottoalbero sinistro differisce* da quella del *destro* di al massimo 1.

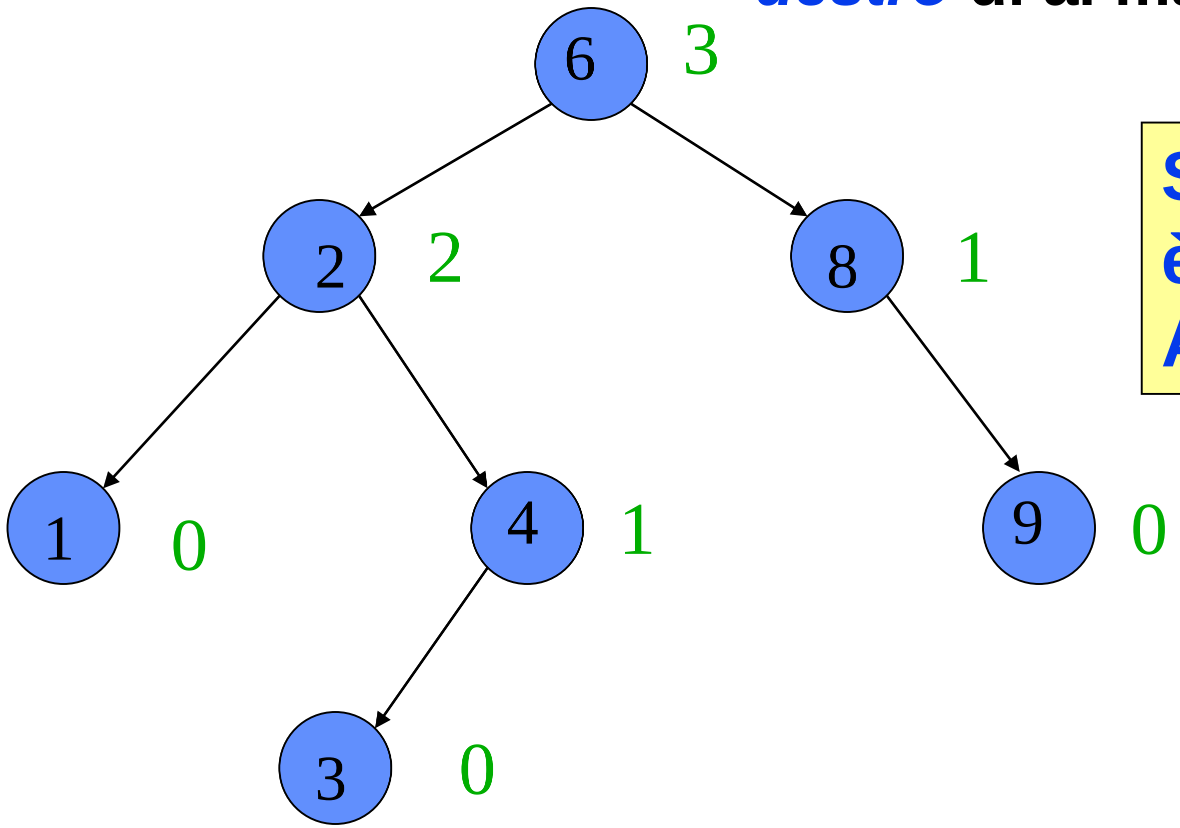


Alberi AVL: alberi binari bilanciati

Altezza(*X*) =
max(Altezza(sinistro(*X*)),
Altezza(destro(*X*))) + 1

Properietà AVL

Ad ogni nodo *X*, l'*altezza* del *sottoalbero sinistro differisce* da quella del *destro* di al massimo 1.



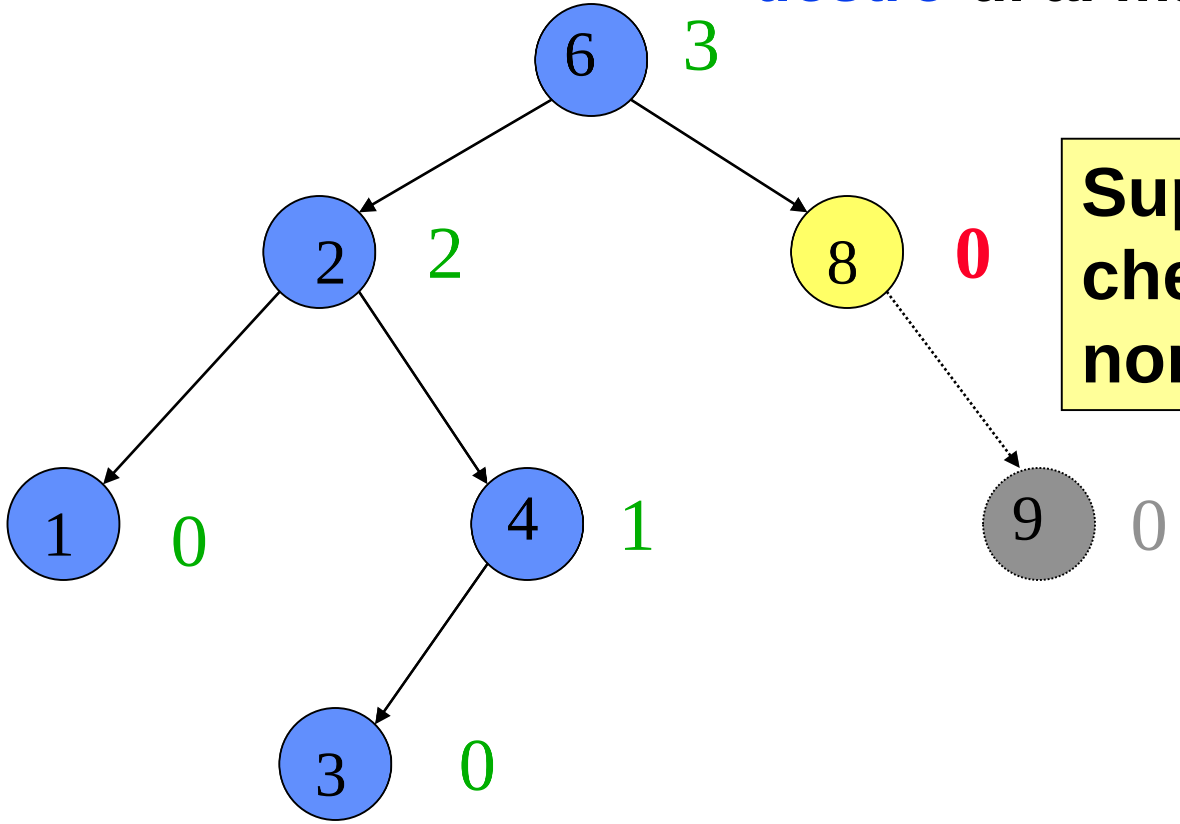
Sì! Questo è un albero AVL.

Alberi AVL: alberi binari bilanciati

Altezza(*X*) =
max(Altezza(sinistro(*X*)),
Altezza(destro(*X*))) + 1

Properietà AVL

Ad ogni nodo *X*, l'*altezza* del *sottoalbero sinistro differisce* da quella del *destro* di al massimo 1.



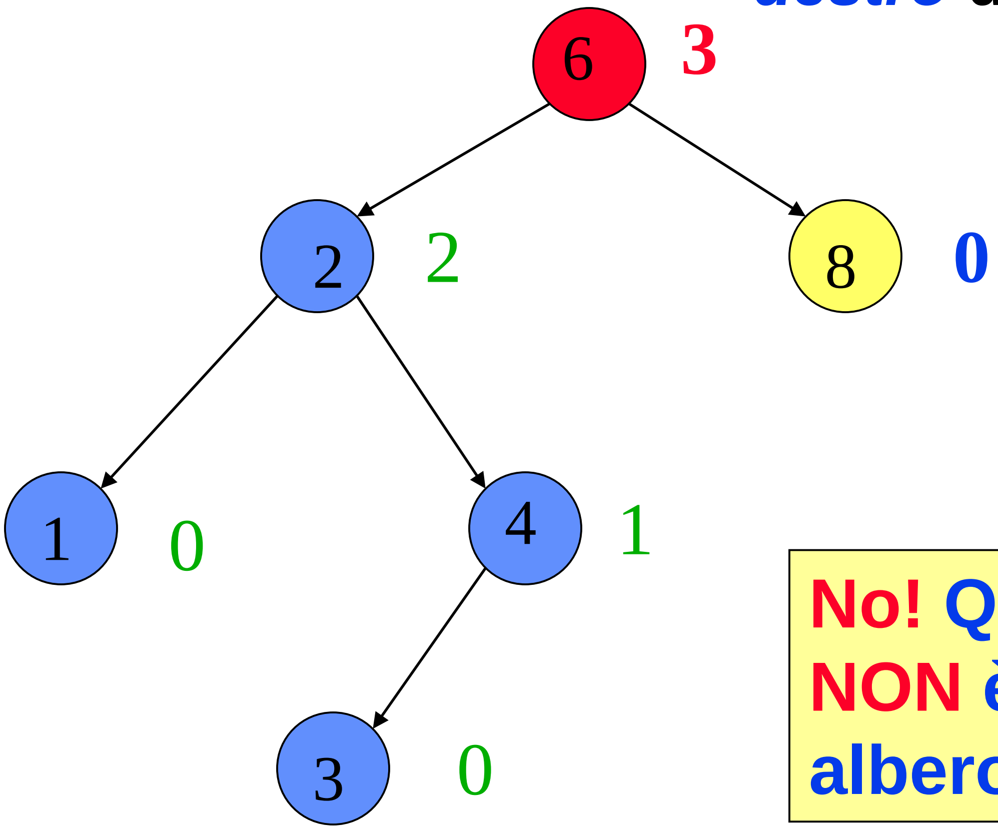
Supponiamo che il nodo 9 non ci sia.

Alberi AVL: alberi binari bilanciati

Altezza(*X*) =
max(Altezza(sinistro(*X*)),
Altezza(destro(*X*))) + 1

Properietà AVL

Ad ogni nodo *X*, l'*altezza* del *sottoalbero sinistro differisce* da quella del *destro* di al massimo 1.



La *Proprietà* AVL *non* è *soddisfatta* dal *nodo* 6.

No! Questo NON è un albero AVL.

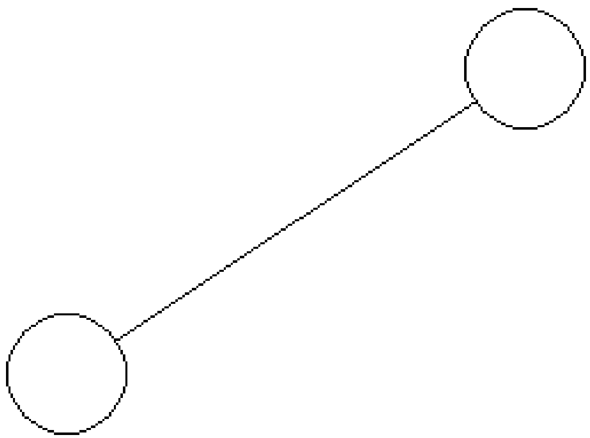
Alberi AVL minimi

Definizione: **fissato *h*, l'*albero AVL* minimo di altezza *h* è l'albero AVL di altezza *h* col minor numero di nodi possibile.**

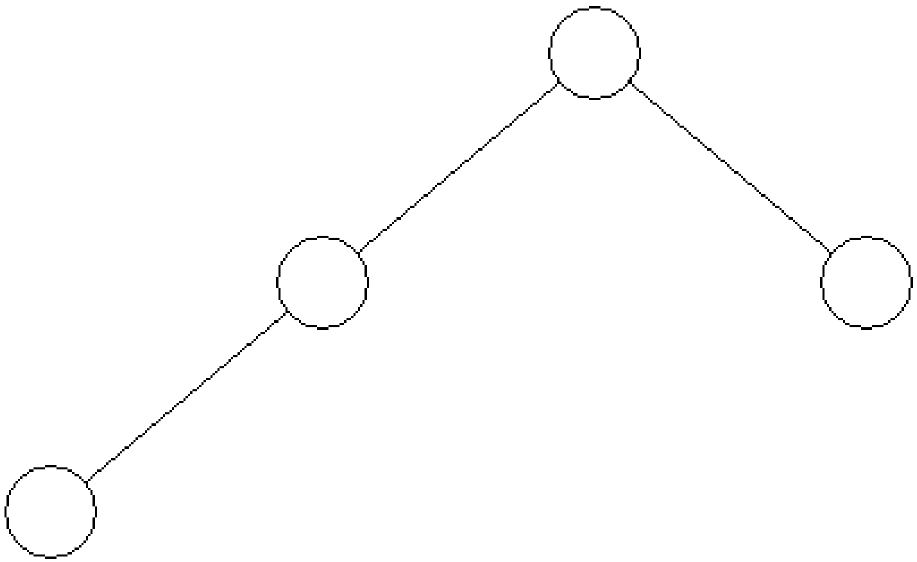
Un Albero AVL minimo di altezza h e con numero di nodi pari n è l'Albero AVL di altezza massima tra tutti gli AVL con n nodi.

Esempi di alberi AVL minimi di diverse altezze (cioè con il numero minimo di nodi)

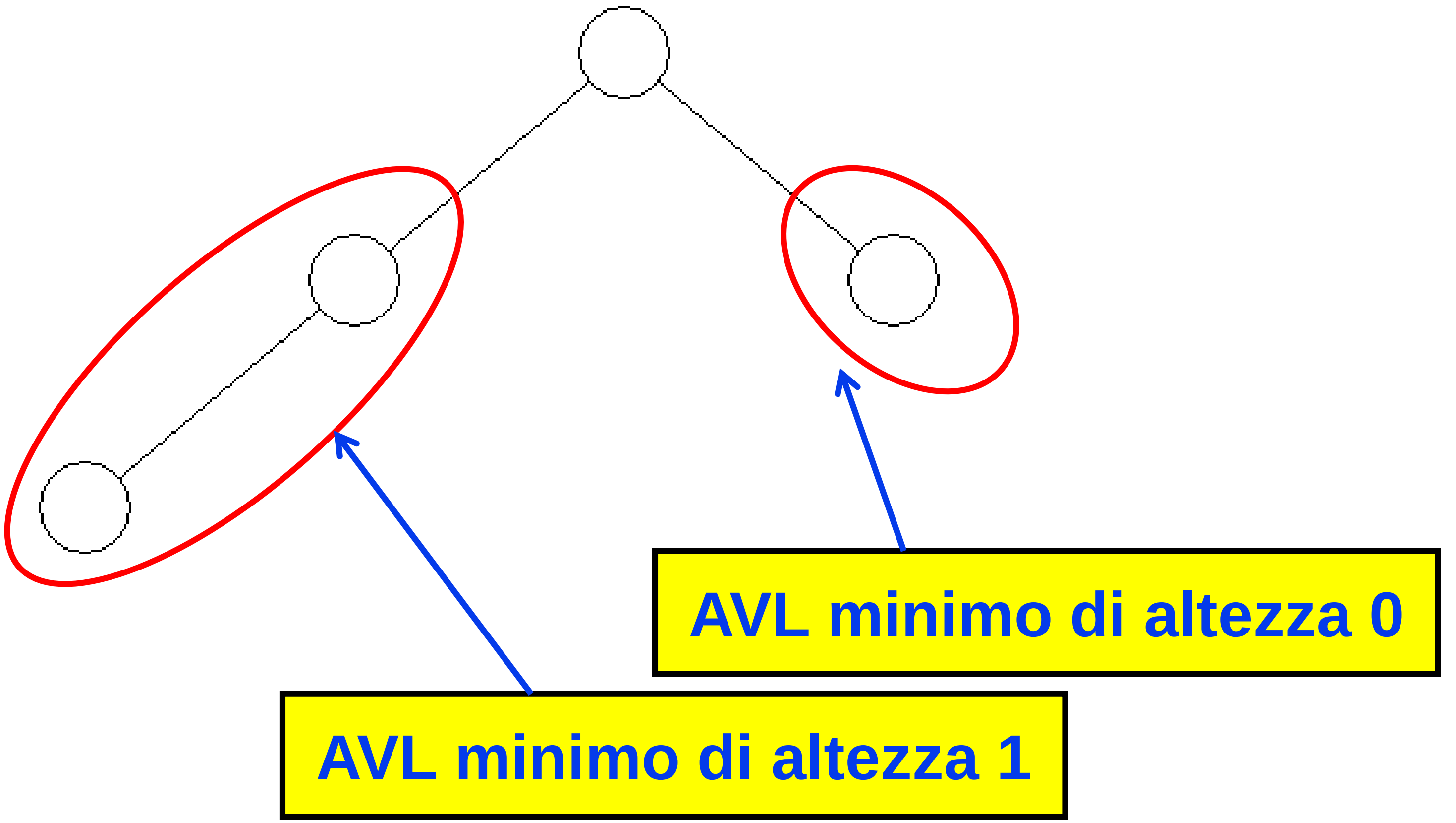
Albero AVL minimo di altezza 1



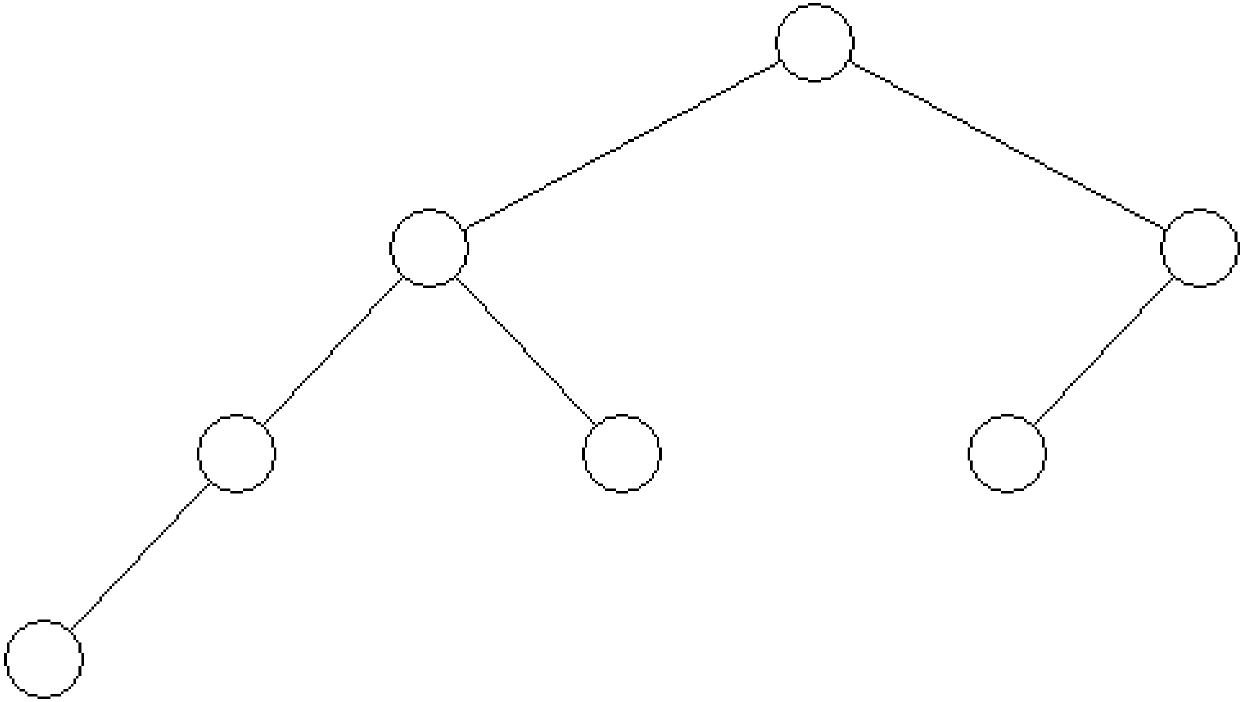
Albero AVL minimo di altezza 2



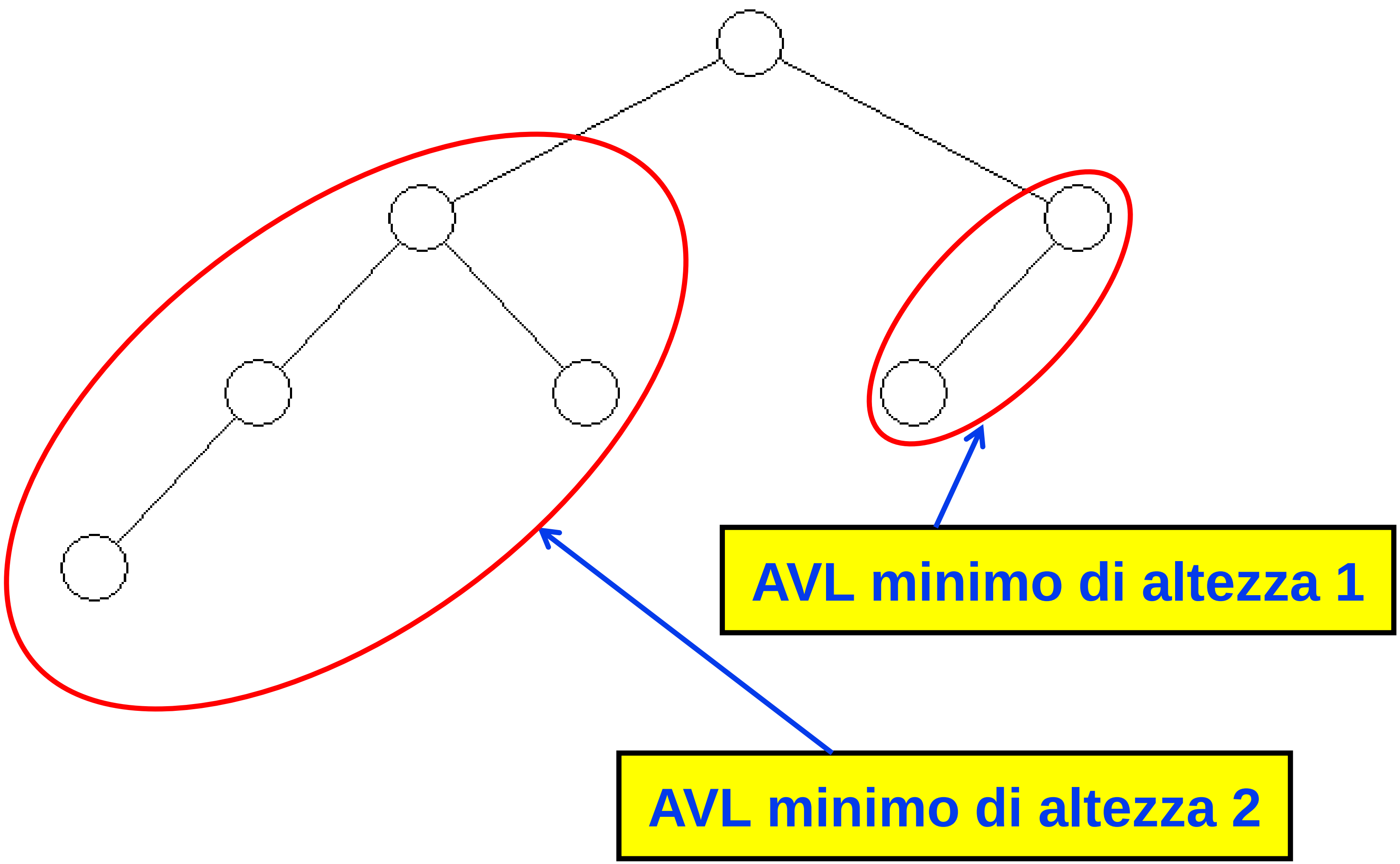
Albero AVL minimo di altezza 2



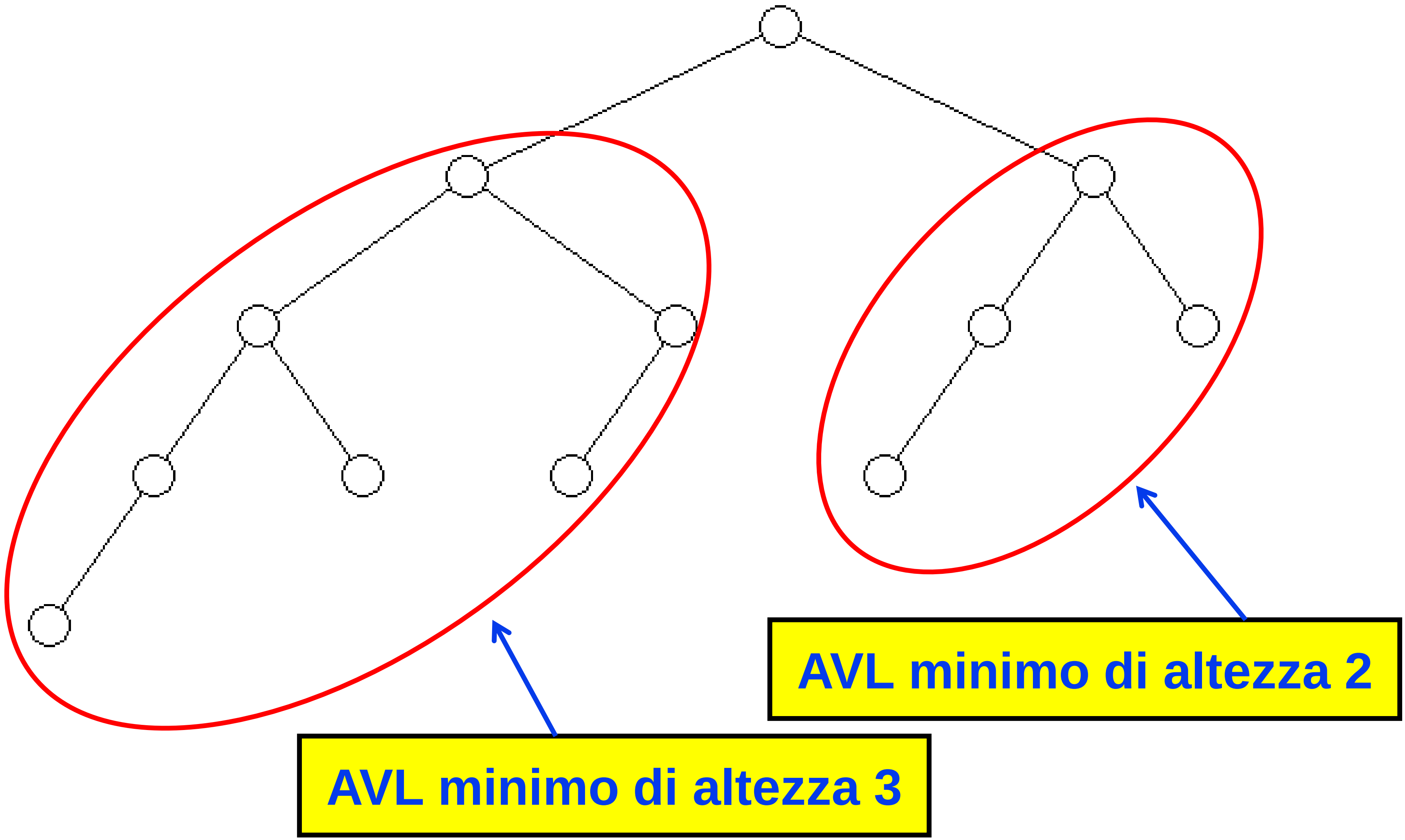
Albero AVL minimo di altezza 3



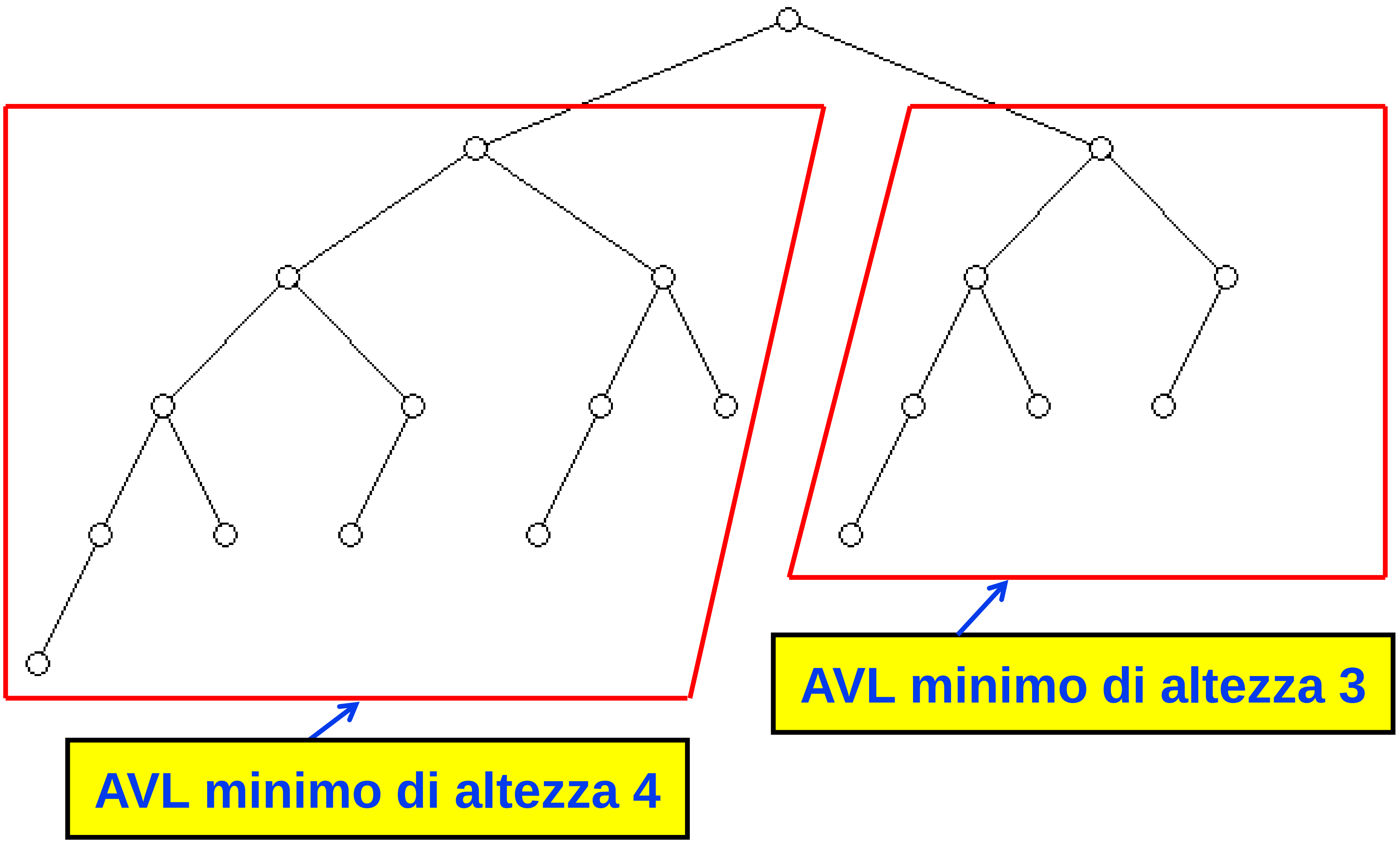
Albero AVL minimo di altezza 3



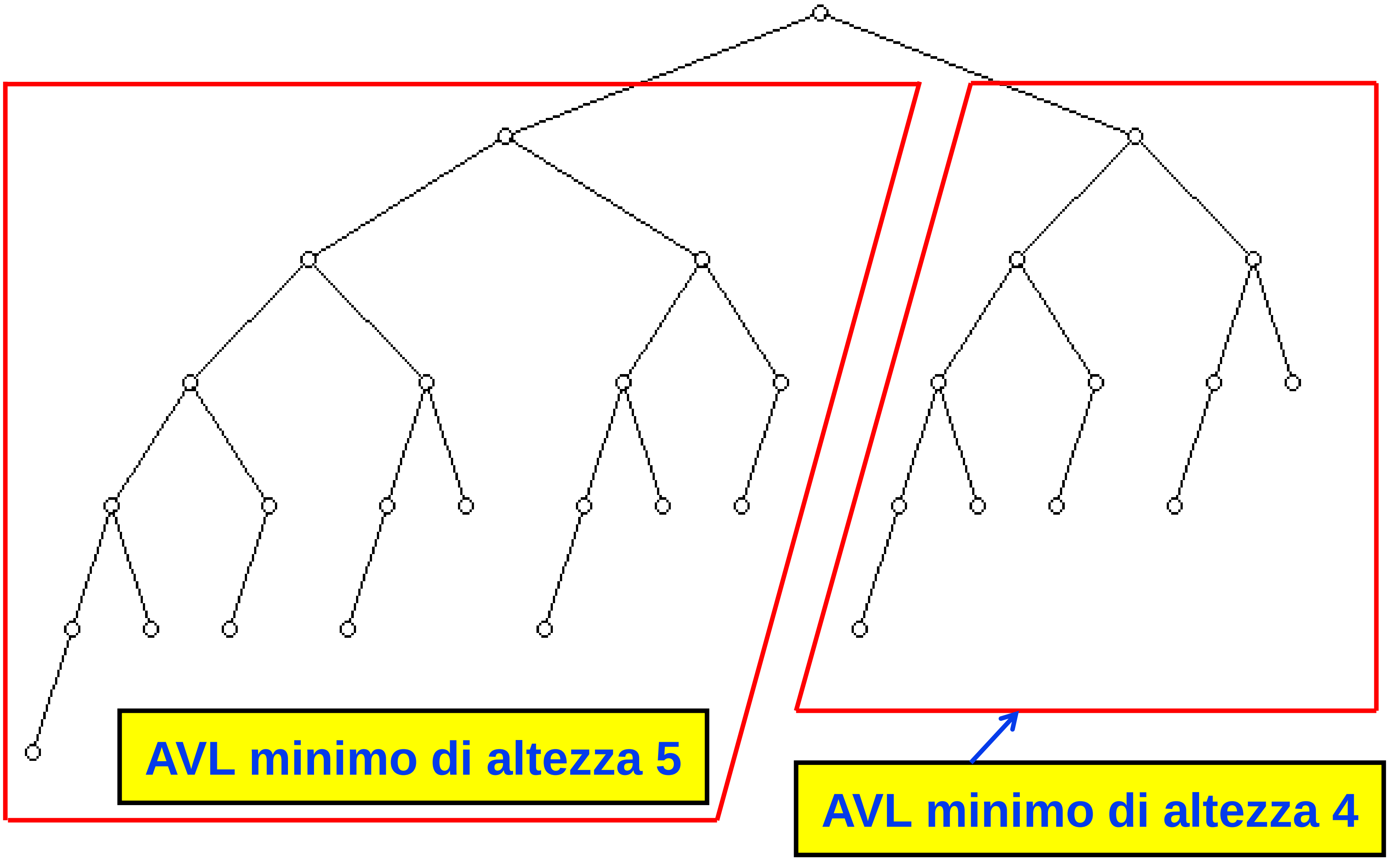
Albero AVL minimo di altezza 4



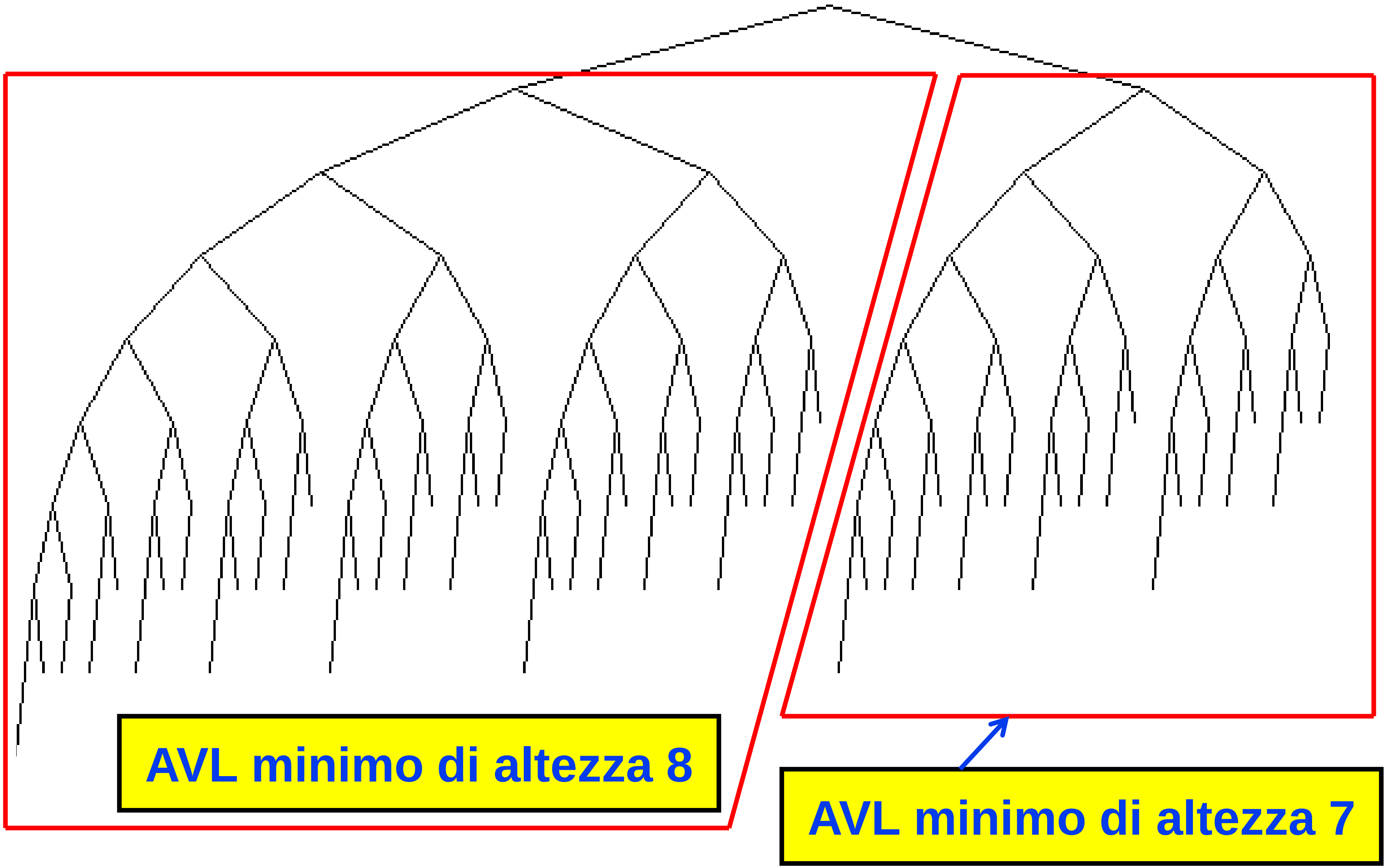
Albero AVL minimo di altezza 5



Albero AVL minimo di altezza 6



Albero AVL minimo di altezza 9



Numero minimo di nodi di un albero AVL

- **Sia *N* è il *numero minimo di nodi* di un albero AVL di altezza *H***

Altezza *H*	0	1	2	3	4	5	6	7	8	9	10
Nodi *N(H)*	1	2	4	**7**	**12**	**20**	33	54	88	143	232

$$N(H) = N(H-1) + N(H-2) + 1$$

Numero minimo di nodi di un albero AVL

Altezza *H*	0	1	2	3	4	**5**	6	**7**	8	9	10
Nodi *N(H)*	1	2	4	7	12	**20**	33	**54**	88	143	232

- **Sia *N* è il *numero minimo di nodi* di un albero AVL di altezza *H***

H	0	1	2	3	4	5	6	7	**8**	9	**10**
Fib(H)	0	1	1	2	3	5	8	13	**21**	34	**55**

- **Si dimostra che**

$$\boxed{N(H) = Fib(H+3) - 1}$$

dove *Fib(k)* è il *k*-esimo *numero di Fibonacci*

Numero minimo di nodi di un albero AVL

Altezza *H*	0	1	2	3	4	5	6	7	8	9	10
Nodi *N(H)*	1	2	4	7	12	20	33	54	88	143	232

- **Sia *N* è il *numero minimo di nodi* di un albero AVL di altezza *H***
- **Si dimostra (ad esempio *per induzione*) che**

$$N(H) = Fib(H+3) - 1$$

dove *Fib(k)* è il *k*-esimo *numero di Fibonacci*

- **Per grandi valori di *k*,**

$$Fib(k) \cong \frac{1}{\sqrt{5}} \left[\frac{1+\sqrt{5}}{2} \right]^k$$

Numero minimo di nodi di un albero AVL

Altezza *H*	0	1	2	3	4	5	6	7	8	9	10
Nodi *N(H)*	1	2	4	7	12	20	33	54	88	143	232

- **Si dimostra che $N(H) = F(H+3) - 1$**
- **Il *numero minimo N* di *nodi* di un albero AVL di altezza *H* è allora dato da**

$$N(H) \cong \frac{1}{\sqrt{5}}\left[\frac{1+\sqrt{5}}{2}\right]^{H+3} - 1$$

$$Fib(k) \cong \frac{1}{\sqrt{5}}\left[\frac{1+\sqrt{5}}{2}\right]^{k}$$

Numero minimo di nodi di un albero AVL

Altezza *H*	0	1	2	3	4	5	6	7	8	9	10
Nodi *N(H)*	1	2	4	7	12	20	33	54	88	143	232

- **Risolvendo per *H* si ottiene:**

$$H \cong 1.44 \log(N+2) - .328$$

- **Il *numero minimo N* di *nodi* di un albero AVL di altezza *H* è allora dato da**

$$N(H) \cong \frac{1}{\sqrt{5}}\left[\frac{1+\sqrt{5}}{2}\right]^{H+3} - 1$$

Numero minimo di nodi di un albero AVL

Altezza *H*	0	1	2	3	4	5	6	7	8	9	10
Nodi *N(H)*	1	2	4	7	12	20	33	54	88	143	232

- **Risolvendo per *H* si ottiene:**

$$H \simeq 1.44\log(N+2) - .328$$

- **Ciò significa che l'*altezza* di un *albero AVL* con *N* nodi *NON* è mai più del 44% *maggiore* dell'*atezza ottima* di un *albero perfettamente bilanciato* con *N* nodi.**

Costruzione di un albero AVL

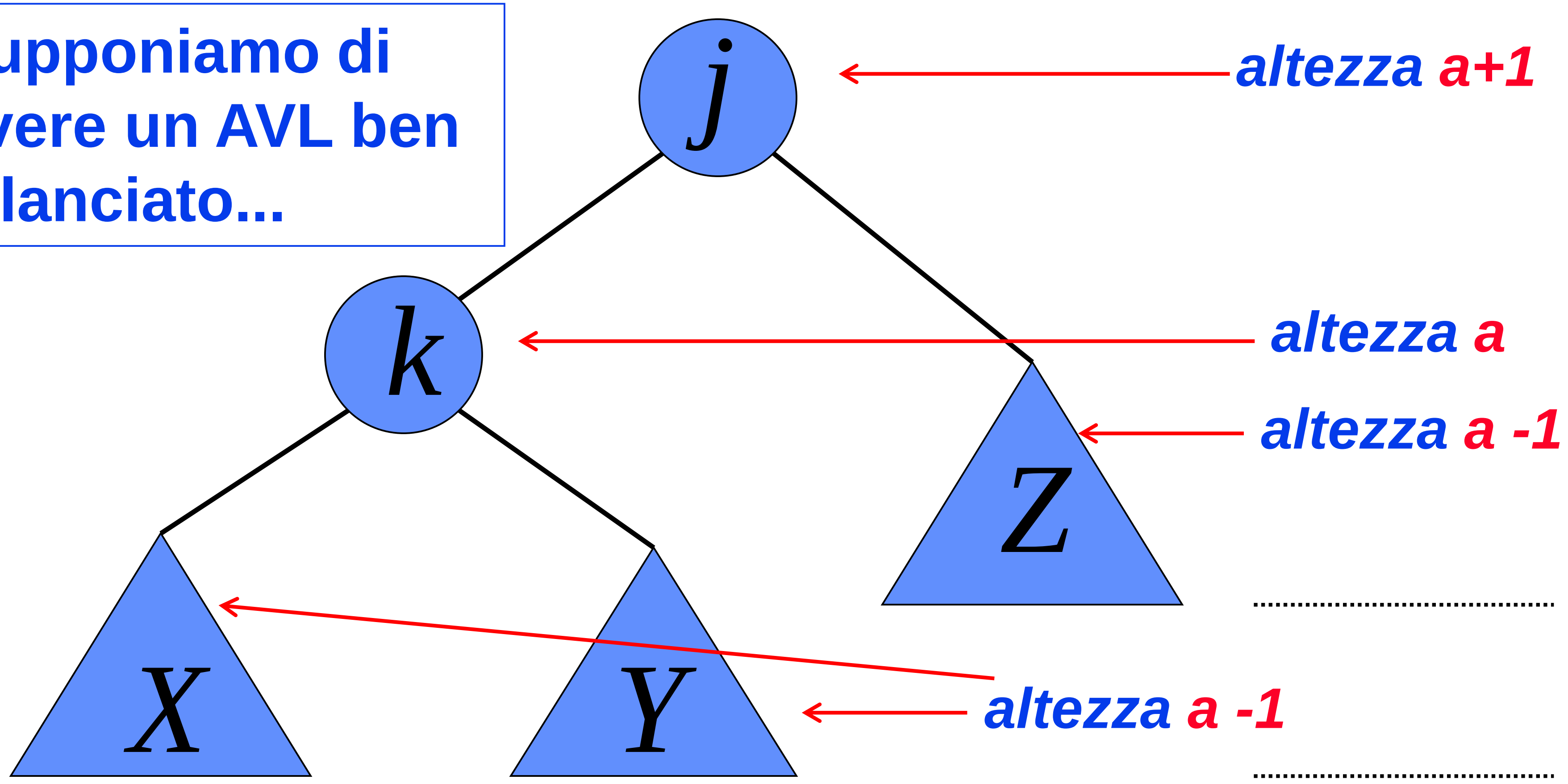
- **Durante la *costruzione* di un *albero AVL*, l'unica volta che è possibile *violare la condizione di bilanciamento* è quando un *nuovo nodo* viene *inserito*.**

- **Ci concentreremo sull'avo più vicino al nuovo nodo inserito, che è divenuto non bilanciato**

- ***Ci sono essenzialmente due casi, chiamati Rotazione singola e Ritazione doppia***

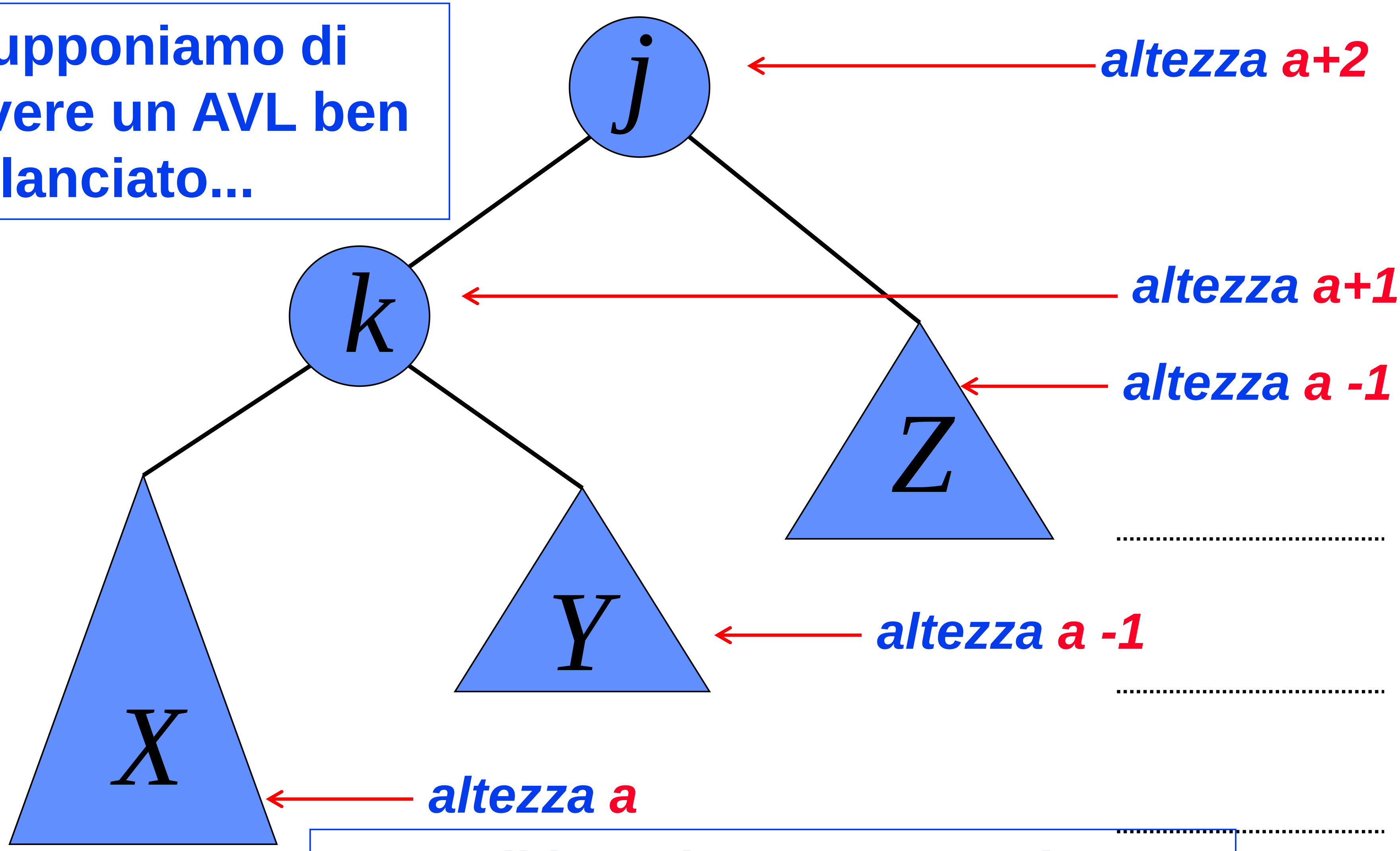
Rotazione in alberi AVL: caso I

Supponiamo di avere un AVL ben bilanciato...



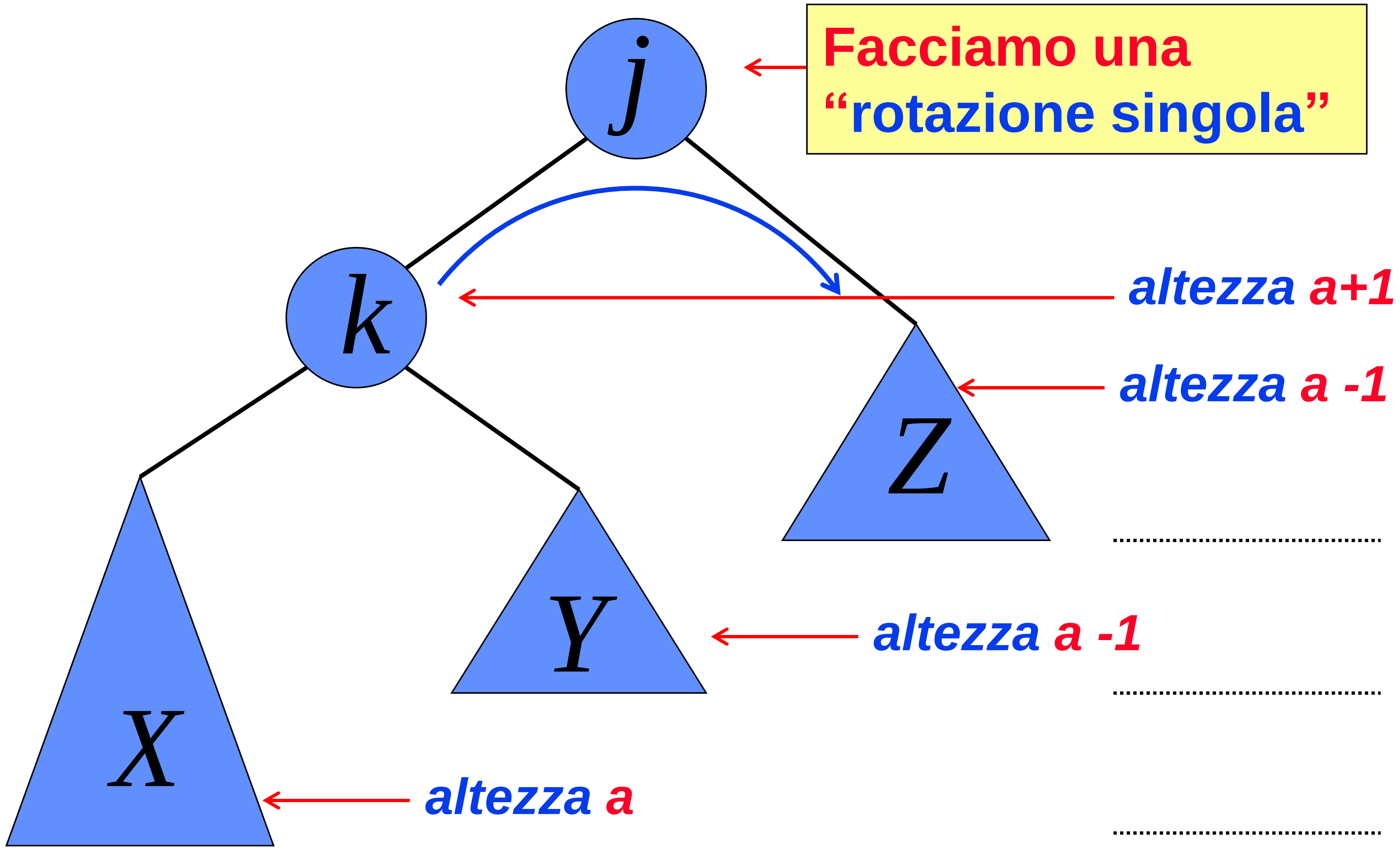
Rotazione in alberi AVL: caso I

Supponiamo di avere un AVL ben bilanciato...



... ma di inserire nel sottoalbero *X*, perdendo così il bilanciamento.

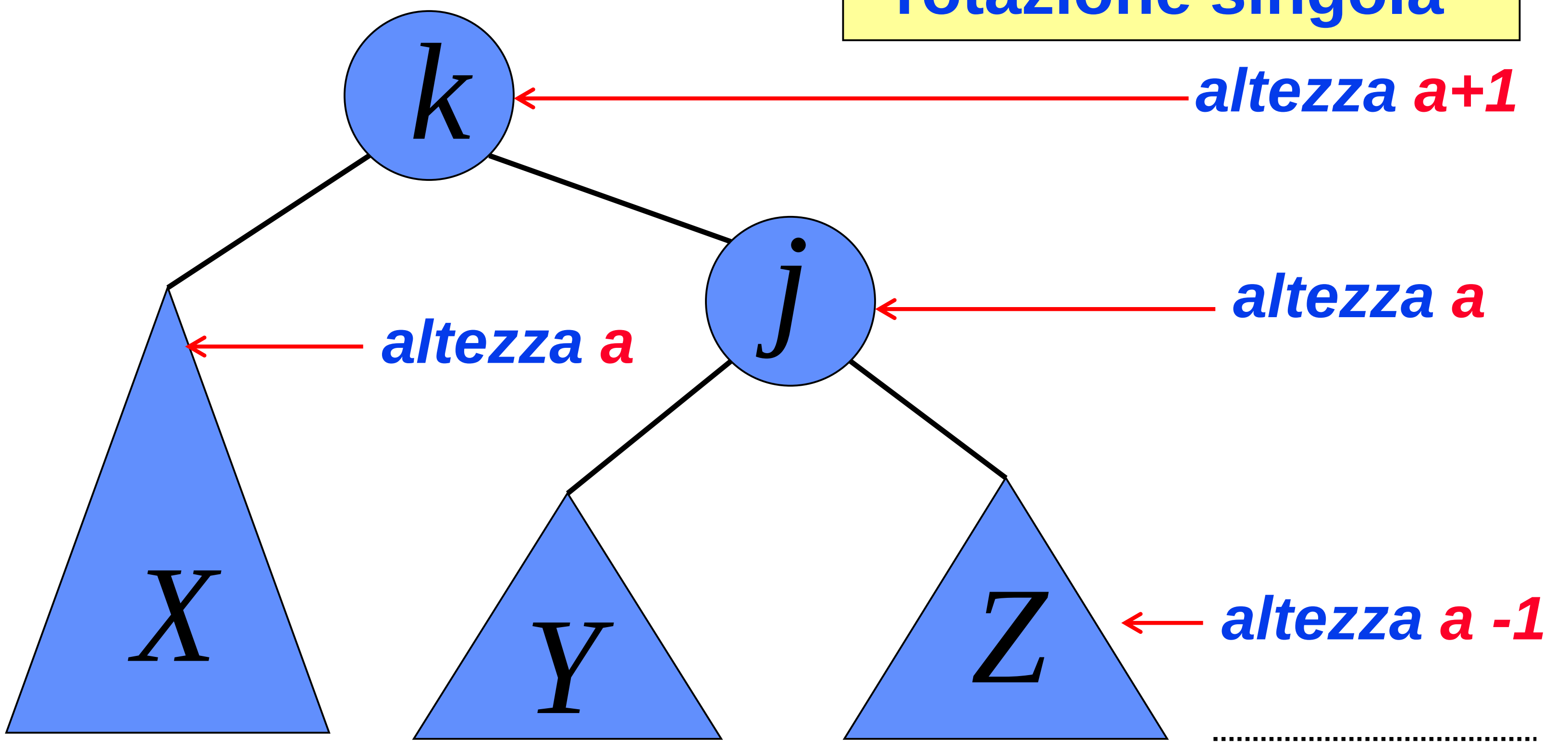
Rotazione singola in alberi AVL



Rotazione singola in alberi AVL

In un singolo passo

Facciamo una "rotazione singola"

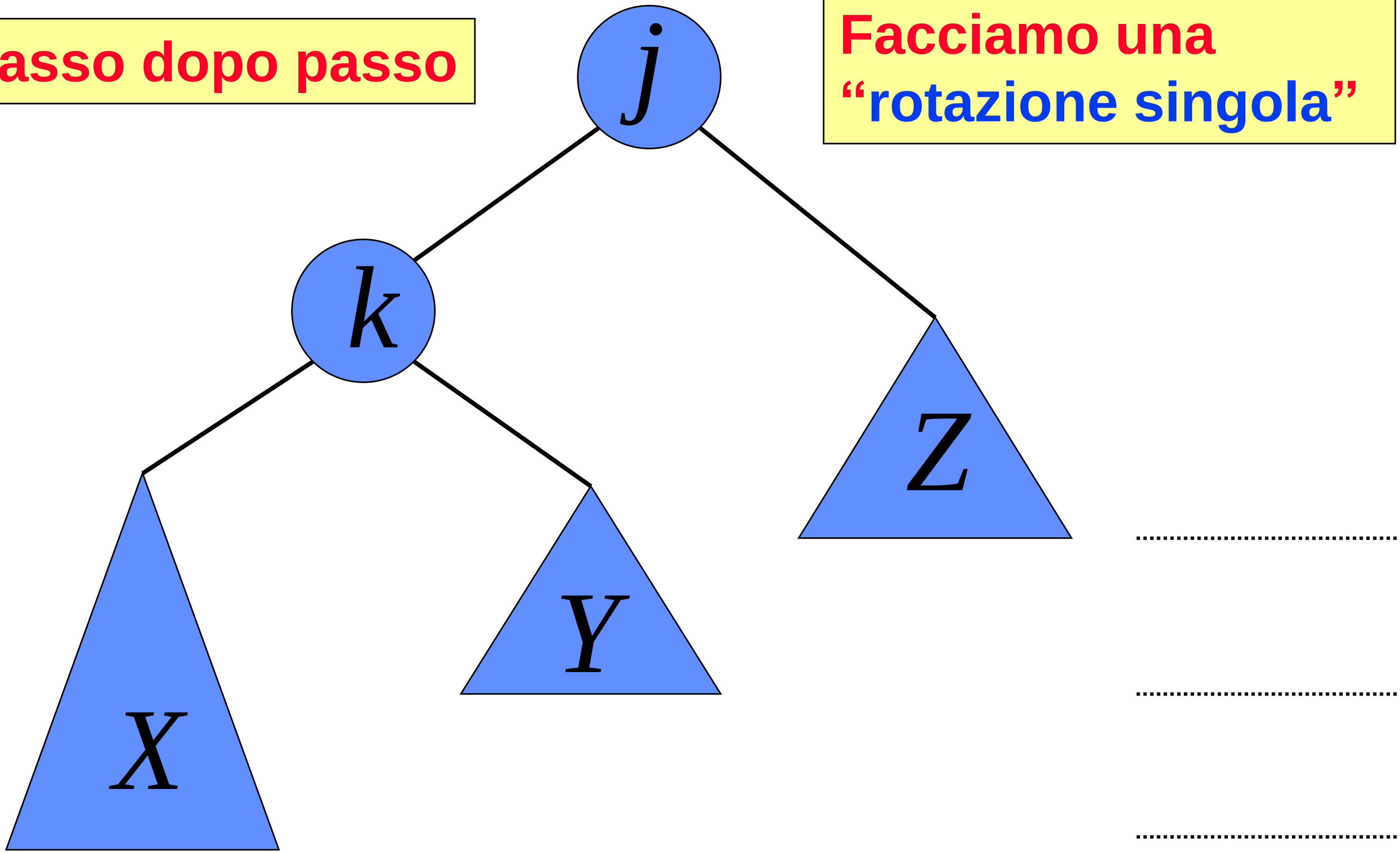


Rotazione singola sinistra

Rotazione singola in alberi AVL

Passo dopo passo

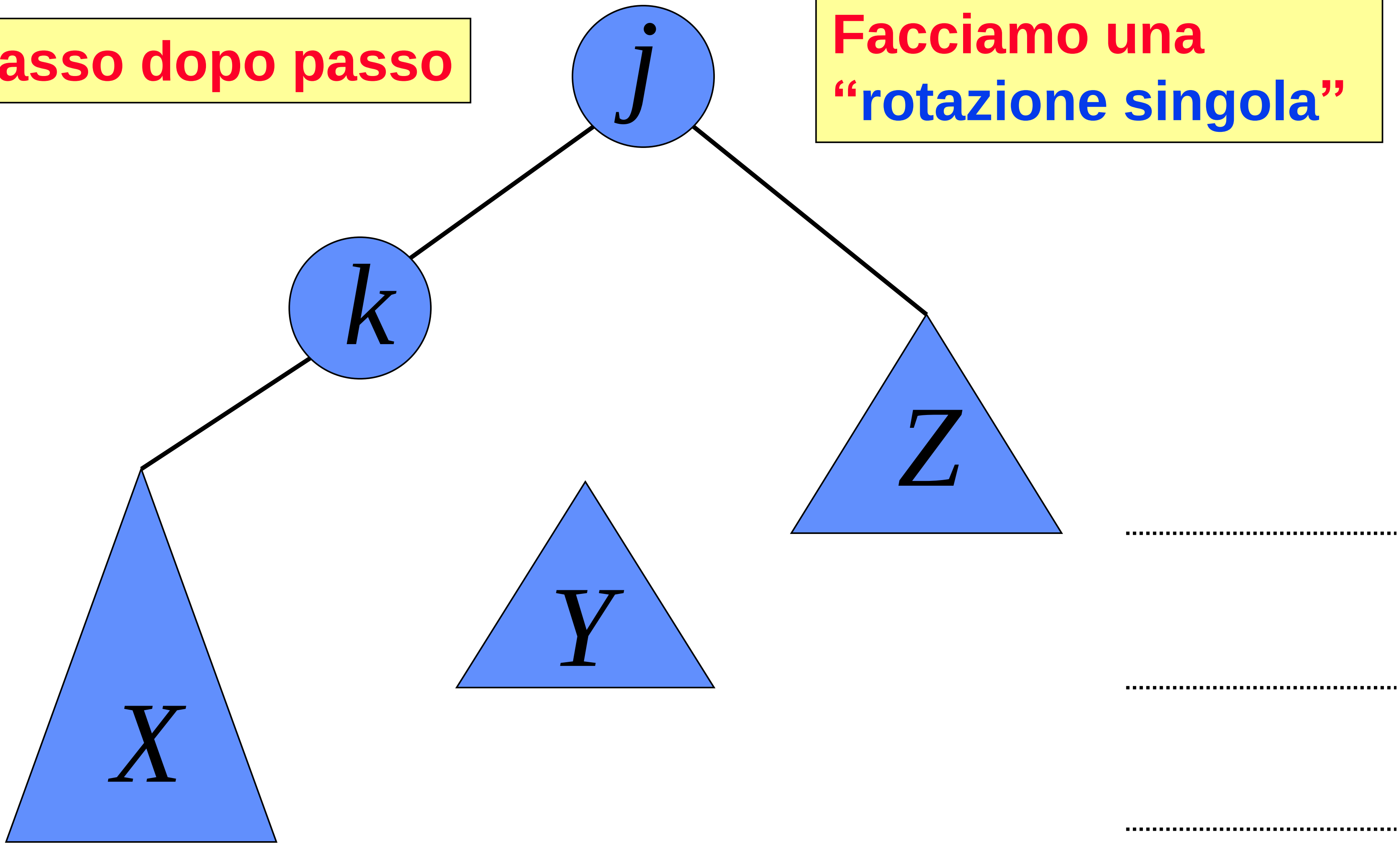
Facciamo una "rotazione singola"



Rotazione singola in alberi AVL

Passo dopo passo

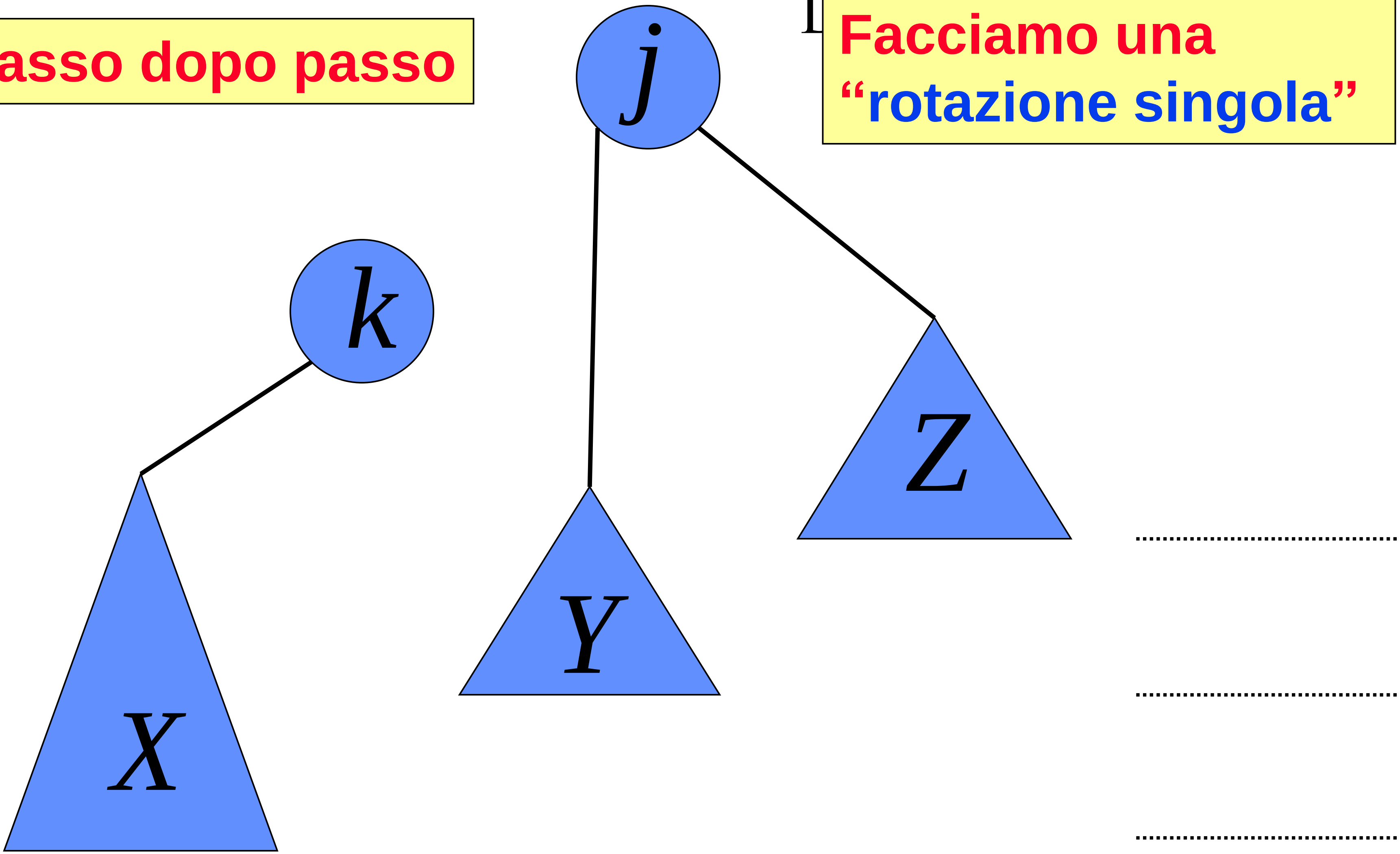
Facciamo una “rotazione singola”



Rotazione singola in alberi AVL

Passo dopo passo

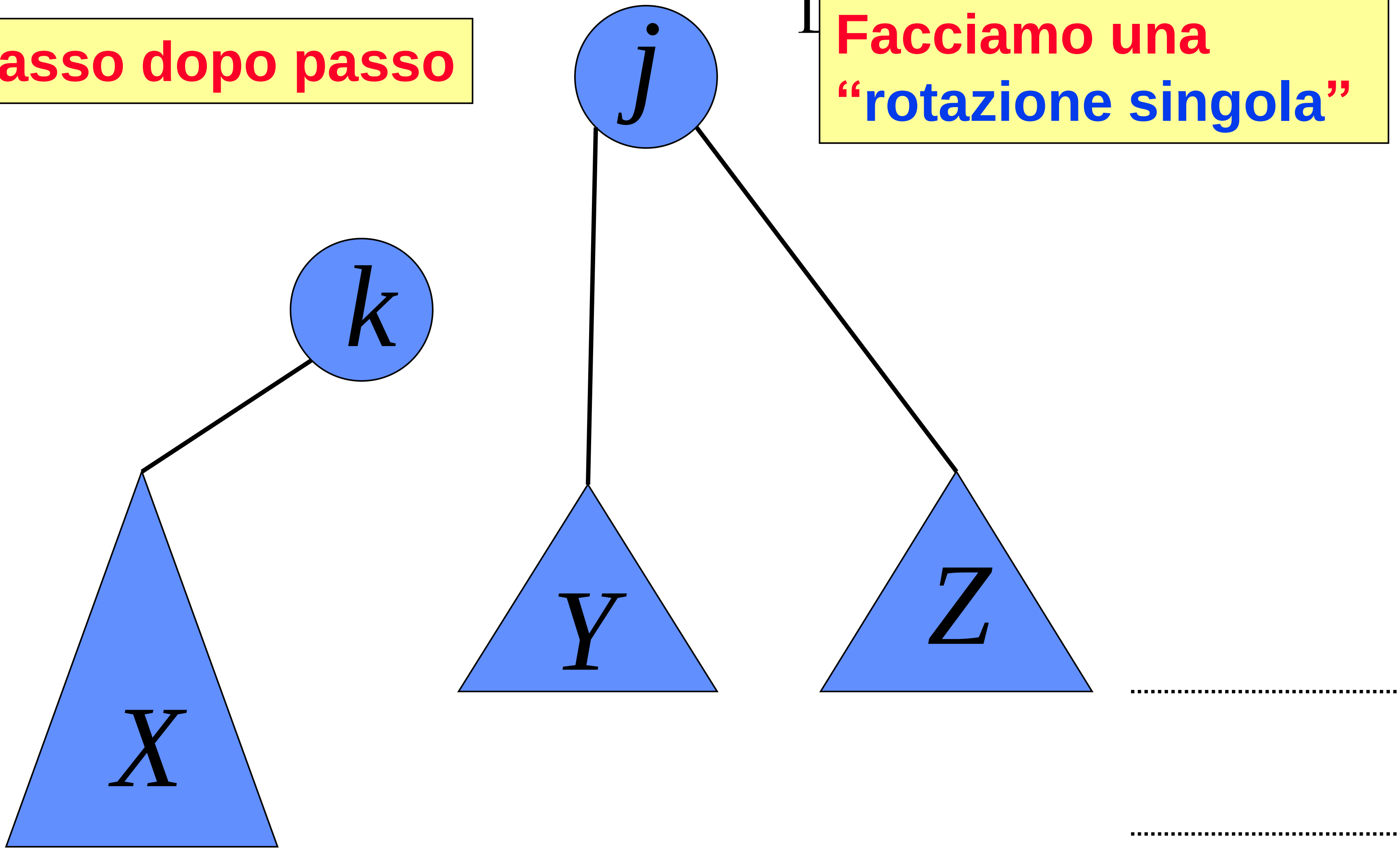
Facciamo una "rotazione singola"



Rotazione singola in alberi AVL

Passo dopo passo

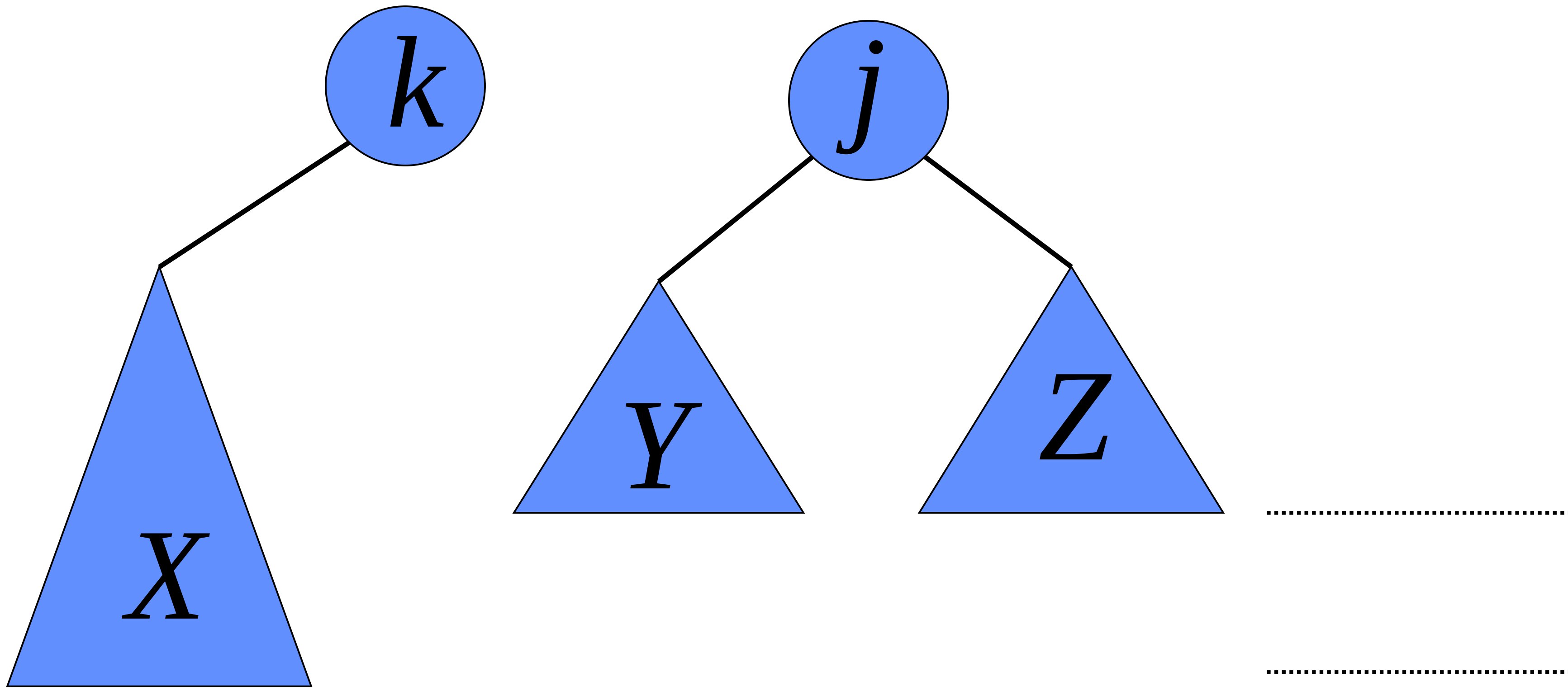
Facciamo una “rotazione singola”



Rotazione singola in alberi AVL

Passo dopo passo

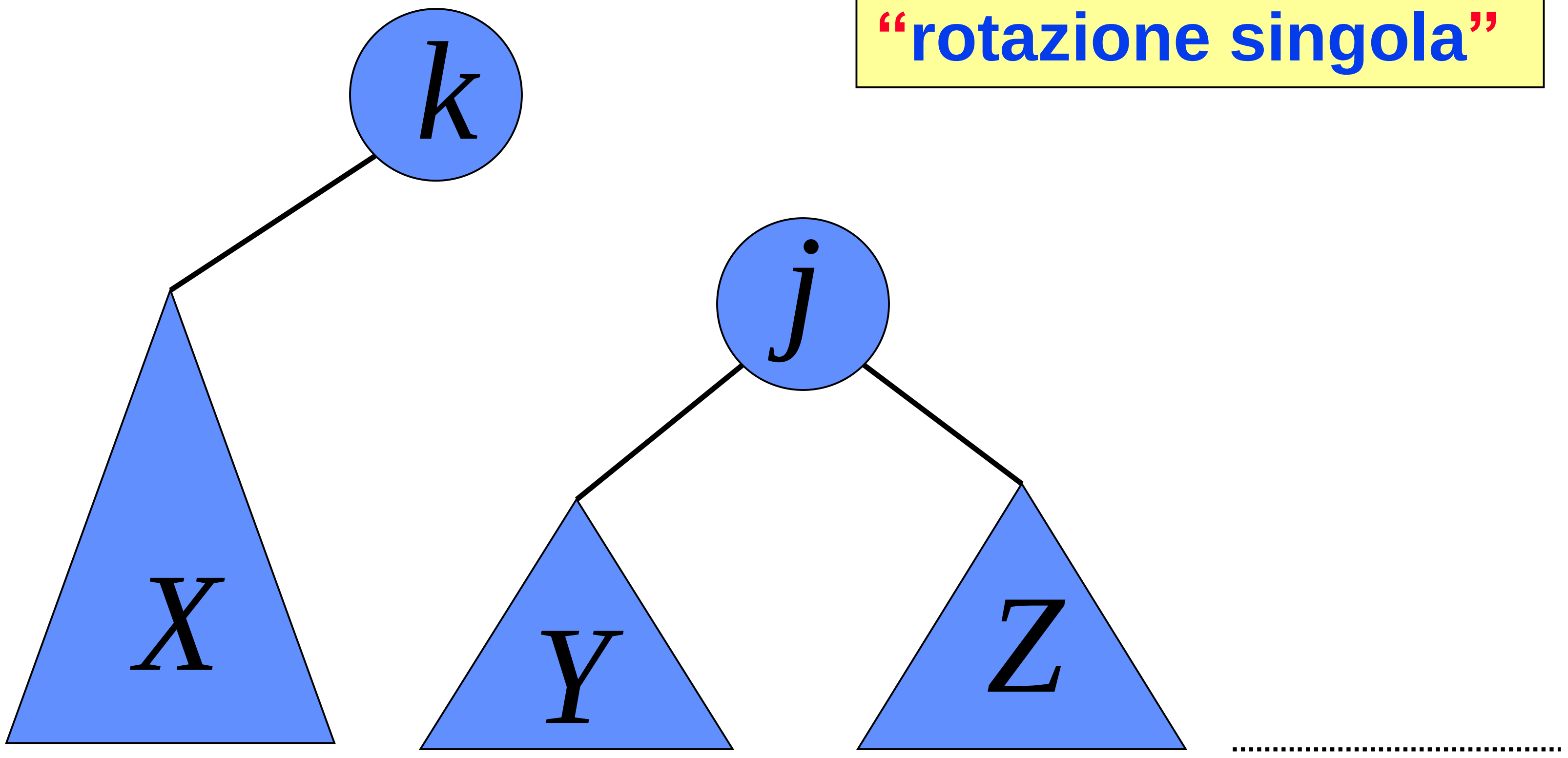
Facciamo una “rotazione singola”



Rotazione singola in alberi AVL

Passo dopo passo

Facciamo una "rotazione singola"

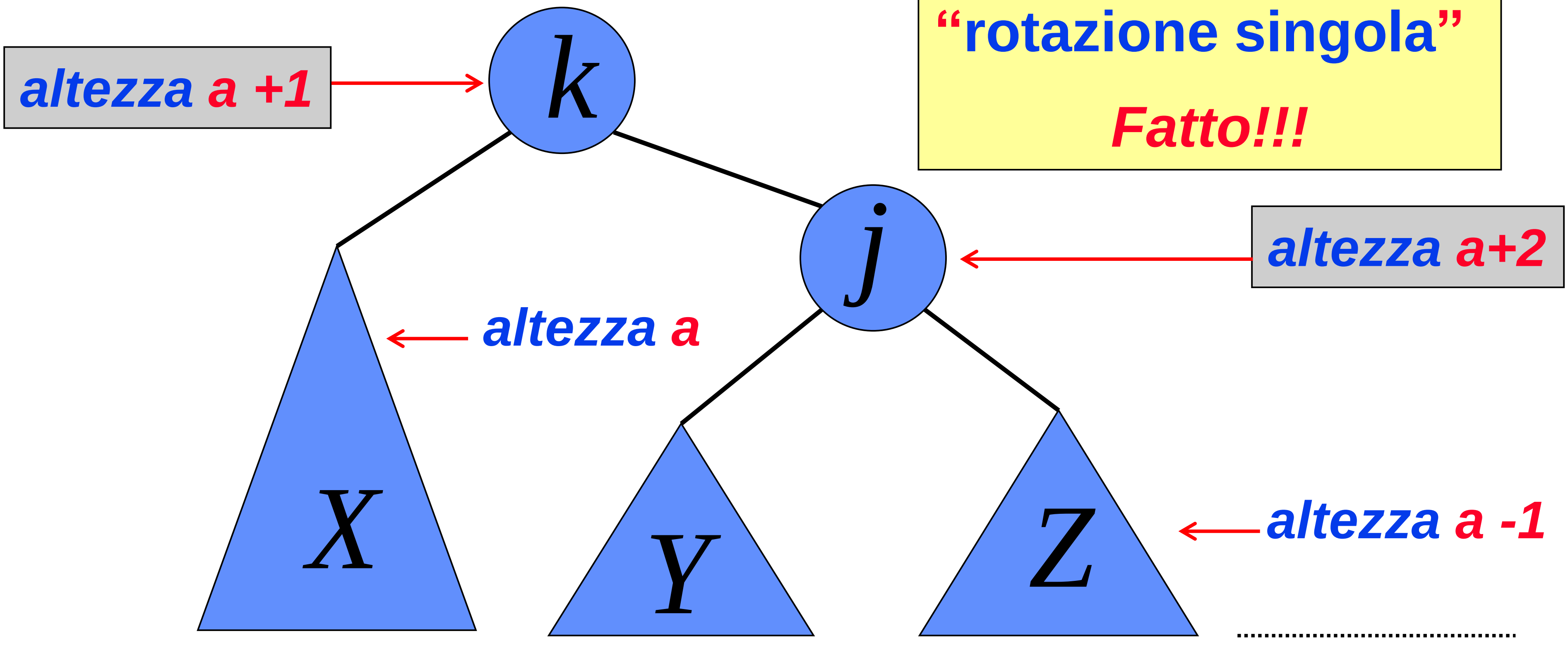


Rotazione singola in alberi AVL

Passo dopo passo

Facciamo una
"rotazione singola"

Fatto!!!



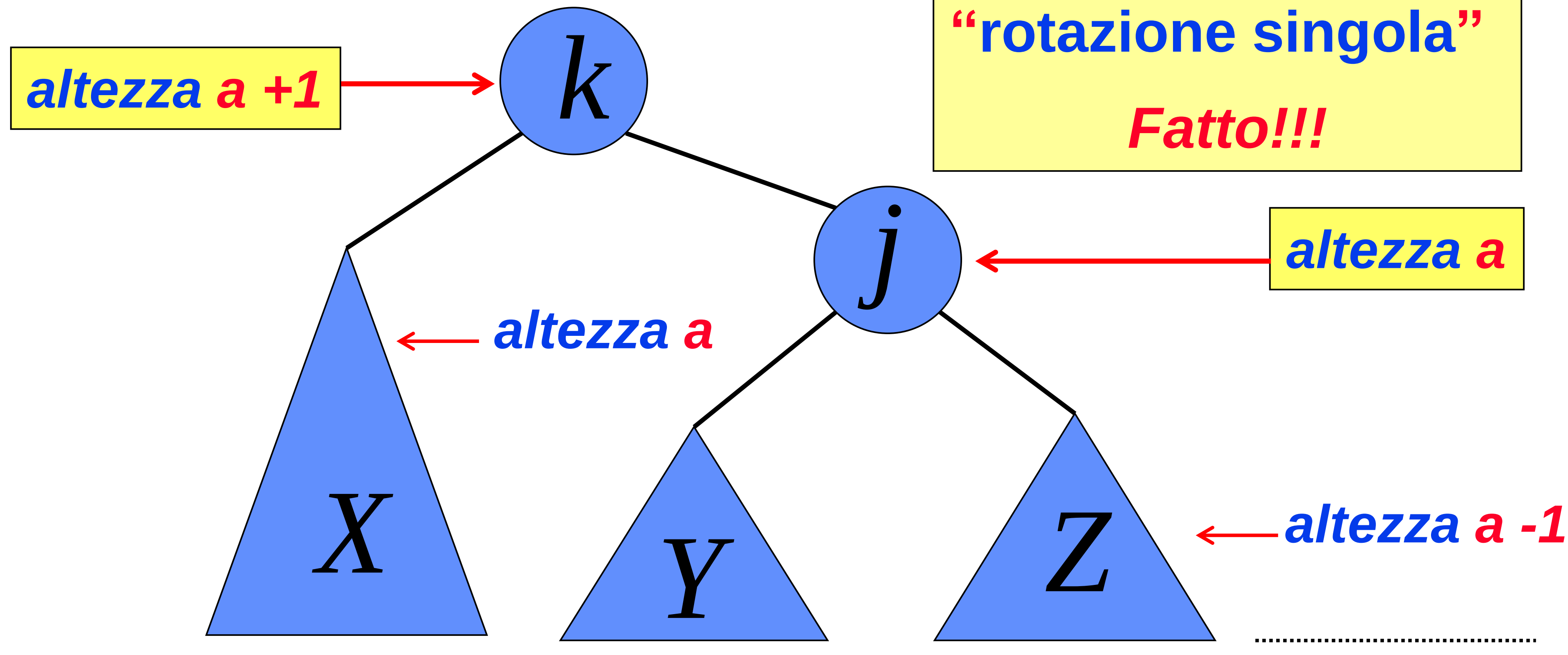
Le altezze dei nodi *K* e *J* sono state aggiornate

Rotazione singola in alberi AVL

Passo dopo passo

Facciamo una "rotazione singola"

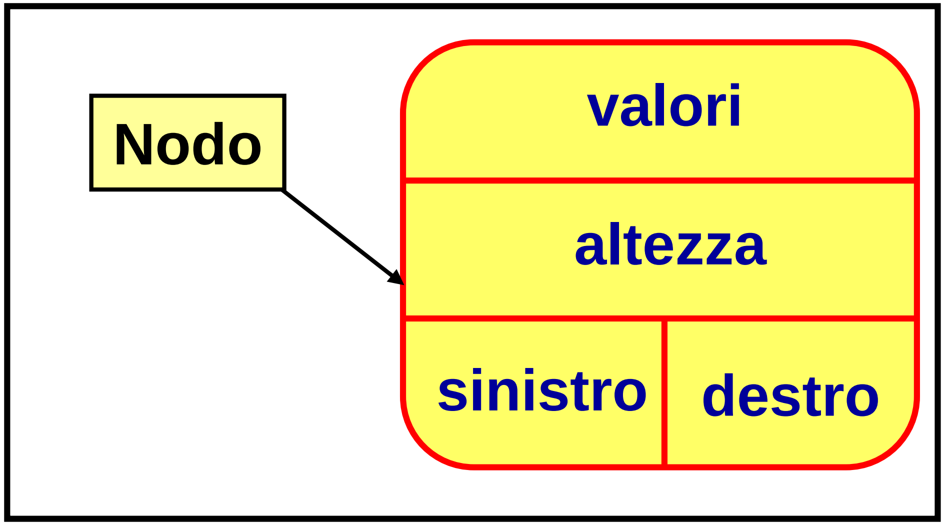
Fatto!!!



Le altezze dei nodi *K* e *J* sono state aggiornate

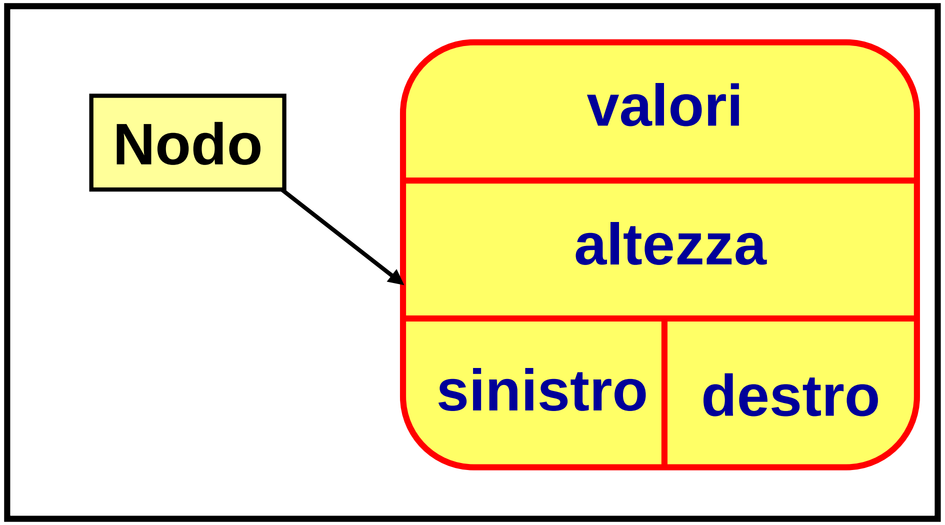
Rotazione singola: algoritmo

```
Altezza(T : albero-AVL)
   IF T = NIL THEN
        return -1
   ELSE
        return T->altezza
```



Rotazione singola: algoritmo

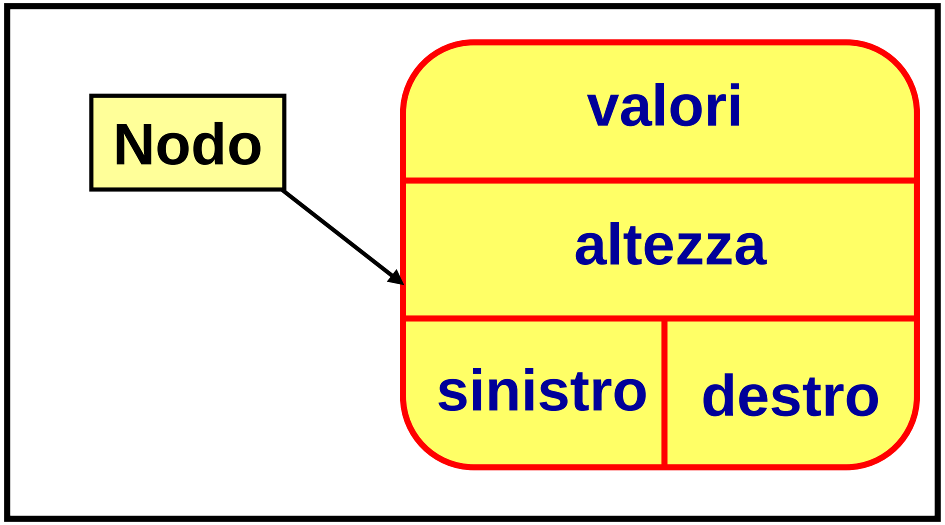
```
Altezza(T : albero-AVL)
  IF T = NIL THEN
      return -1
  ELSE
      return T->altezza
```




```
Rotazione-SS(T : albero-AVL)
    newroot = T->sx
    T->sx = newroot->dx
    newroot->dx = T
    T->altezza = max(Altezza(T->sx), Altezza(T->dx)) + 1
    newroot->altezza = max(Altezza(newroot->sx),
                           Altezza(newroot->dx)) + 1
    return newroot
```

Rotazione singola: algoritmo

```
Altezza(T : albero-AVL)
   IF T = NIL
       THEN return -1
       ELSE return T->altezza
```

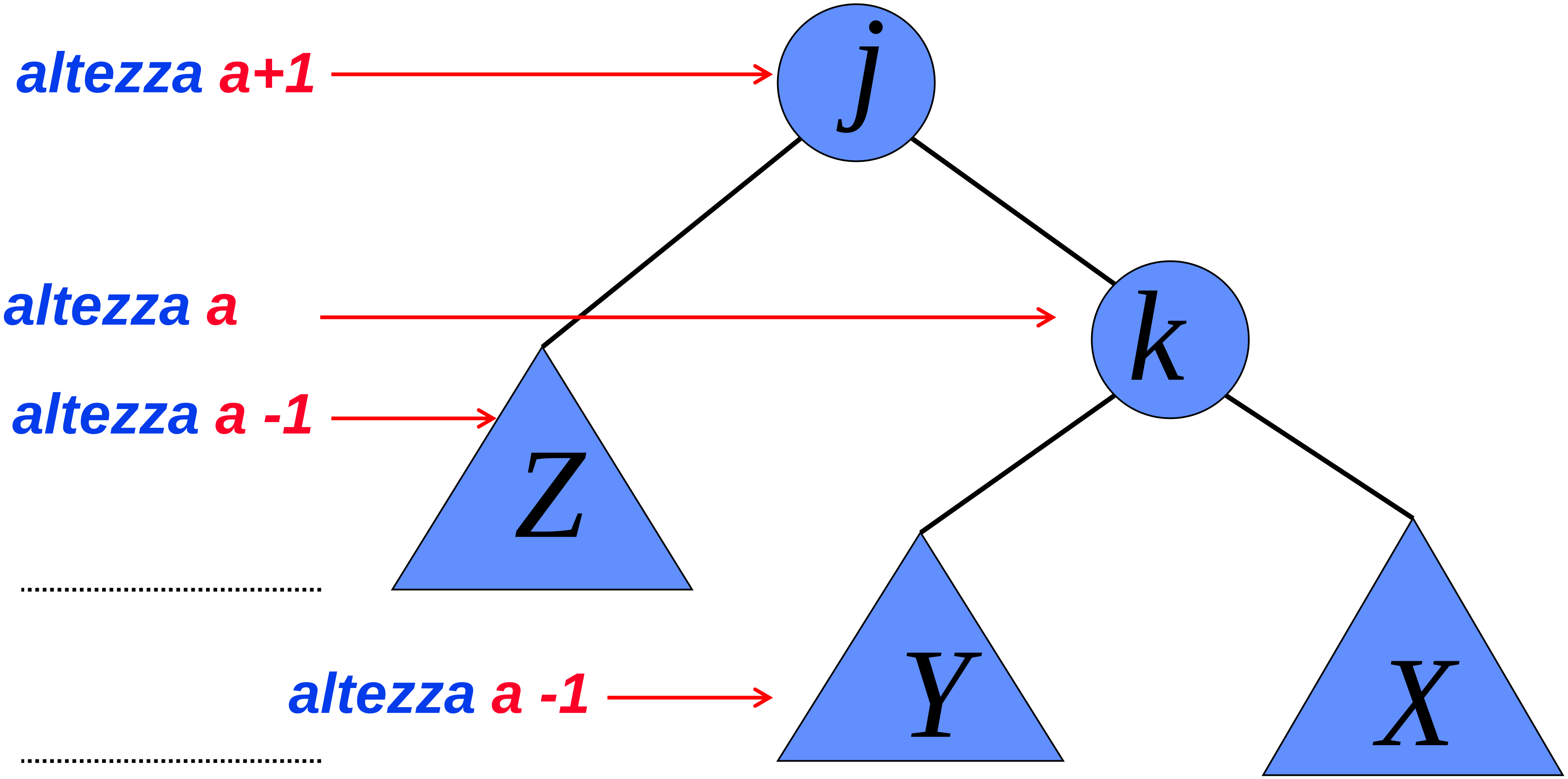



```
Rotazione-SS(T : albero-AVL)
    newroot = T->sx
    T->sx = newroot->dx
    newroot->dx = T
    T->altezza = max(Altezza(T->sx), Altezza(T->dx)) + 1
    newroot->altezza = max(Altezza(newroot->sx),
                           Altezza(newroot->dx)) + 1
    return newroot
```

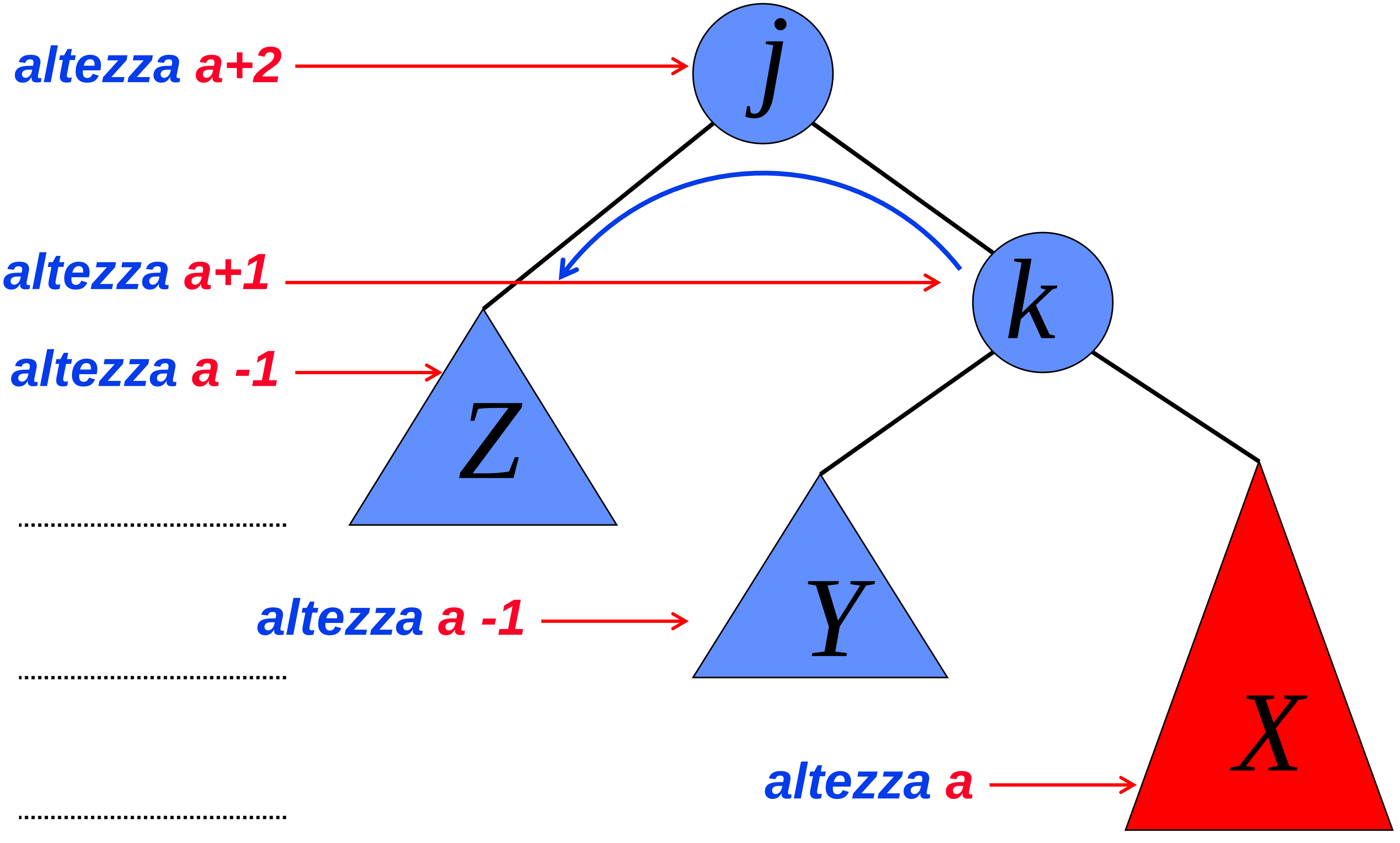
Rotazione

Aggiornamento altezze

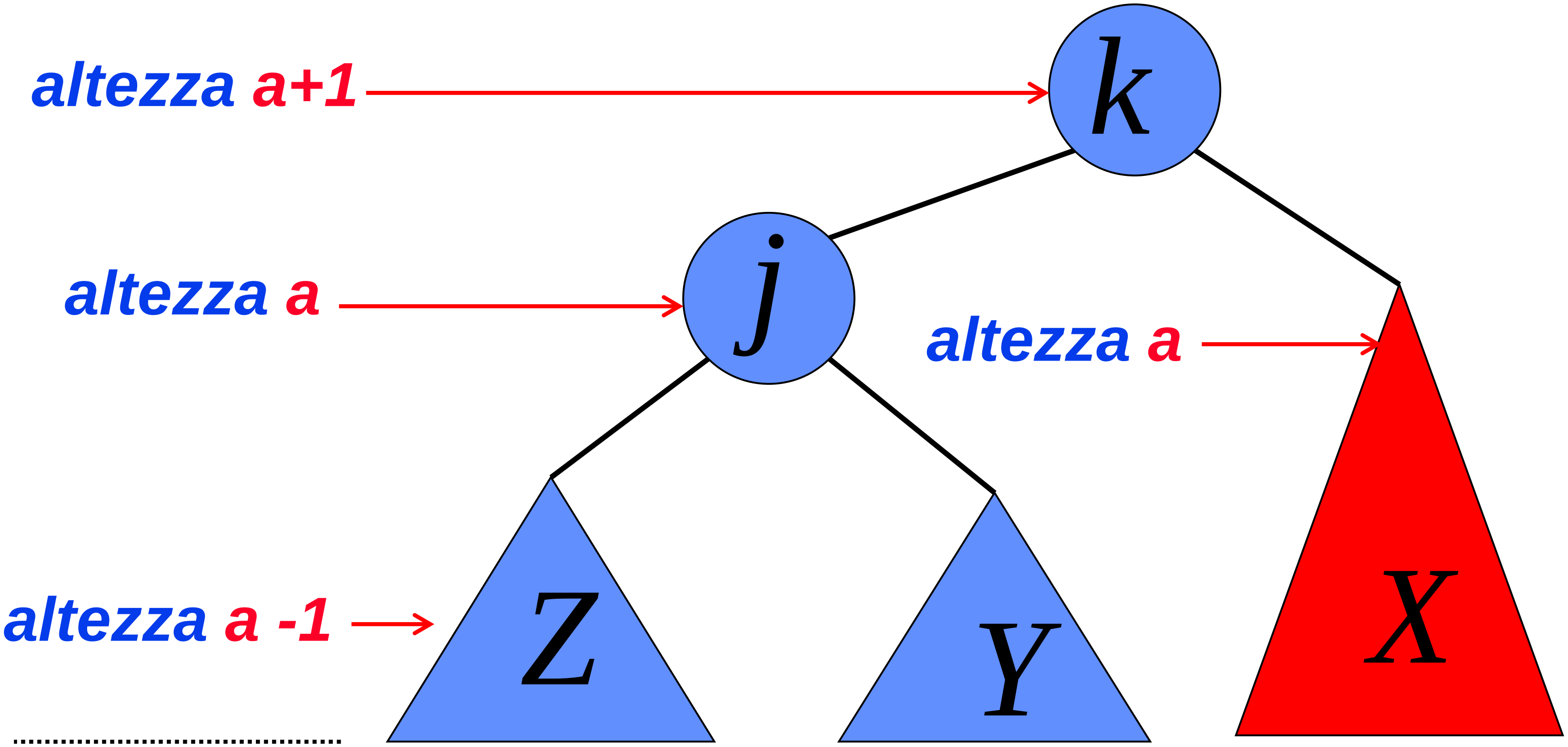
Rotazione in alberi AVL: caso I



Rotazione in alberi AVL: caso I



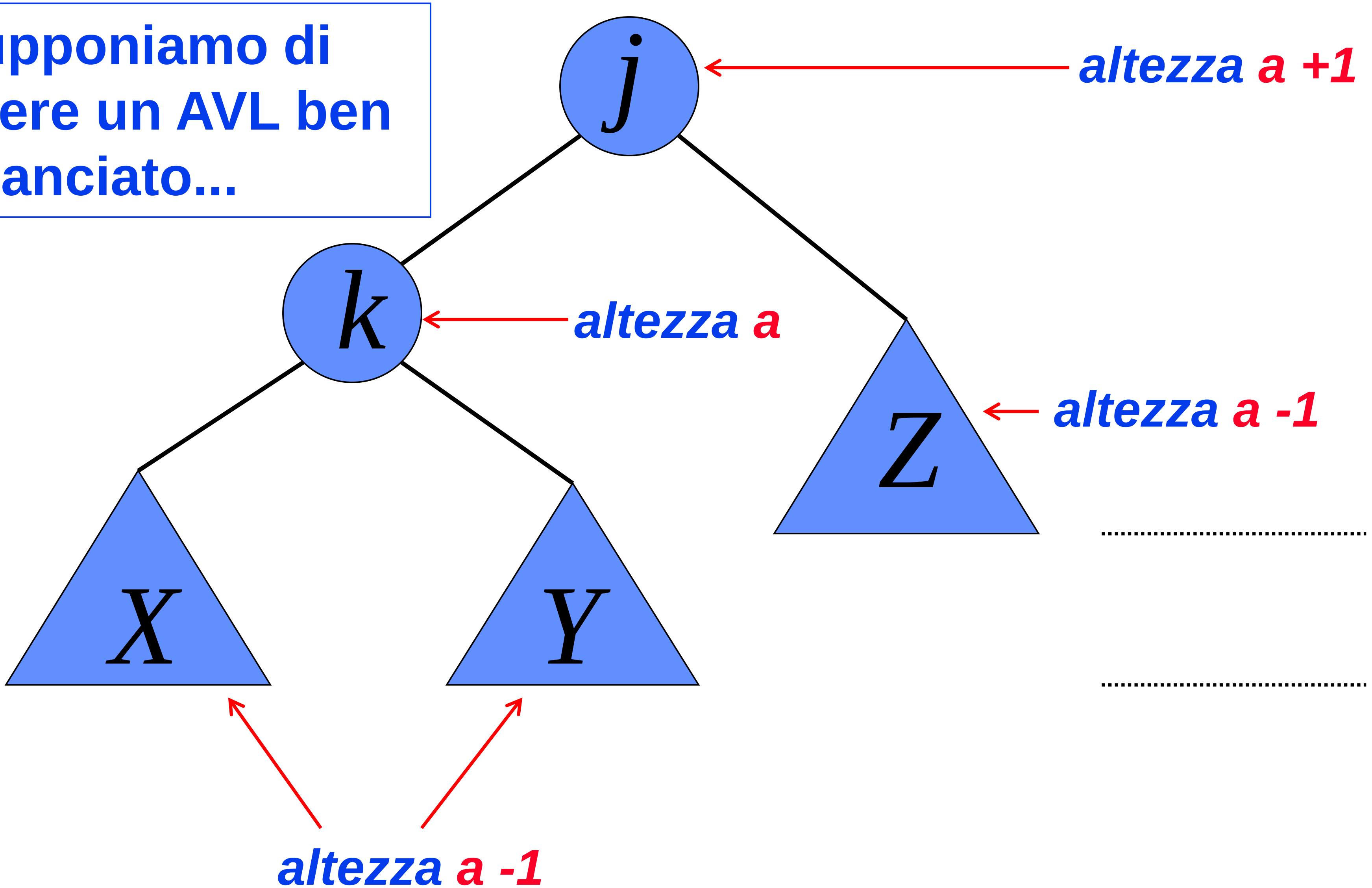
Rotazione in alberi AVL: caso I



Rotazione singola destra

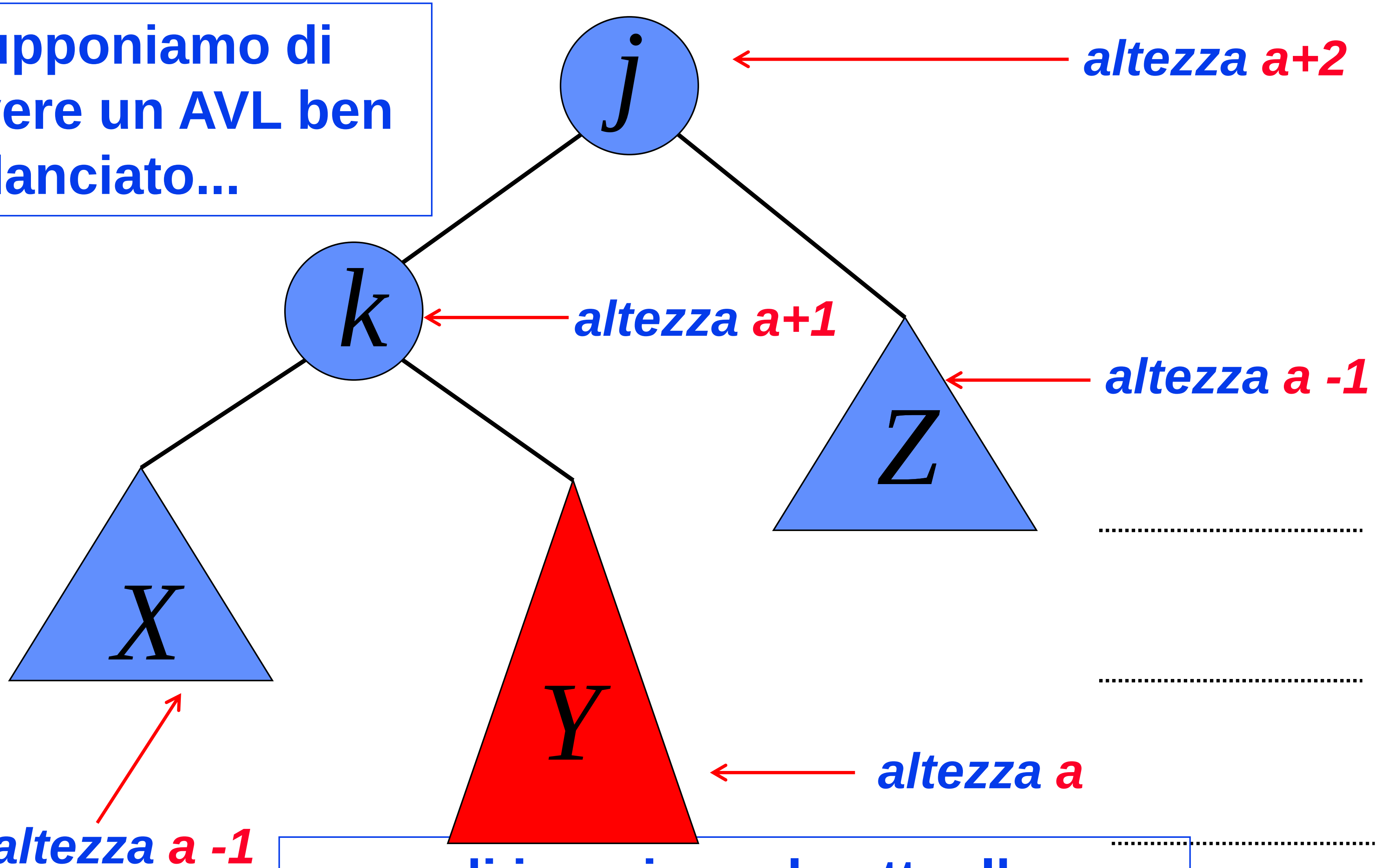
Rotazione in alberi AVL: caso II

Supponiamo di avere un AVL ben bilanciato...



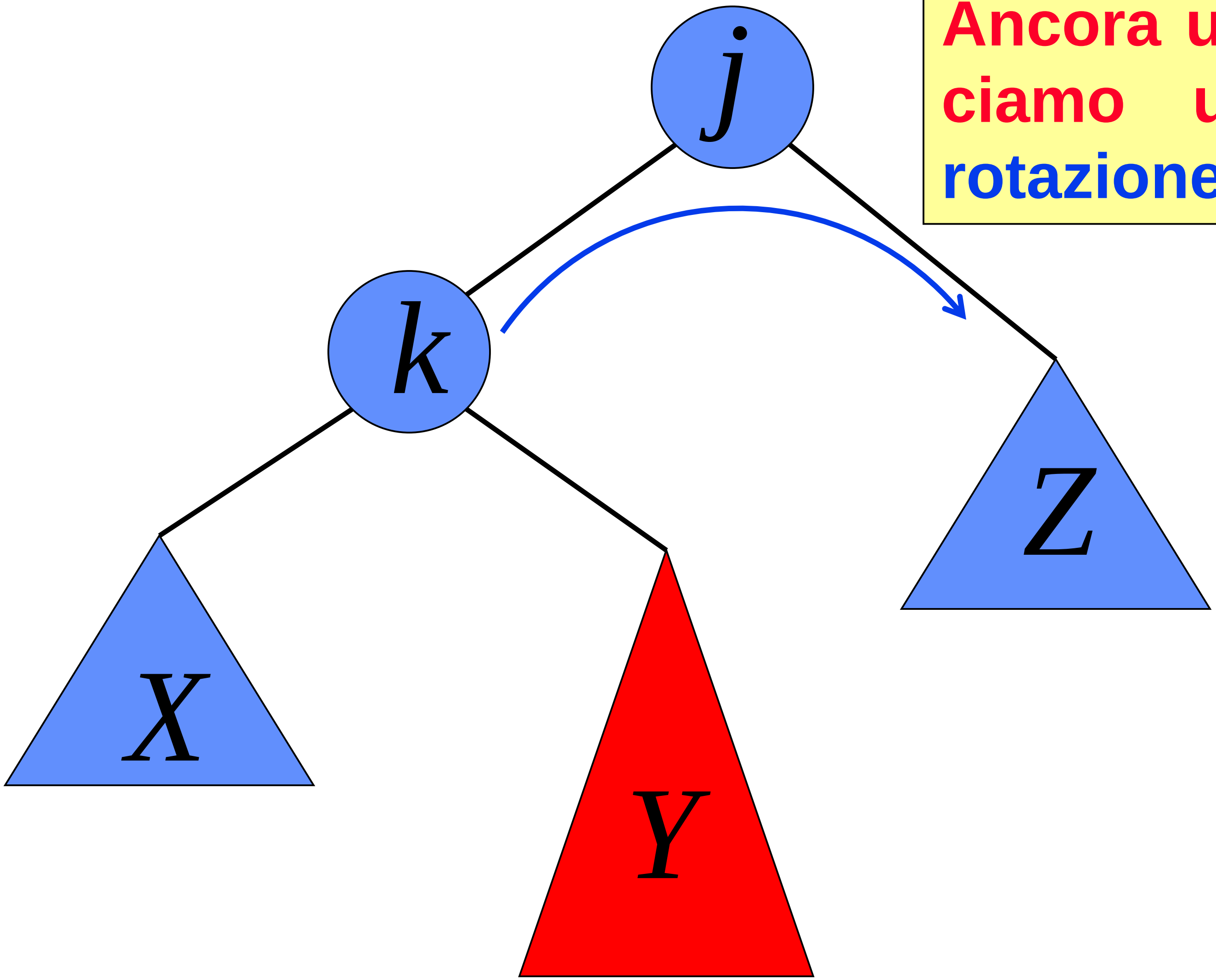
Rotazione in alberi AVL: caso II

Supponiamo di avere un AVL ben bilanciato...



... ma di inserire nel sottoalbero *Y*, perdendo così il bilanciamento.

Rotazione in alberi AVL: caso II



Ancora una volta, facciamo una "singola rotazione

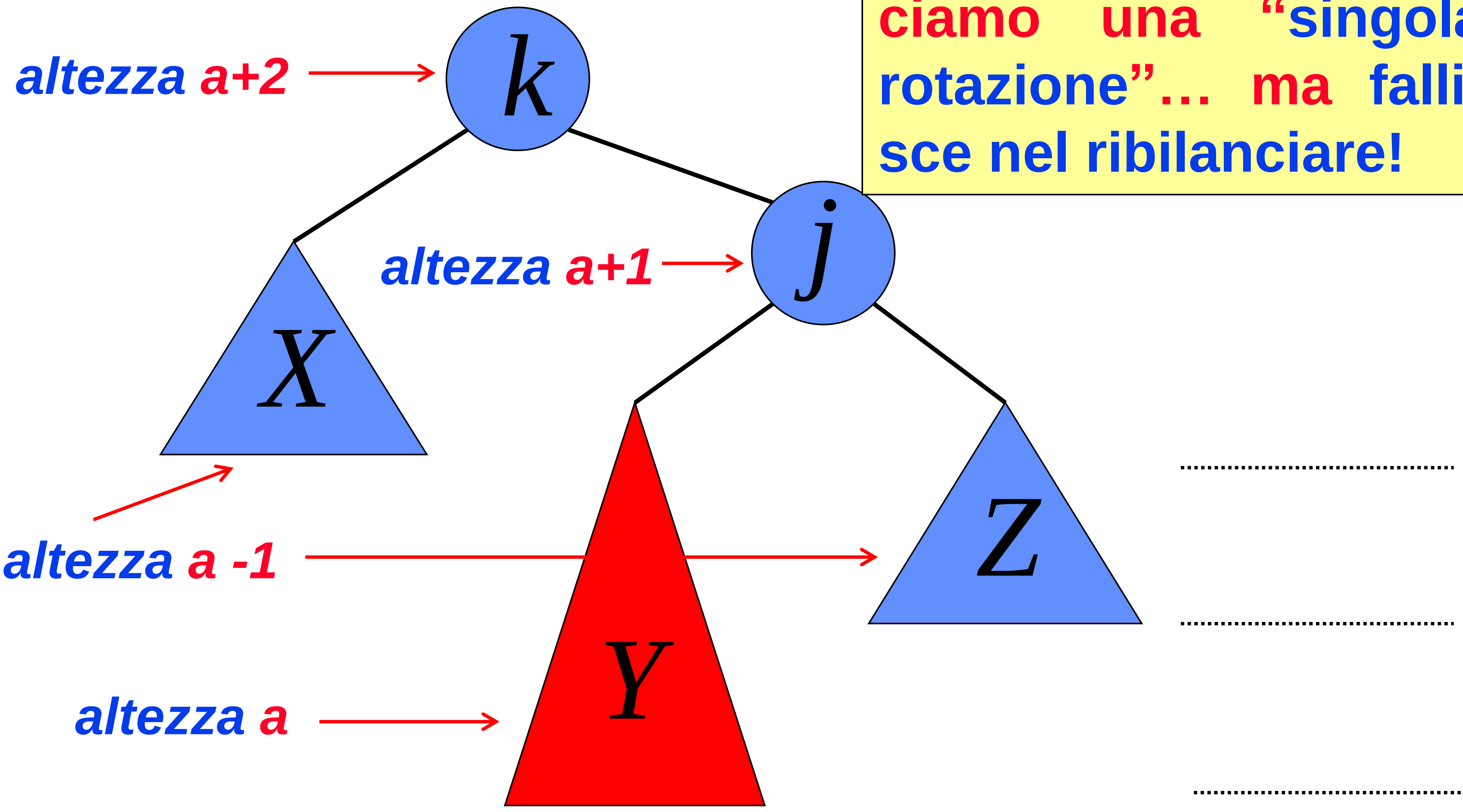
....

....

....

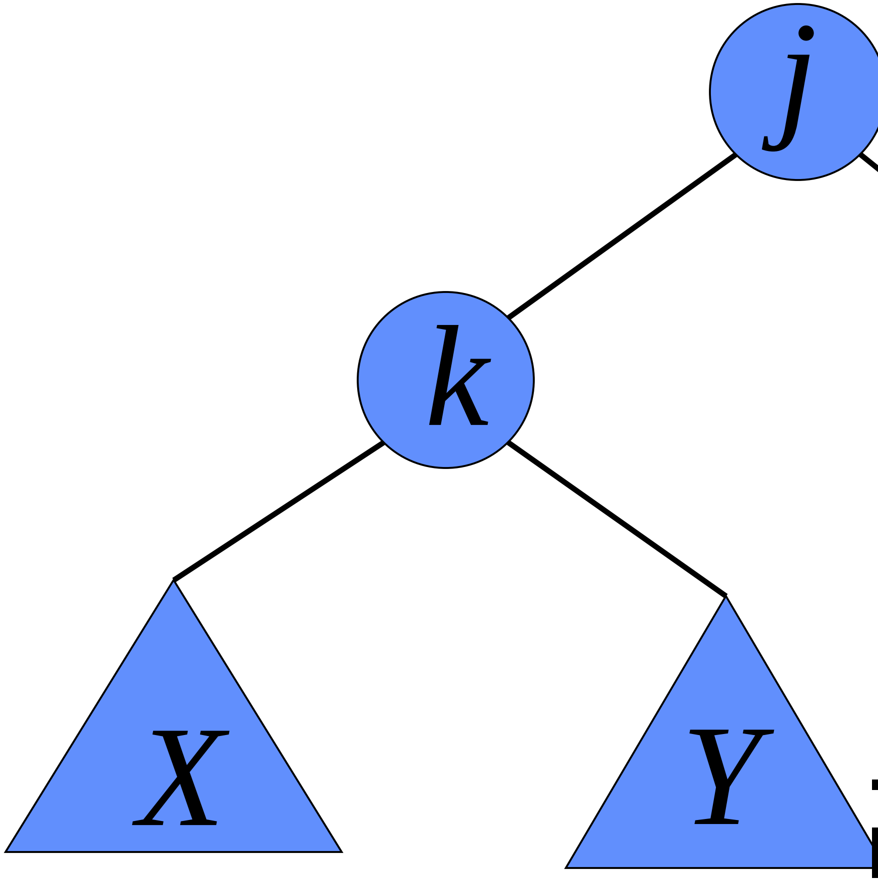
Rotazione in alberi AVL: caso II

Ancora una volta, facciamo una "singola rotazione"… ma fallisce nel ribilanciare!



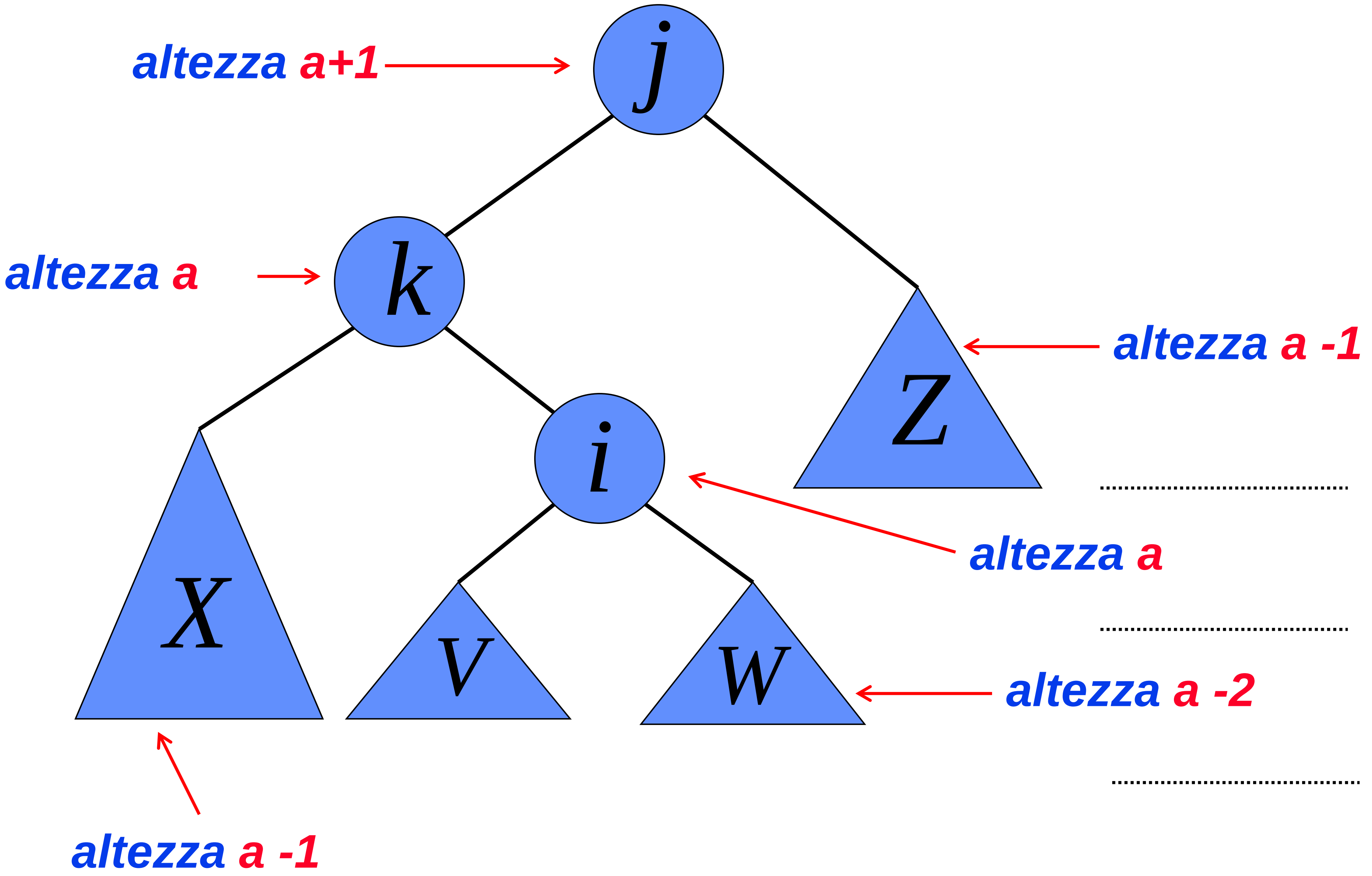
Rotazione in alberi AVL: caso II

Torniamo indietro da dove siamo partiti!

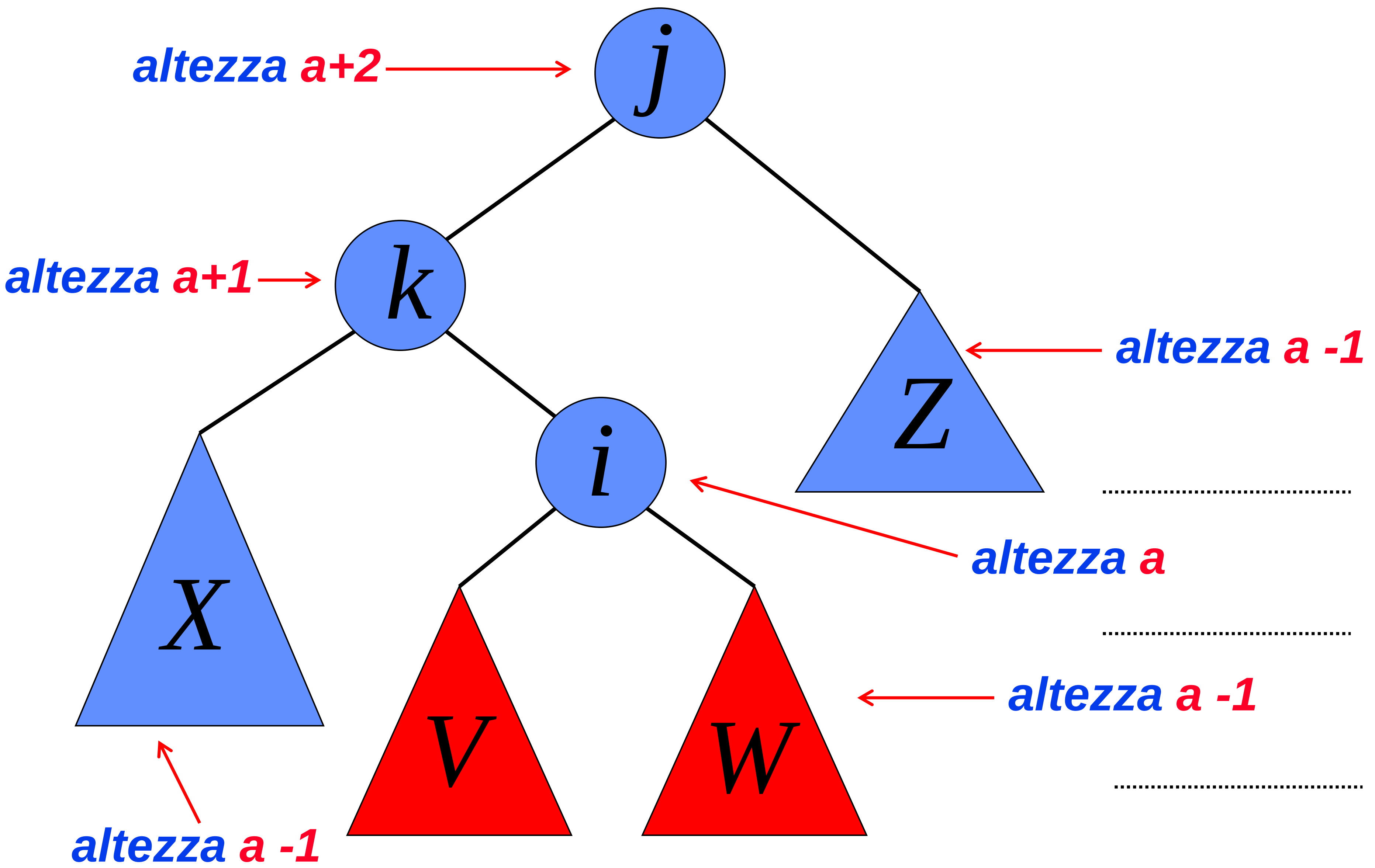


... e consideriamo la struttura di *Y*

Rotazione in alberi AVL: caso II

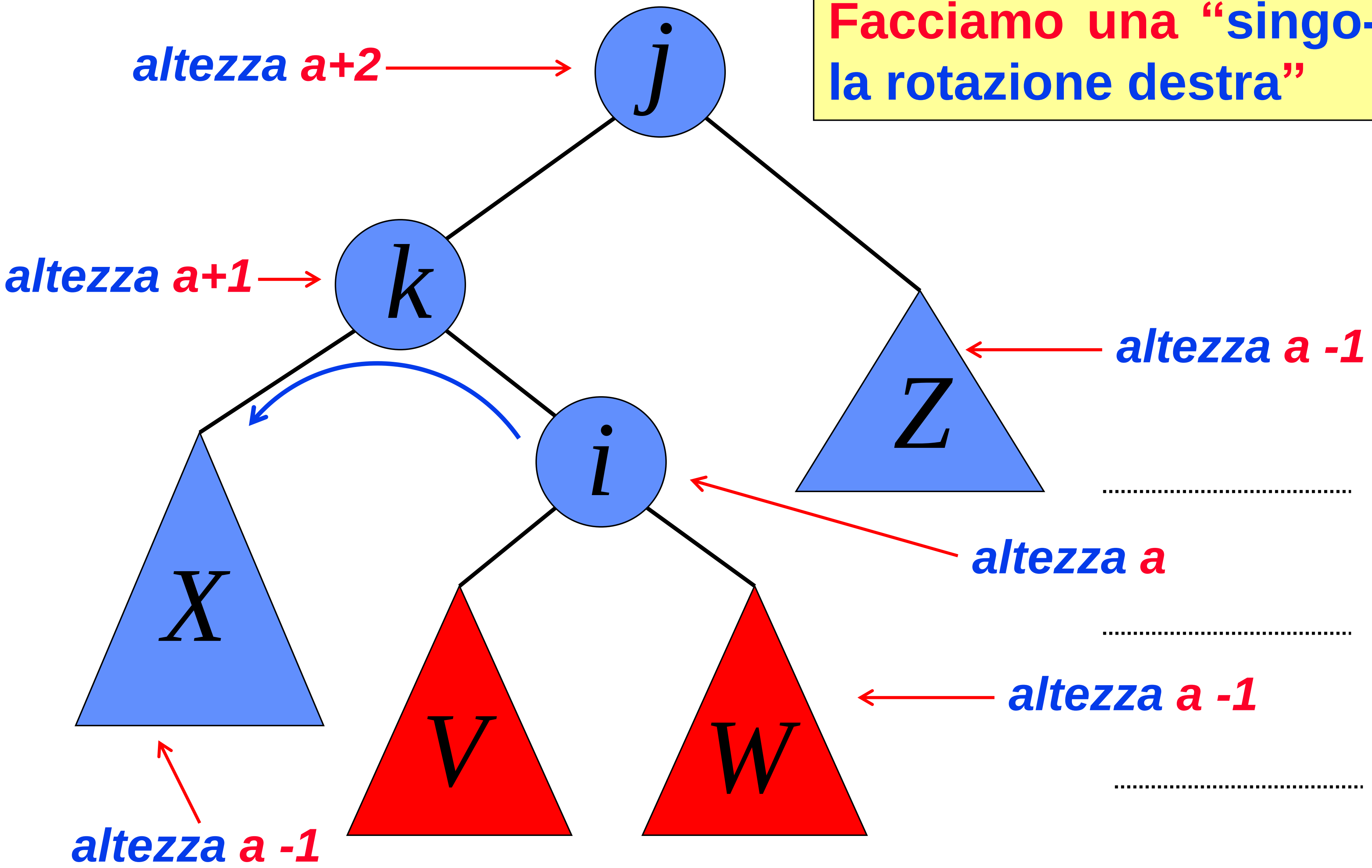


Rotazione in alberi AVL: caso II

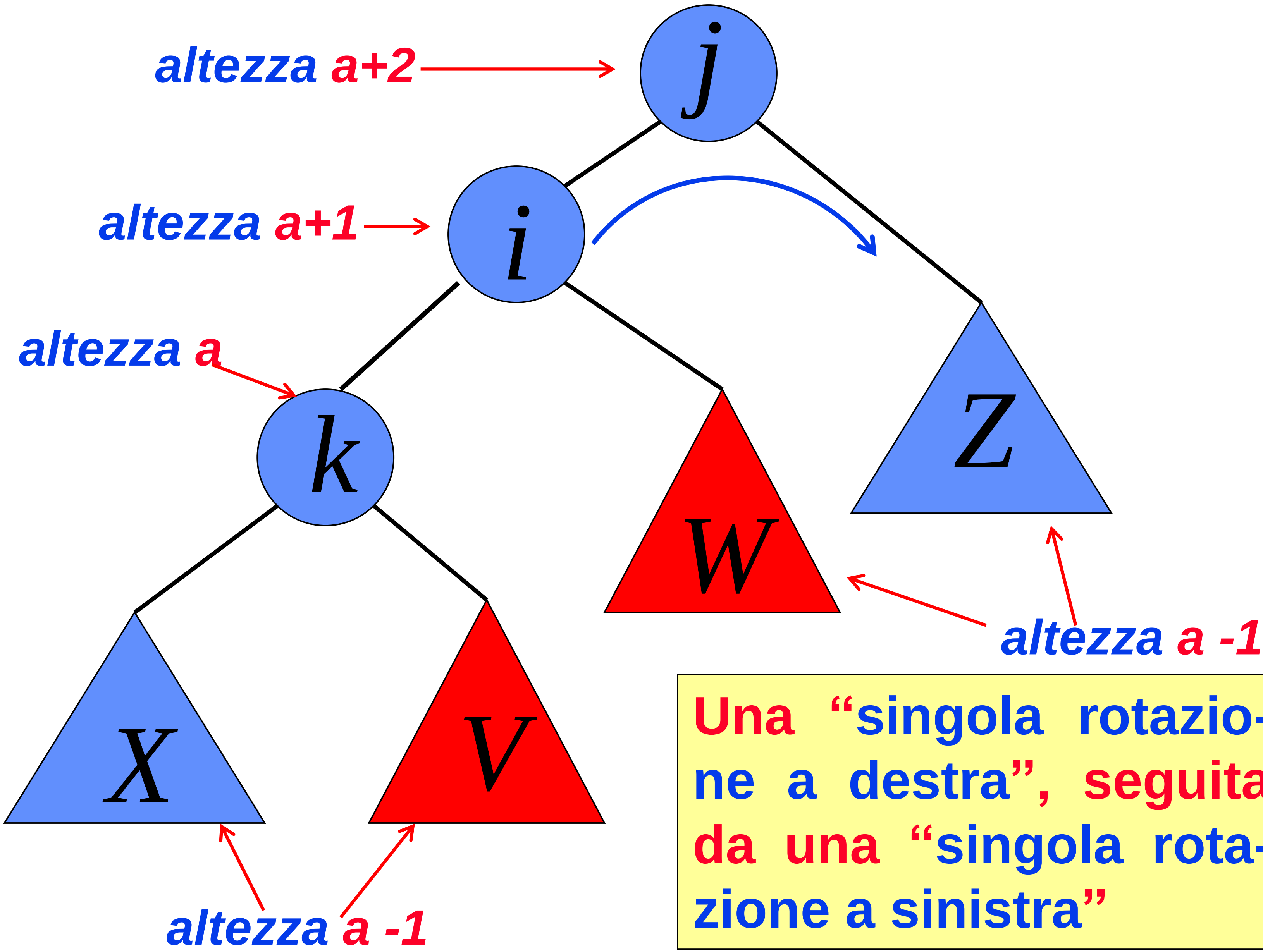


Rotazione in alberi AVL: caso II

Facciamo una "singola rotazione destra"

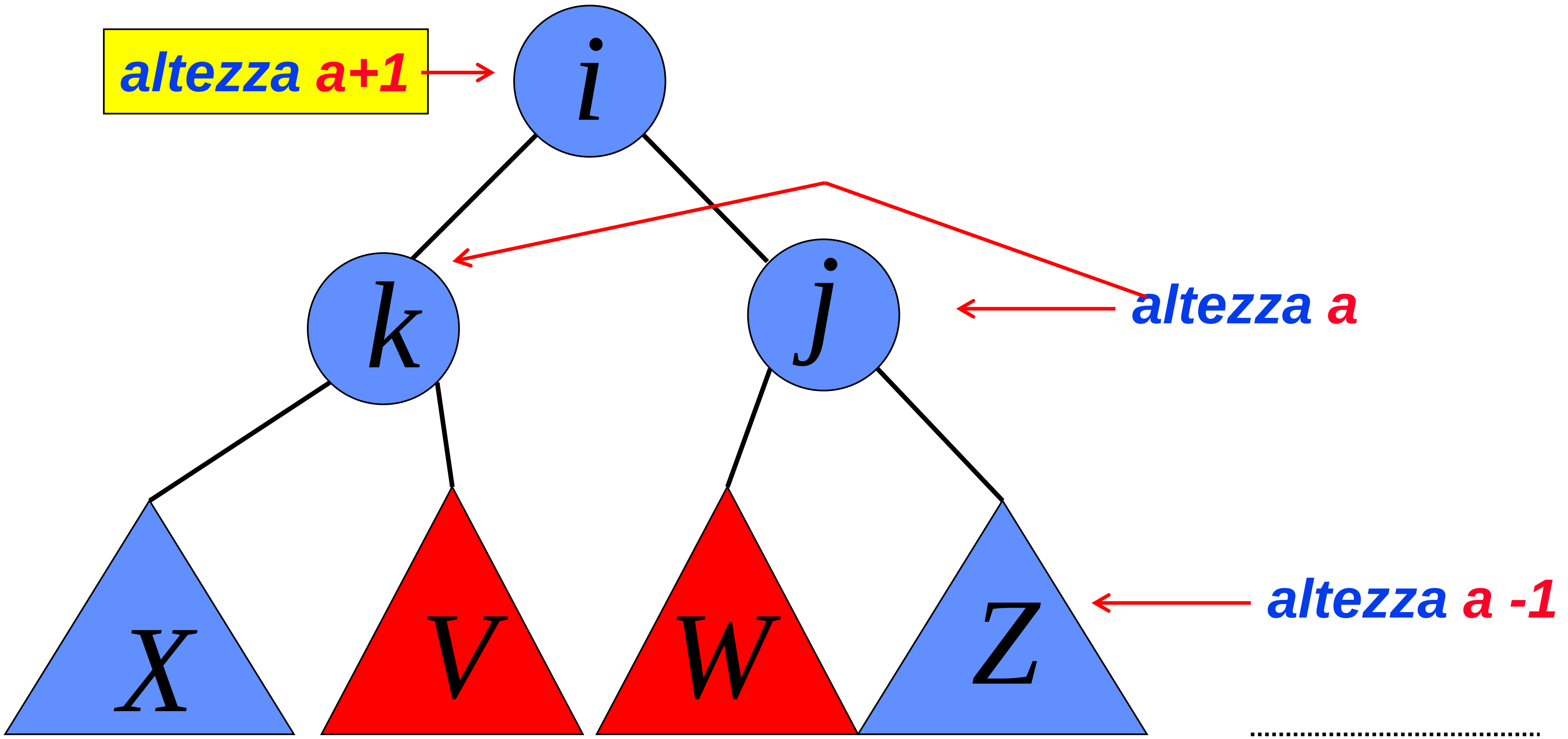


Rotazione in alberi AVL: caso II



Una "singola rotazione a destra", seguita da una "singola rotazione a sinistra"

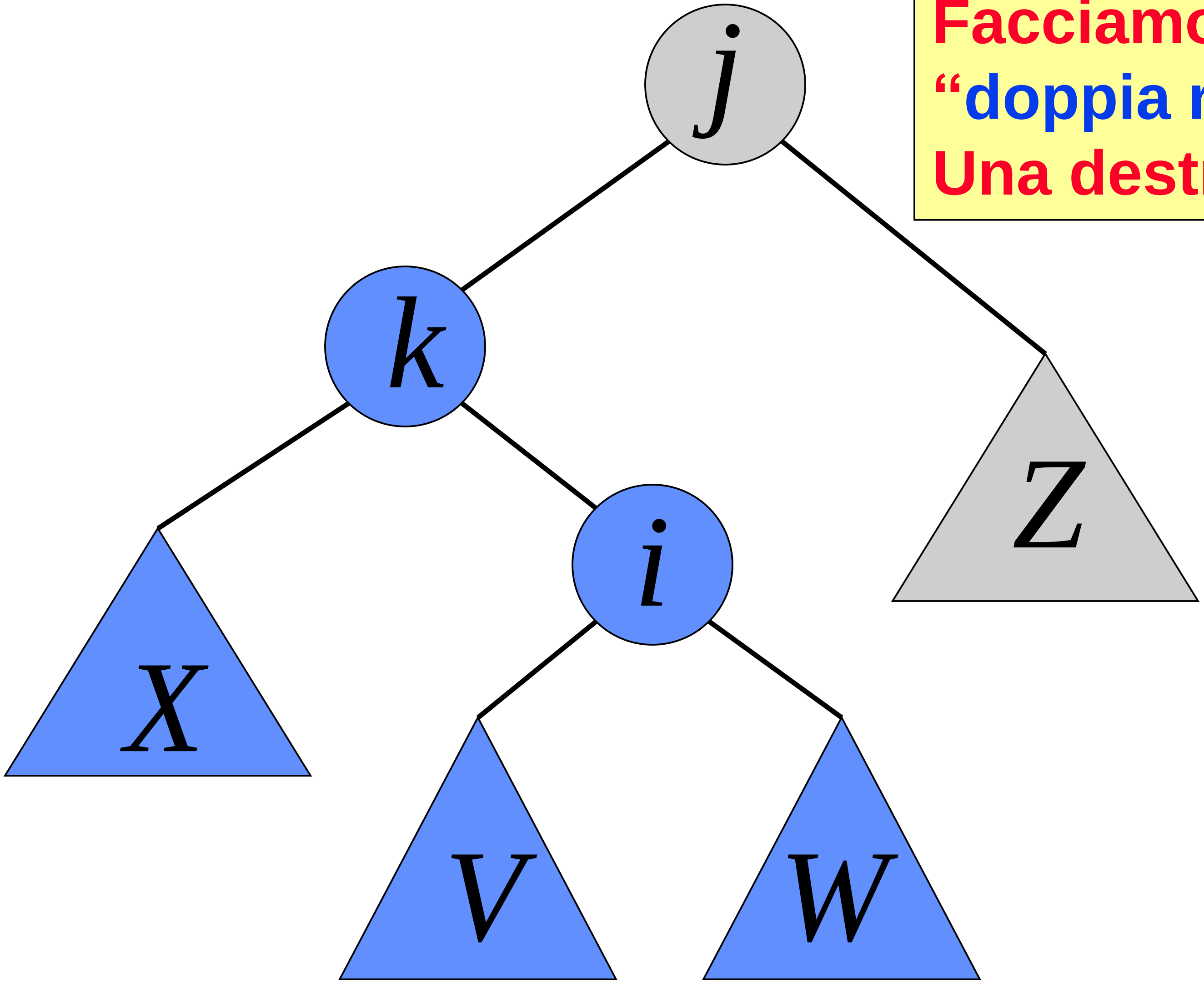
Rotazione in alberi AVL: caso II



Rotazione doppia sinistra

Rotazione doppia in alberi AVL

Facciamo ora una "doppia rotazione"
Una destra

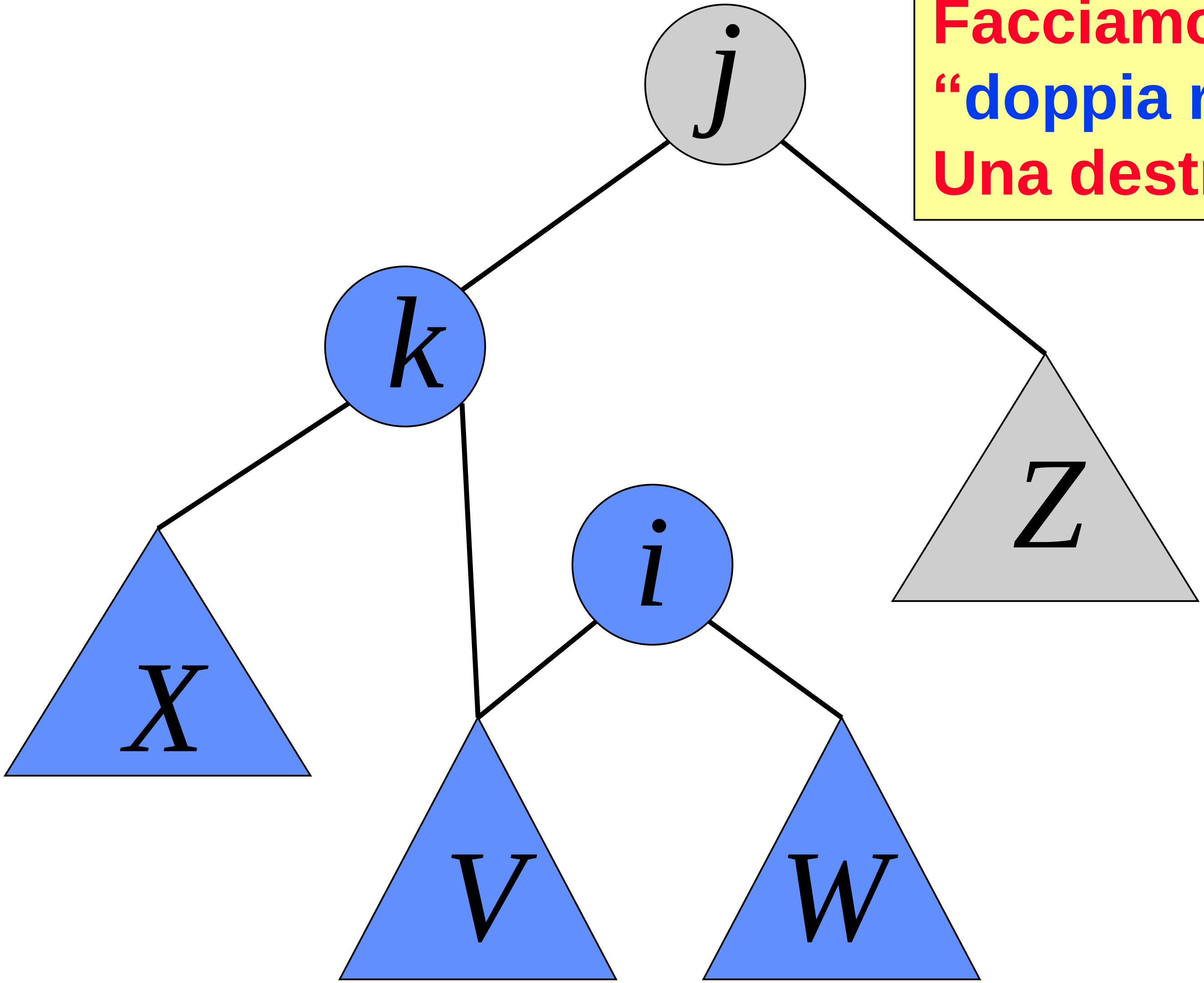


....

....

....

Rotazione doppia in alberi AVL



Facciamo ora una "doppia rotazione"
Una destra

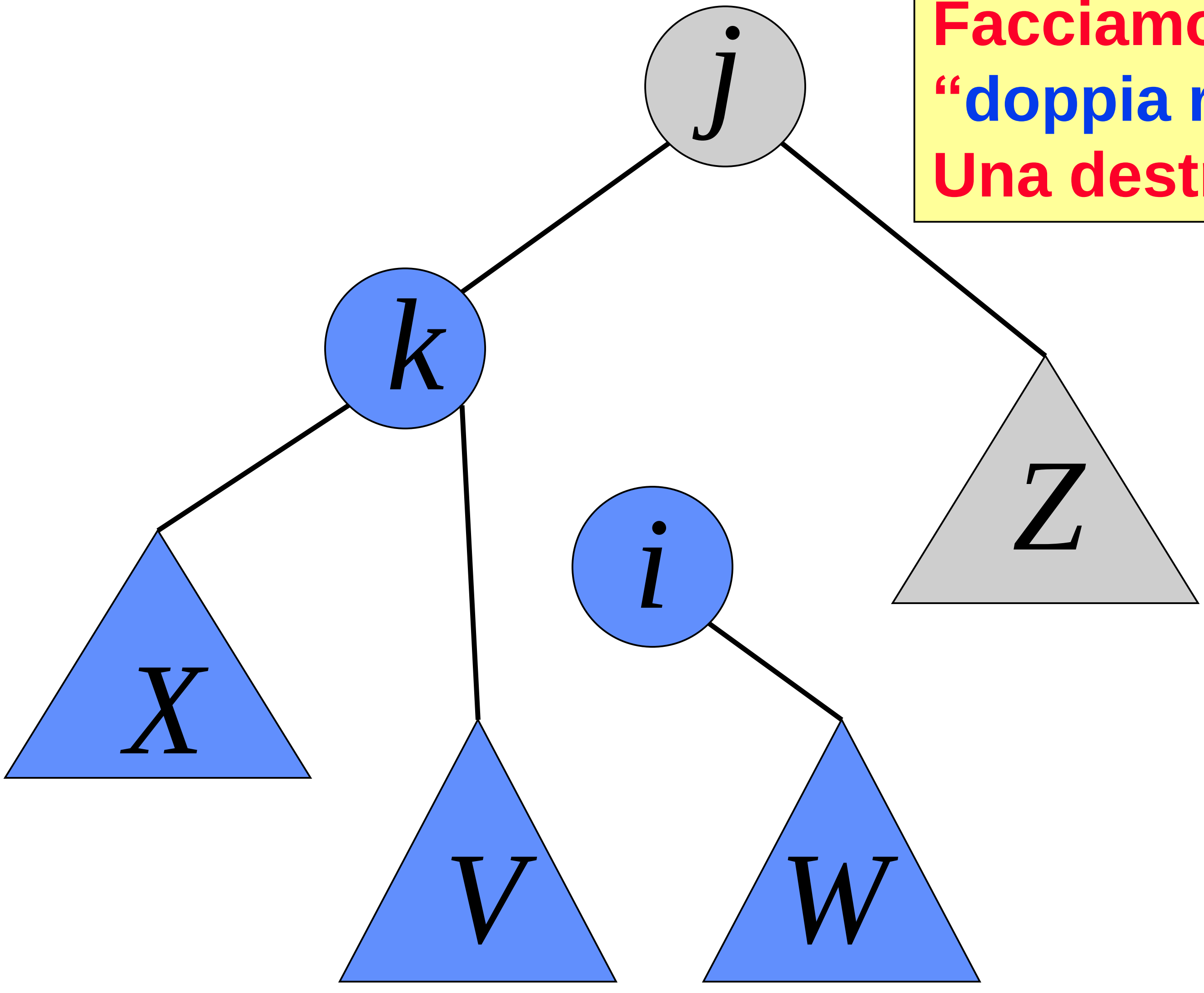
....

....

....

Rotazione doppia in alberi AVL

Facciamo ora una "doppia rotazione"
Una destra



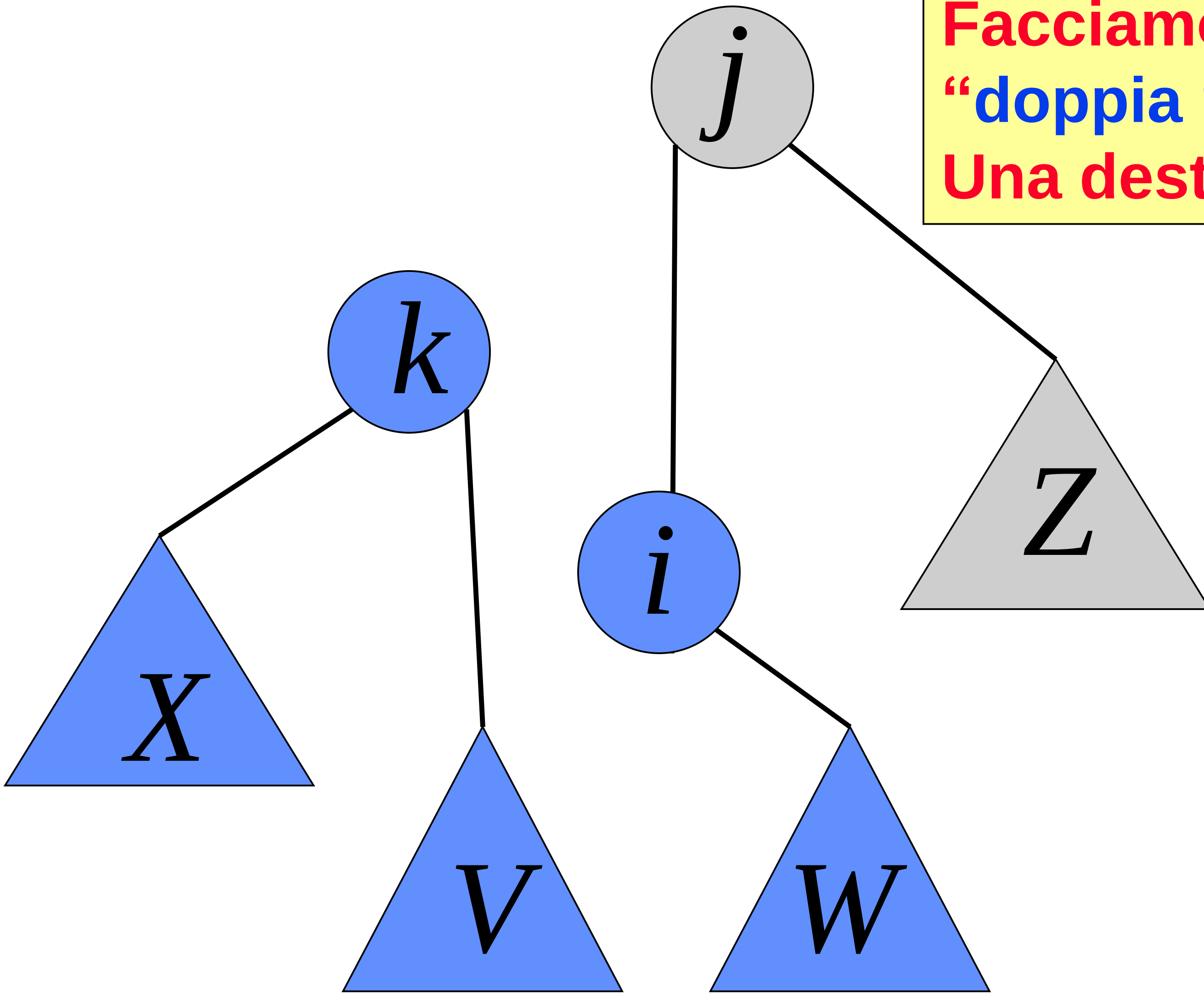
....

....

....

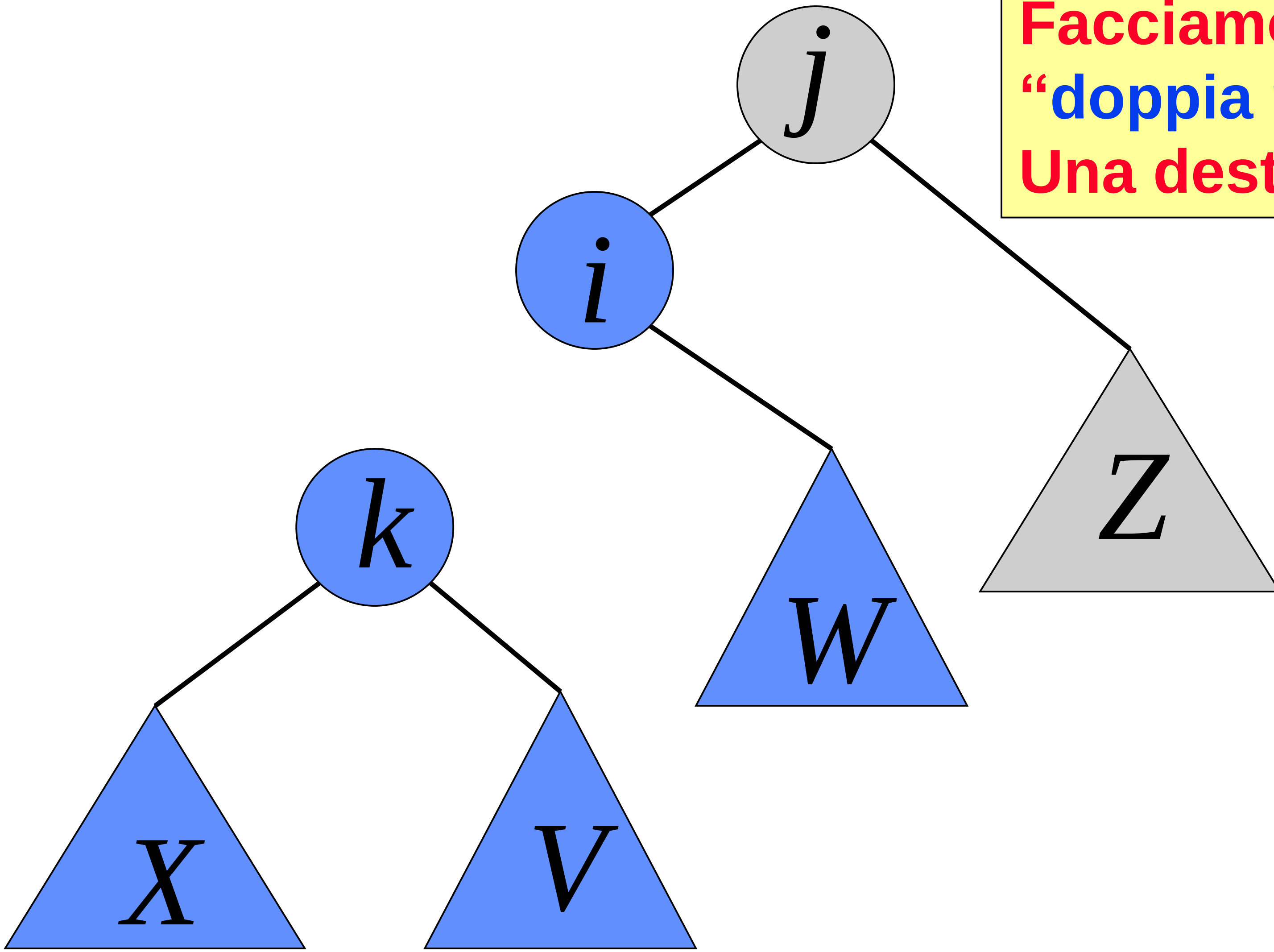
Rotazione doppia in alberi AVL

Facciamo ora una **"doppia rotazione"**
Una destra

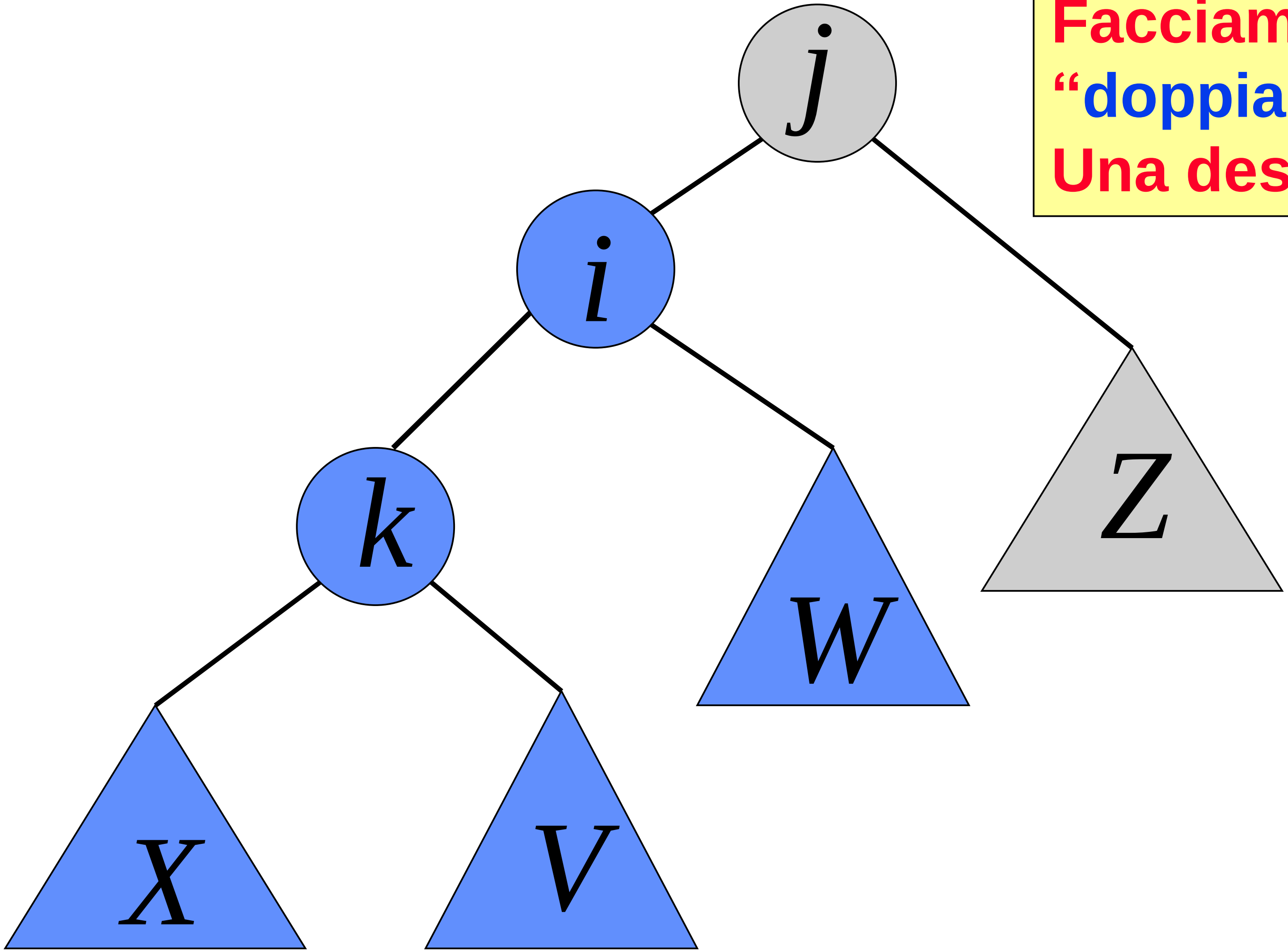


Rotazione doppia in alberi AVL

Facciamo ora una "doppia rotazione"
Una destra



Rotazione doppia in alberi AVL



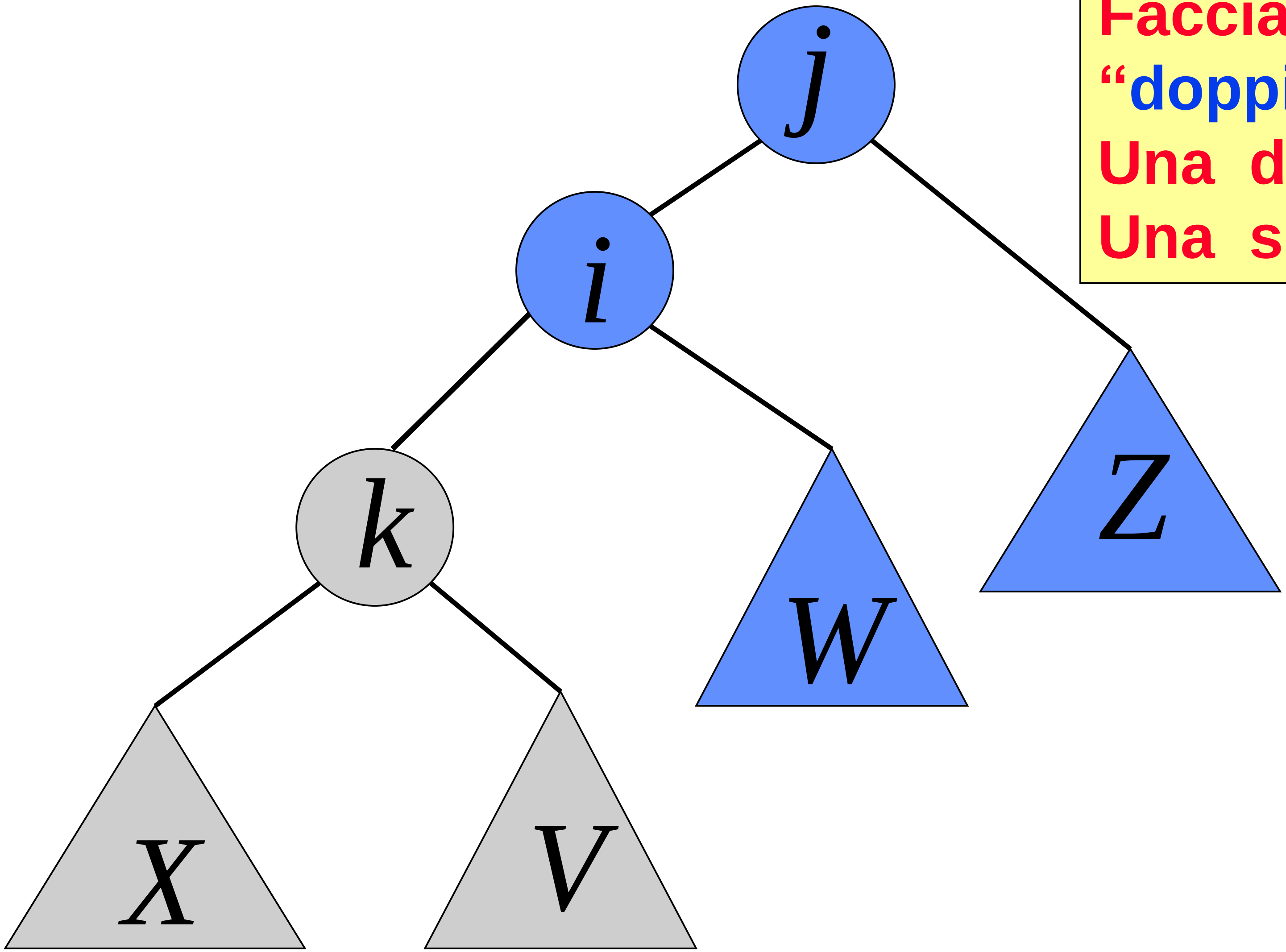
Facciamo ora una "doppia rotazione" Una destra

....

....

....

Rotazione doppia in alberi AVL



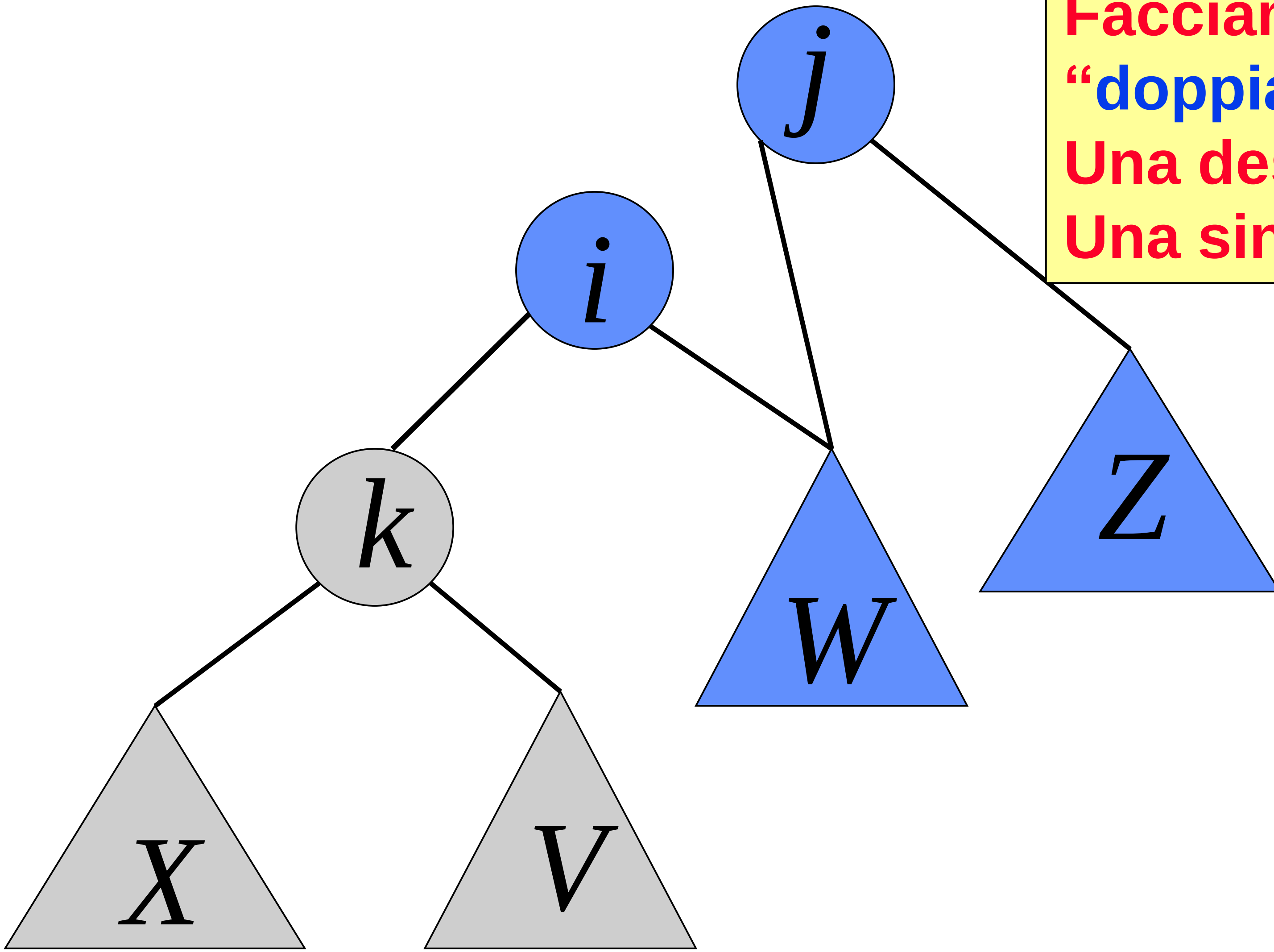
Facciamo ora una "doppia rotazione"
Una destra
Una sinistra

....

....

....

Rotazione doppia in alberi AVL



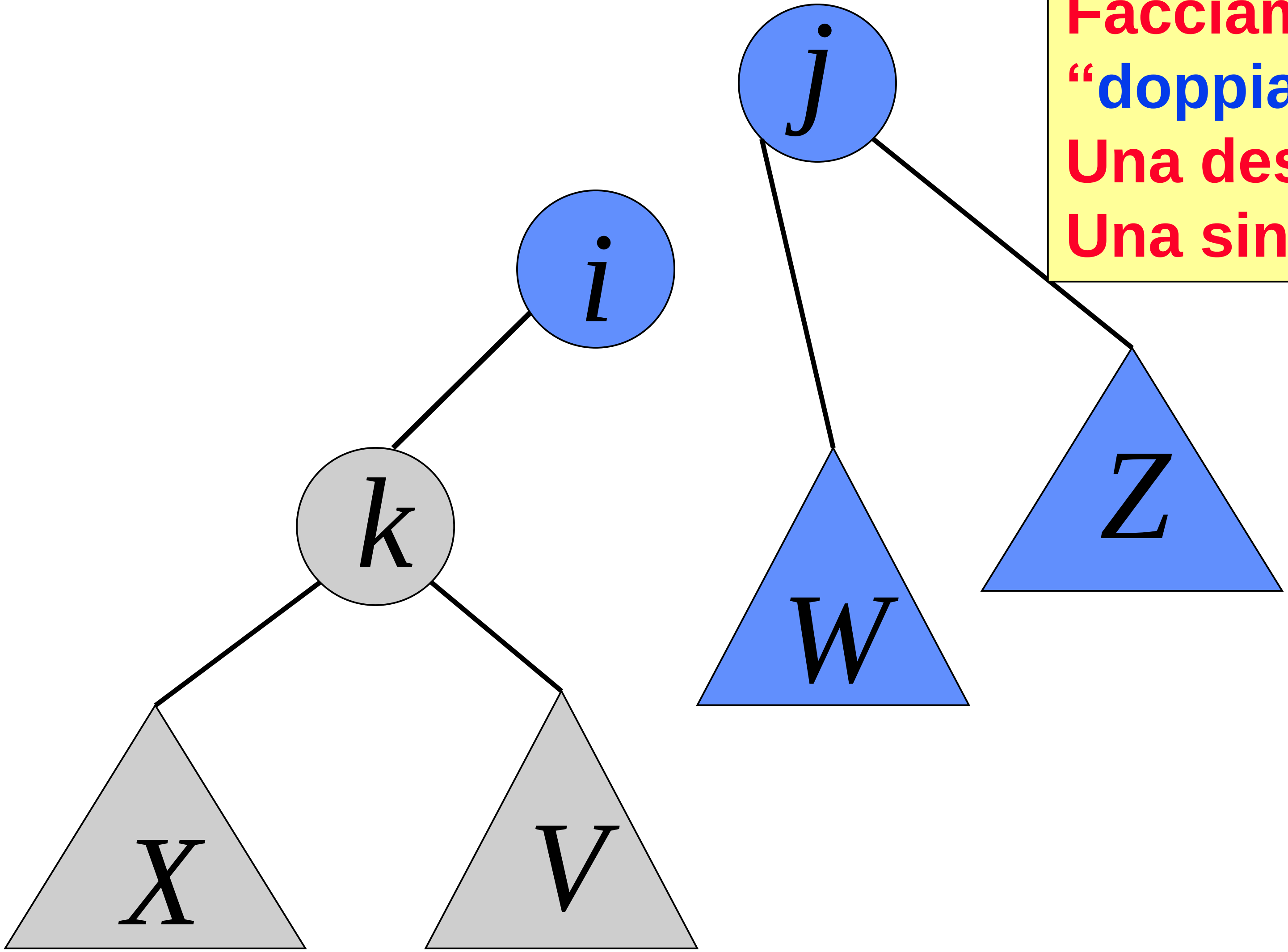
Facciamo ora una
“doppia rotazione”
Una destra
Una sinistra

....

....

....

Rotazione doppia in alberi AVL



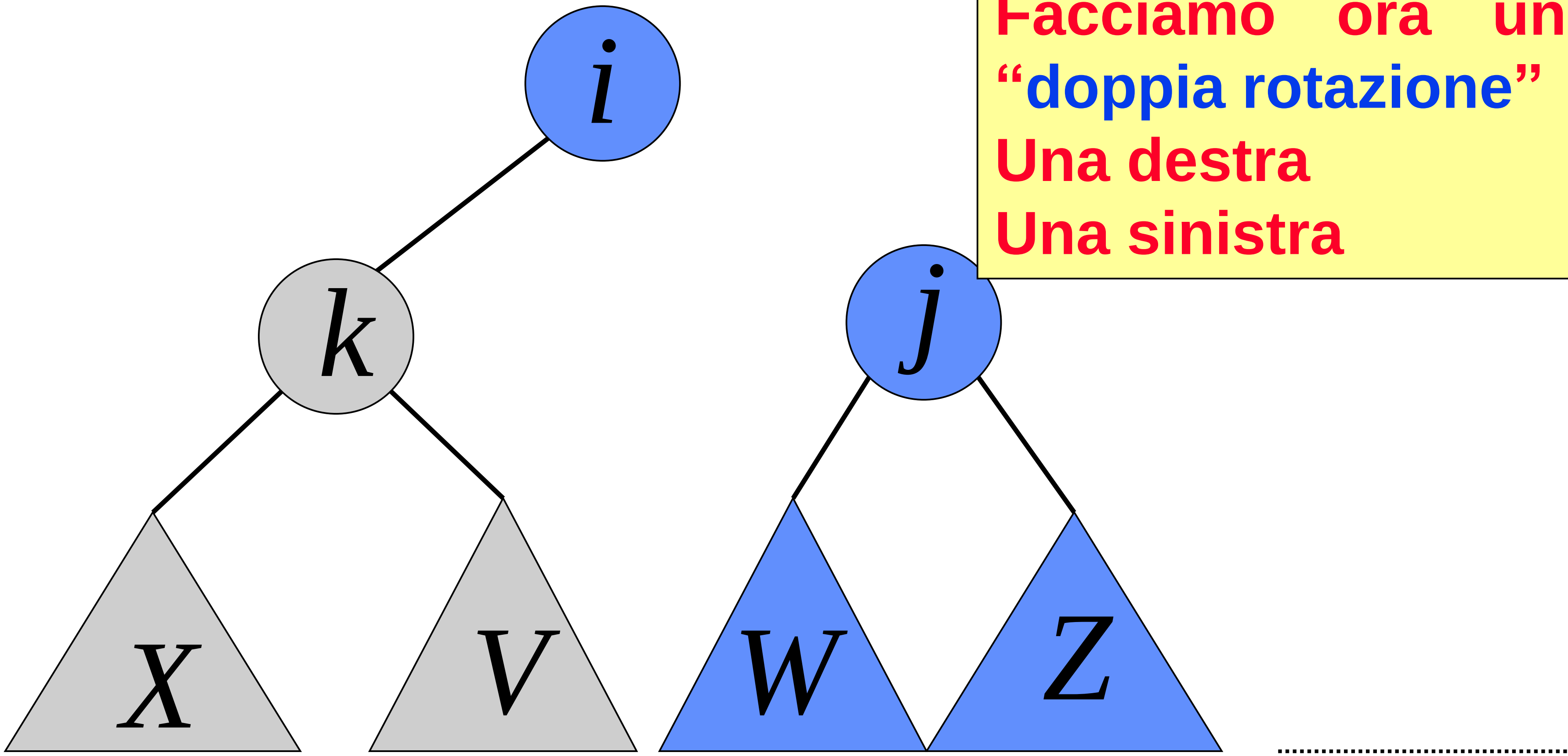
Facciamo ora una “doppia rotazione”
Una destra
Una sinistra

....

....

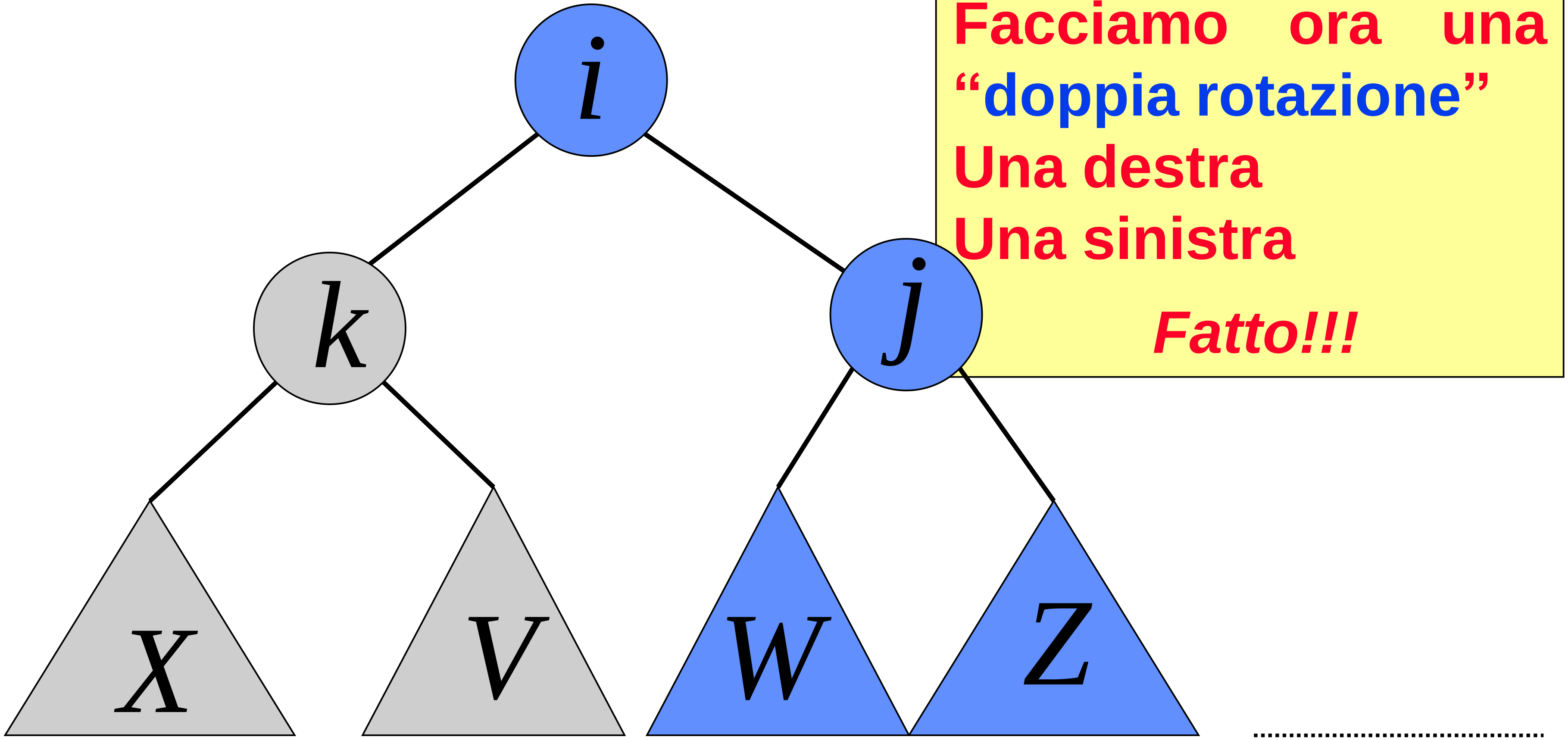
....

Rotazione doppia in alberi AVL



Facciamo ora una "doppia rotazione"
Una destra
Una sinistra

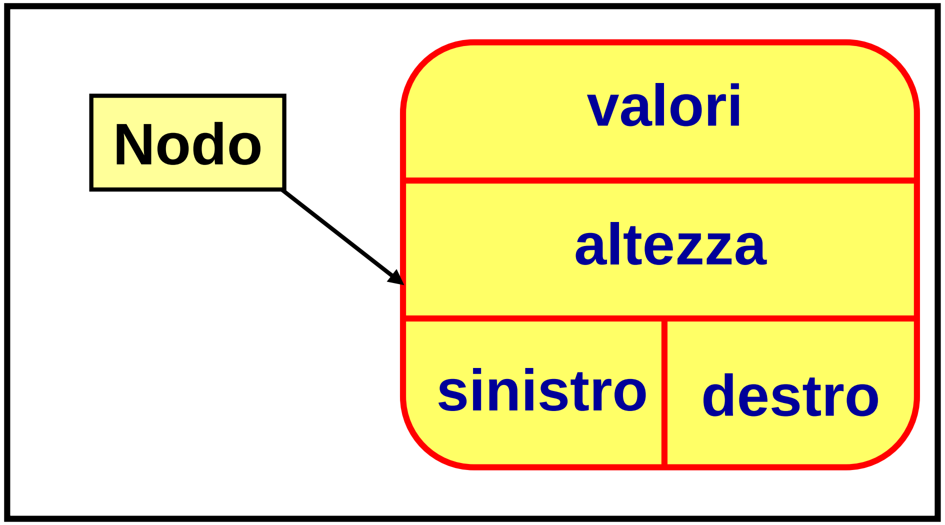
Rotazione doppia in alberi AVL



Facciamo ora una **"doppia rotazione"**
Una destra
Una sinistra

Fatto!!!

Rotazione doppia: algoritmo



Rotazione-DS*(T* : albero-AVL*)*

    T->sx* = Rotazione-SD*(T->sx)
    return* Rotazione-SS*(T)

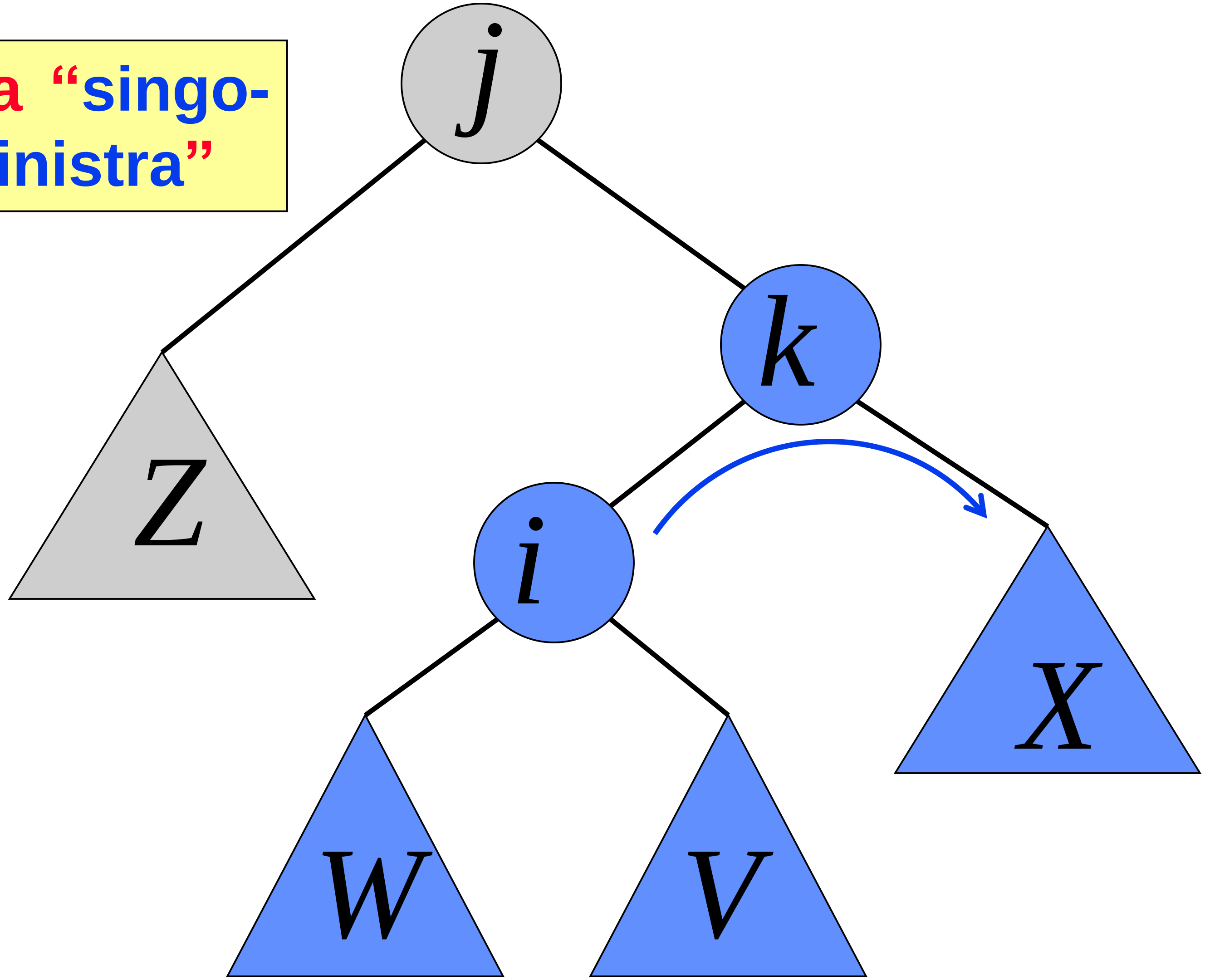
Rotazione doppia in alberi AVL

Facciamo una "singola rotazione sinistra"

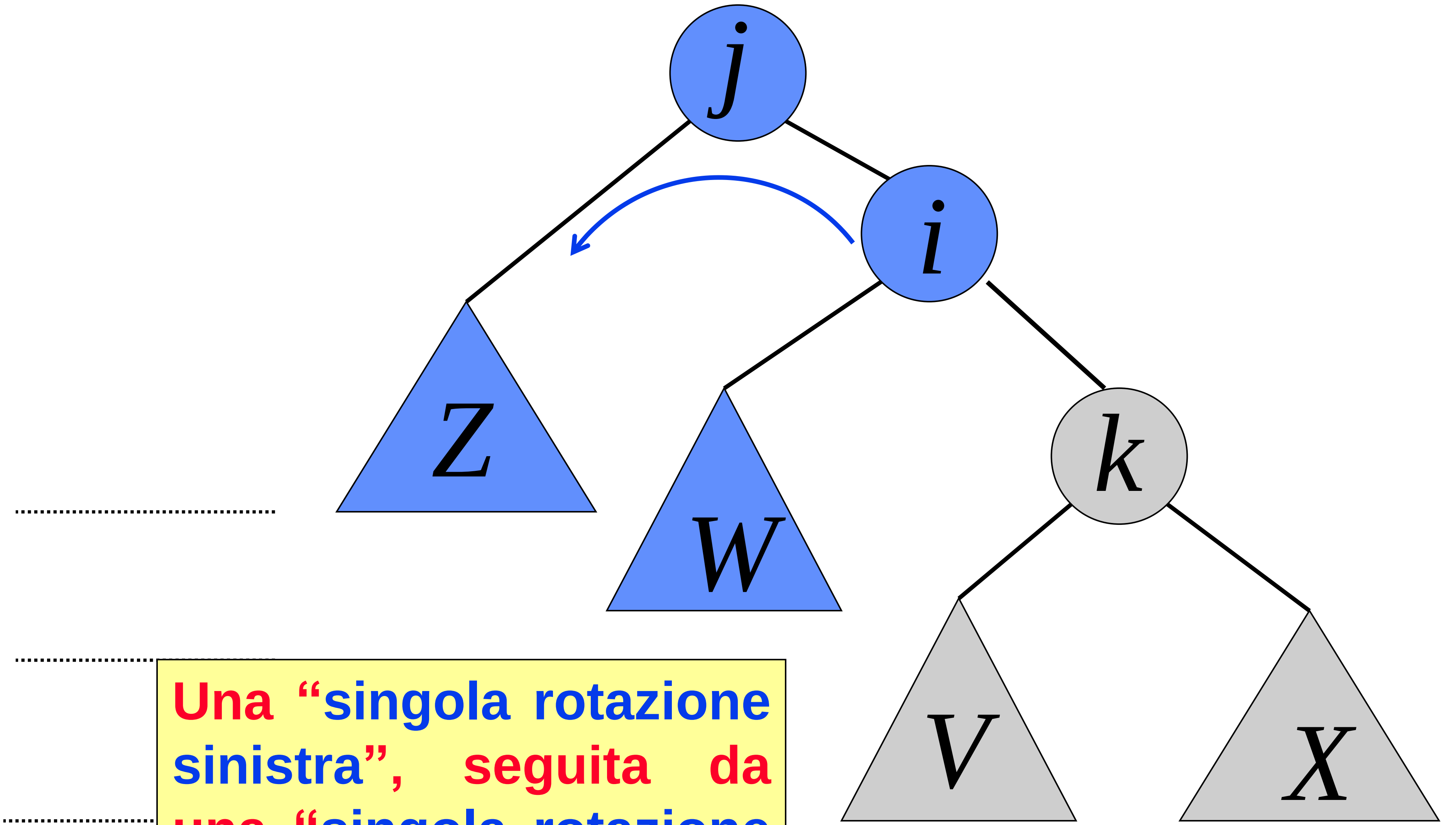
....

....

....

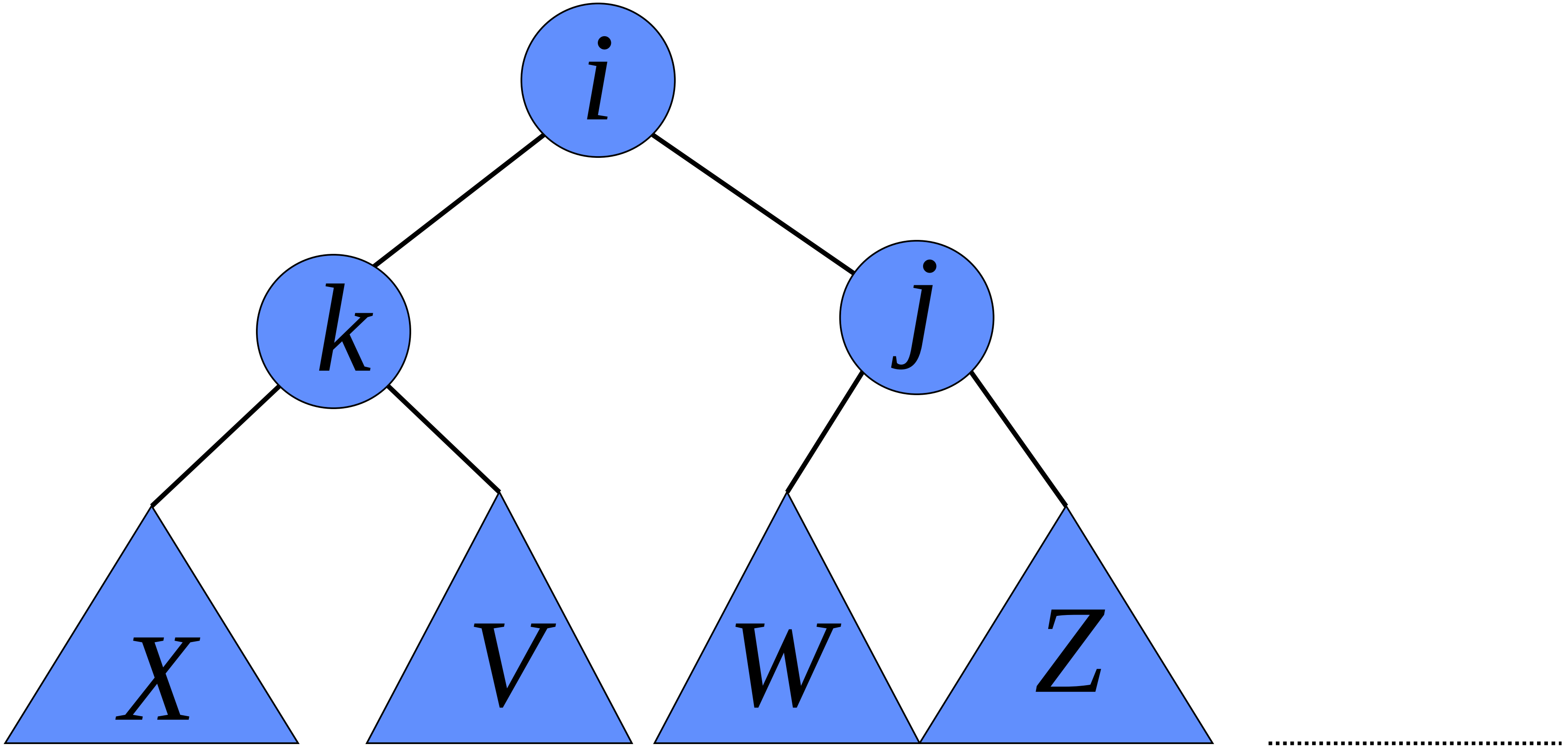


Rotazione doppia in alberi AVL



Una "singola rotazione sinistra", seguita da una "singola rotazione destra"

Rotazione doppia in alberi AVL



Rotazione doppia destra

Inserimento in alberi AVL: algoritmo

```
Inserimento-in-AVL(x : chiave, T : albero-AVL)
    IF T = NIL THEN Alloca T
                         T->altezza=0
                         T->key = x
    ELSE
      IF x < T->Key  THEN
                 T->sx = Inserimento-in-AVL(x,T->sx)
                 T = Ribilancia-Sx(T)
         ELSE
            IF x > T.Key THEN
                  T->dx = Inserimento-in-AVL(x,T->dx)
                  T = Ribilancia-Dx(T)
          ELSE {x è già presente}
    return T
```

Inserimento in alberi AVL: algoritmo

```
Ribilancia-Sx(T)
   IF Altezza(T->sx) - Altezza(T->dx) = 2 THEN
        IF is-SS(T->sx) THEN
              T = Rotazione-SS(T)
        ELSE
              T = Rotazione-DS(T)
   ELSE
              T->altezza = 1 + max(Altezza(T->sx), Altezza(T->dx))
   return T
```

```
is-SS(T : albero-AVL)
     return Altezza(T->sx) > Altezza(T->dx)
```

Inserimento in alberi AVL: algoritmo

- **Un** ***inserimento a quante operazioni di bilanciamento può dar luogo?***

Inserimento in alberi AVL: algoritmo

- Un *inserimento a quante operazioni di bilanciamento può dar luogo?*

La risposta è: al più una

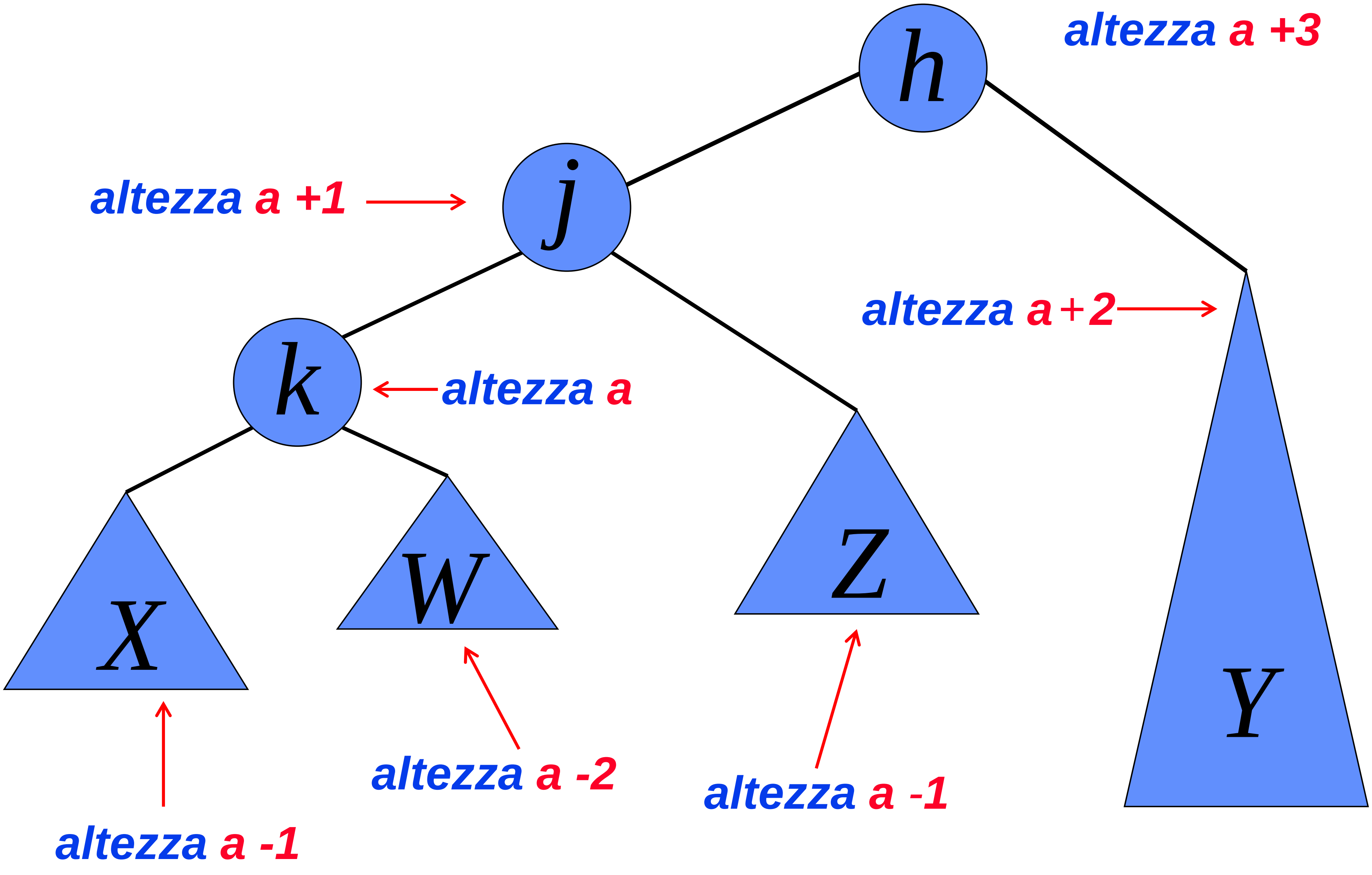
Perché?

```
Cancellazione-AVL(T:albero-AVL, key:chiave)
  IF T != NIL THEN
    IF T.Key > key THEN
        T->sx = Cancellazione-AVL(T->sx, key)
        T=Bilancia_DX(T)
    ELSE IF T->Key < key THEN
              /* cancellazione simmetrica a destra */
    ELSE IF T->sx = NIL OR T->dx = NIL THEN
           T = elimina-nodo(T)
    ELSE
         tmp = Stacca-Min-AVL(T->dx,T)
         scambia T->Key con tmp->Key
         T = Bilancia_SX(T)
         dealloca(tmp)
  return T
```

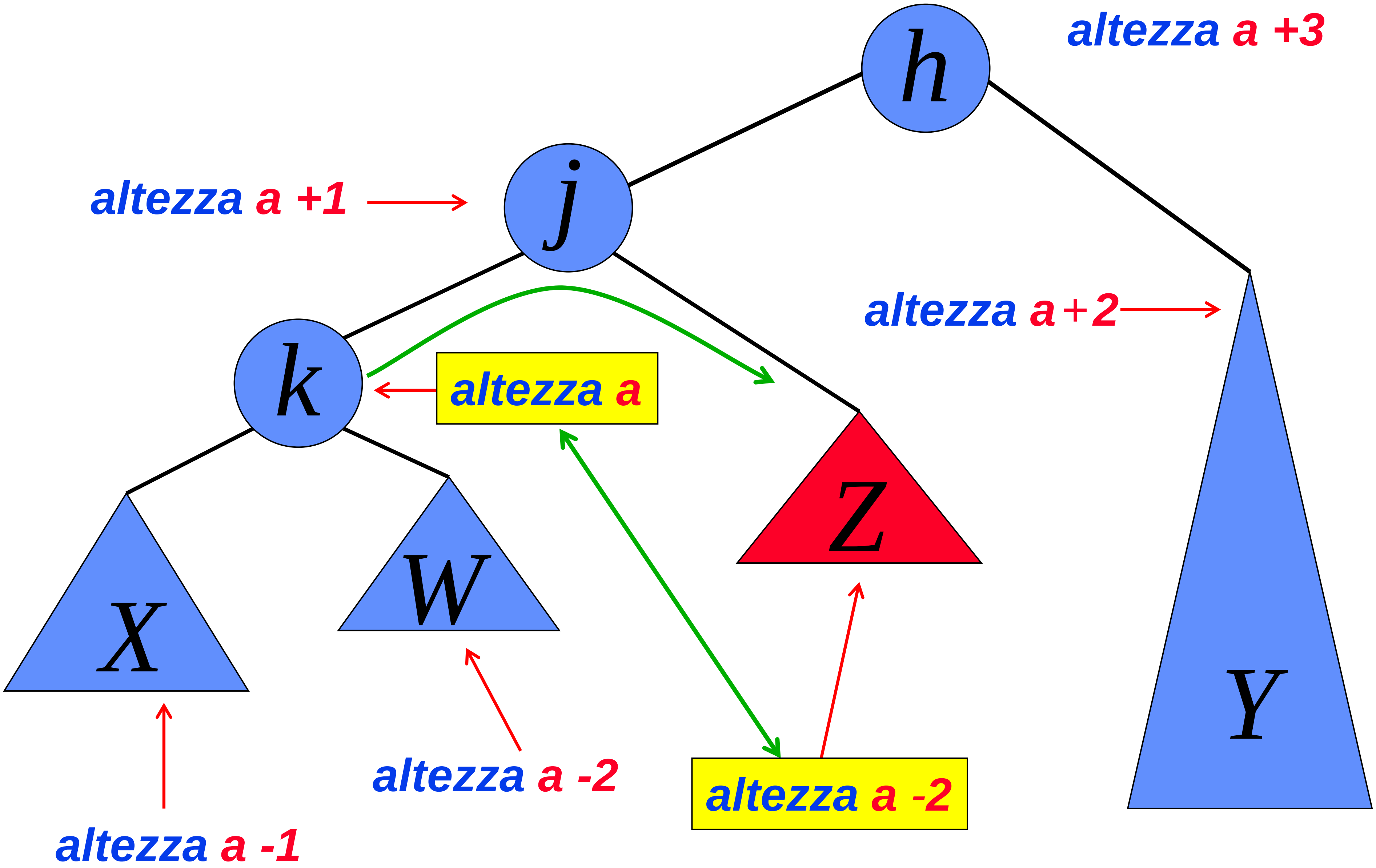
```
Stacca-Min-AVL(T:albero-AVL , P:albero-AVL)
   IF T != NIL THEN
         IF T->sx != NIL THEN
              ret = Stacca-Min-AVL(T->sx, T)
              newroot = Bilancia_DX(T)
         ELSE
              ret = T
              newroot = T->dx
         IF T = P->sx THEN  P->sx = newroot
                        ELSE  P->dx = newroot
   return ret
```

```
Bilancia_DX(T)
  IF Altezza(T->dx) - Altezza(T->sx) = 2 THEN
     IF is-SD(T->dx) THEN  T = Rotazione-SD(T)
                     ELSE  T = Rotazione-DD(T)
  ELSE
       T->altezza = 1 + max(Altezza(T->sx), Altezza(T->dx))
  return T
```

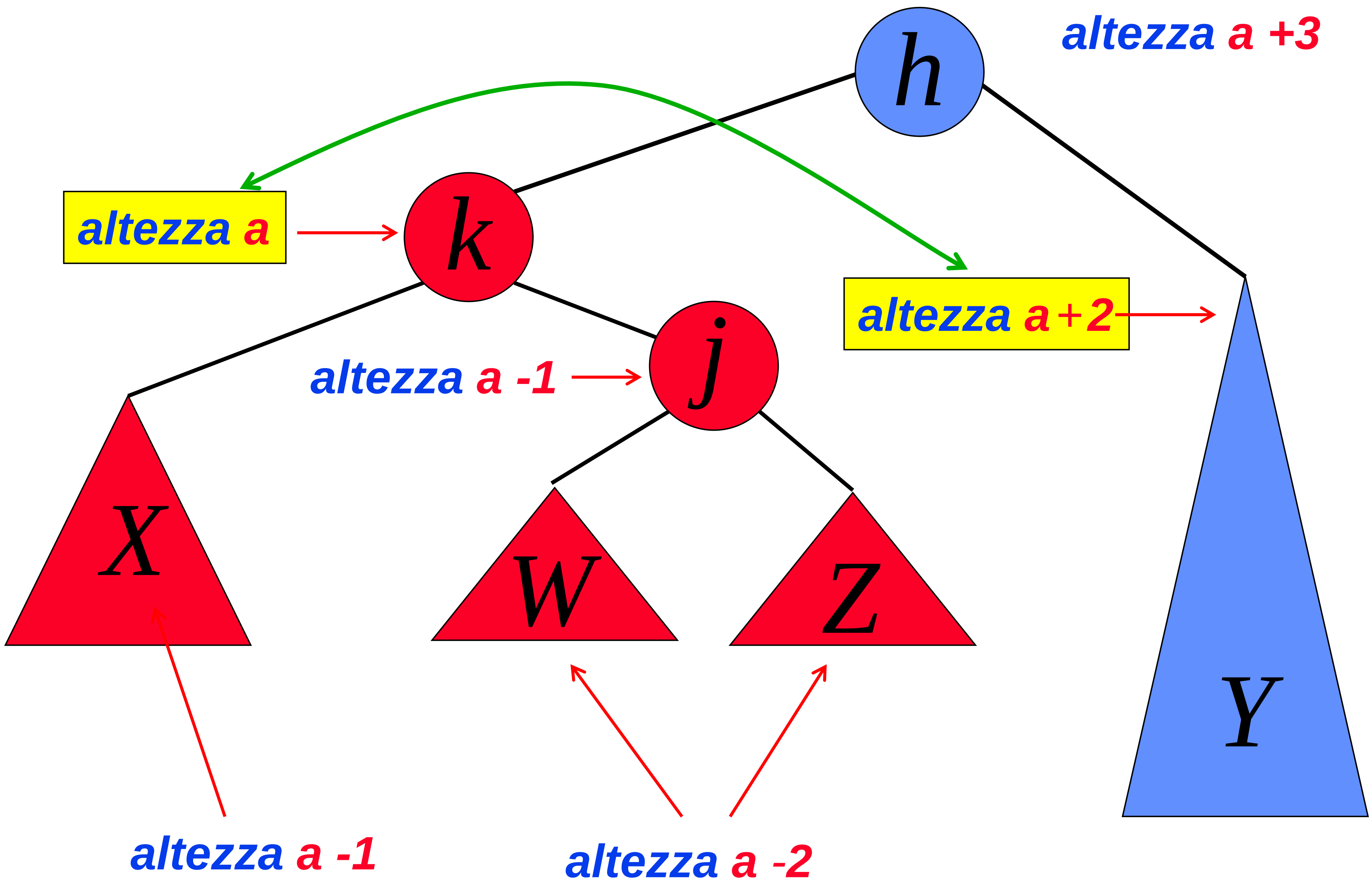
Cancellazione in alberi AVL



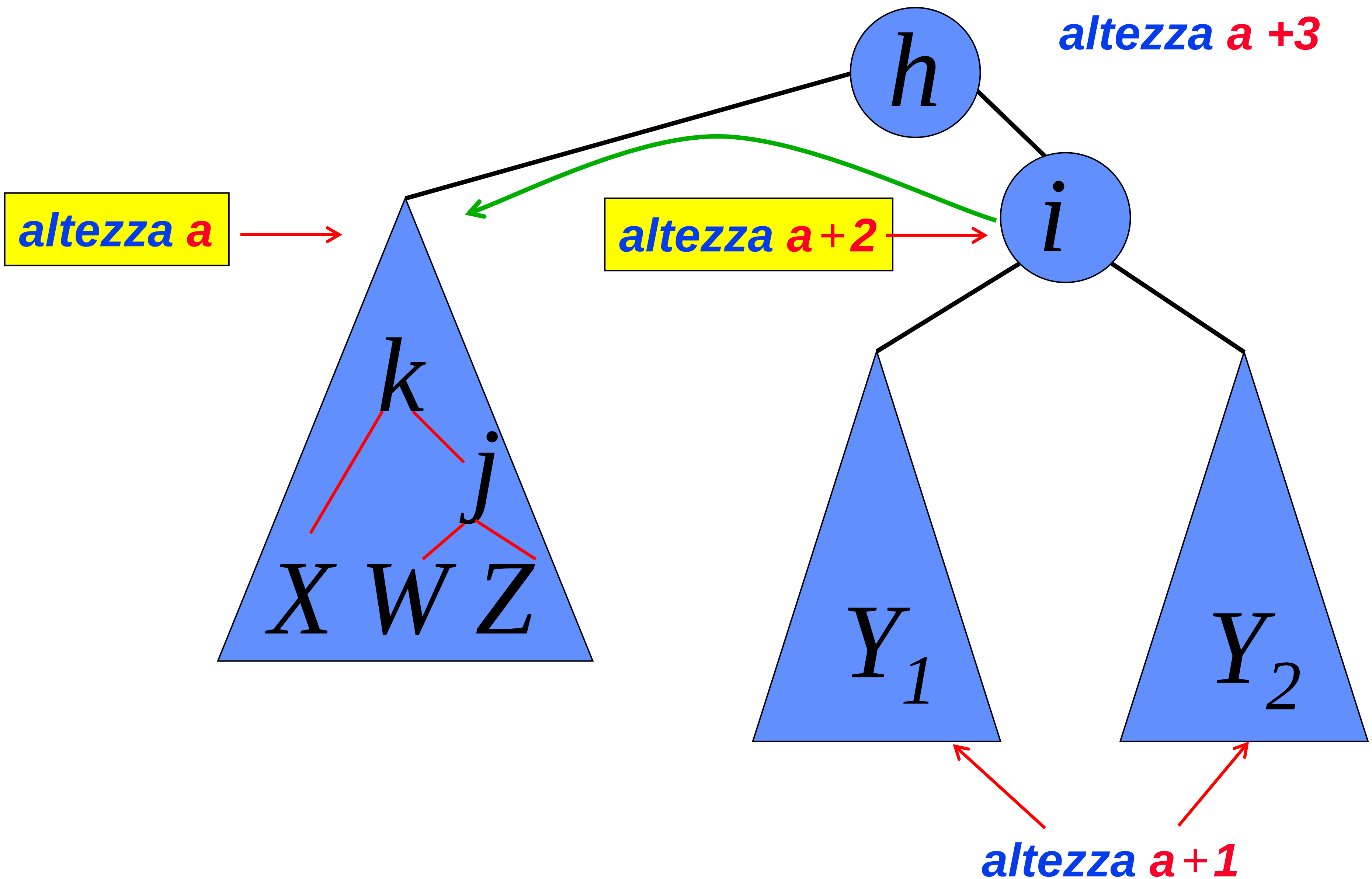
Cancellazione in alberi AVL



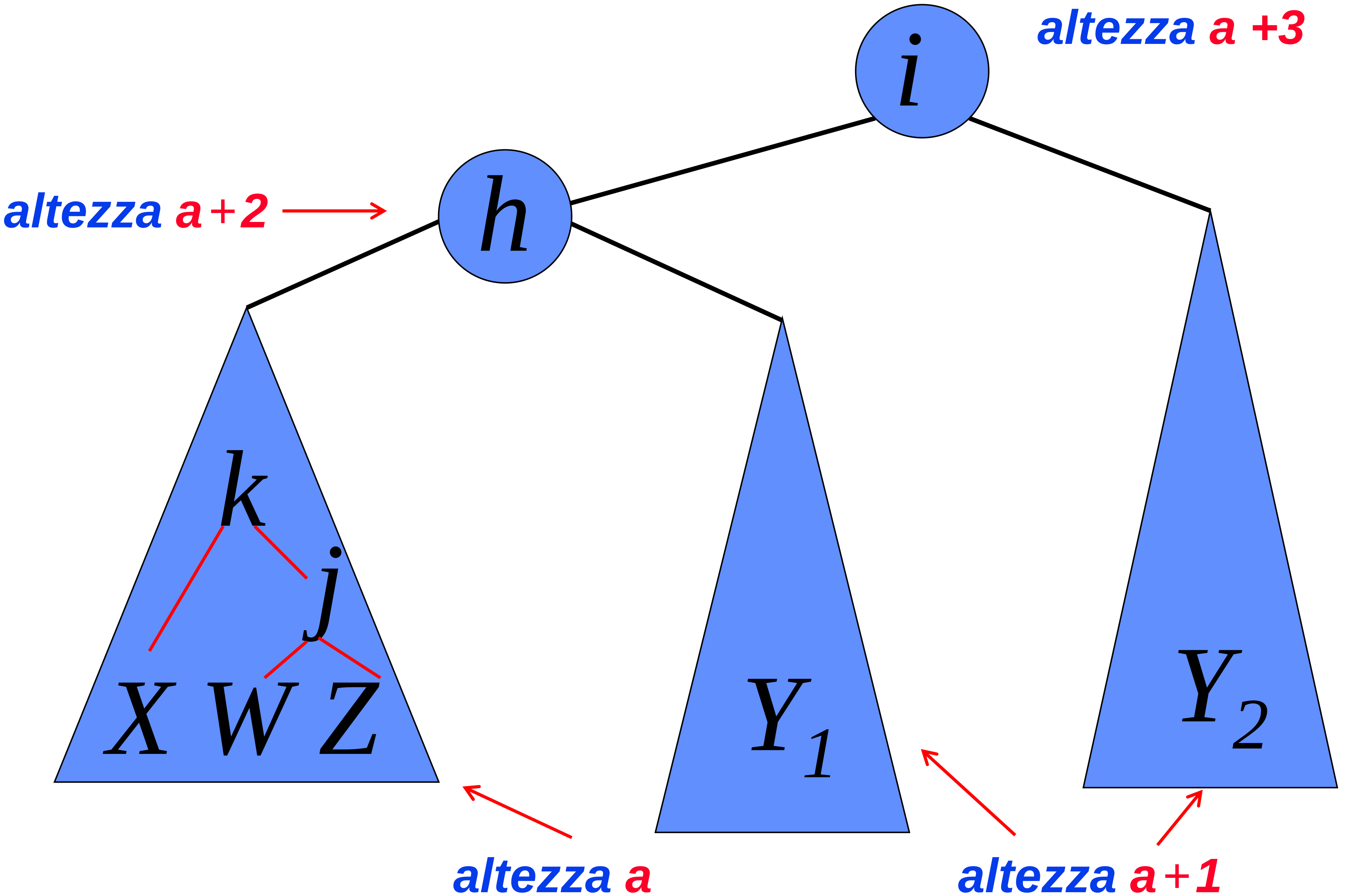
Cancellazione in alberi AVL



Cancellazione in alberi AVL



Cancellazione in alberi AVL



Cancellazione in alberi AVL

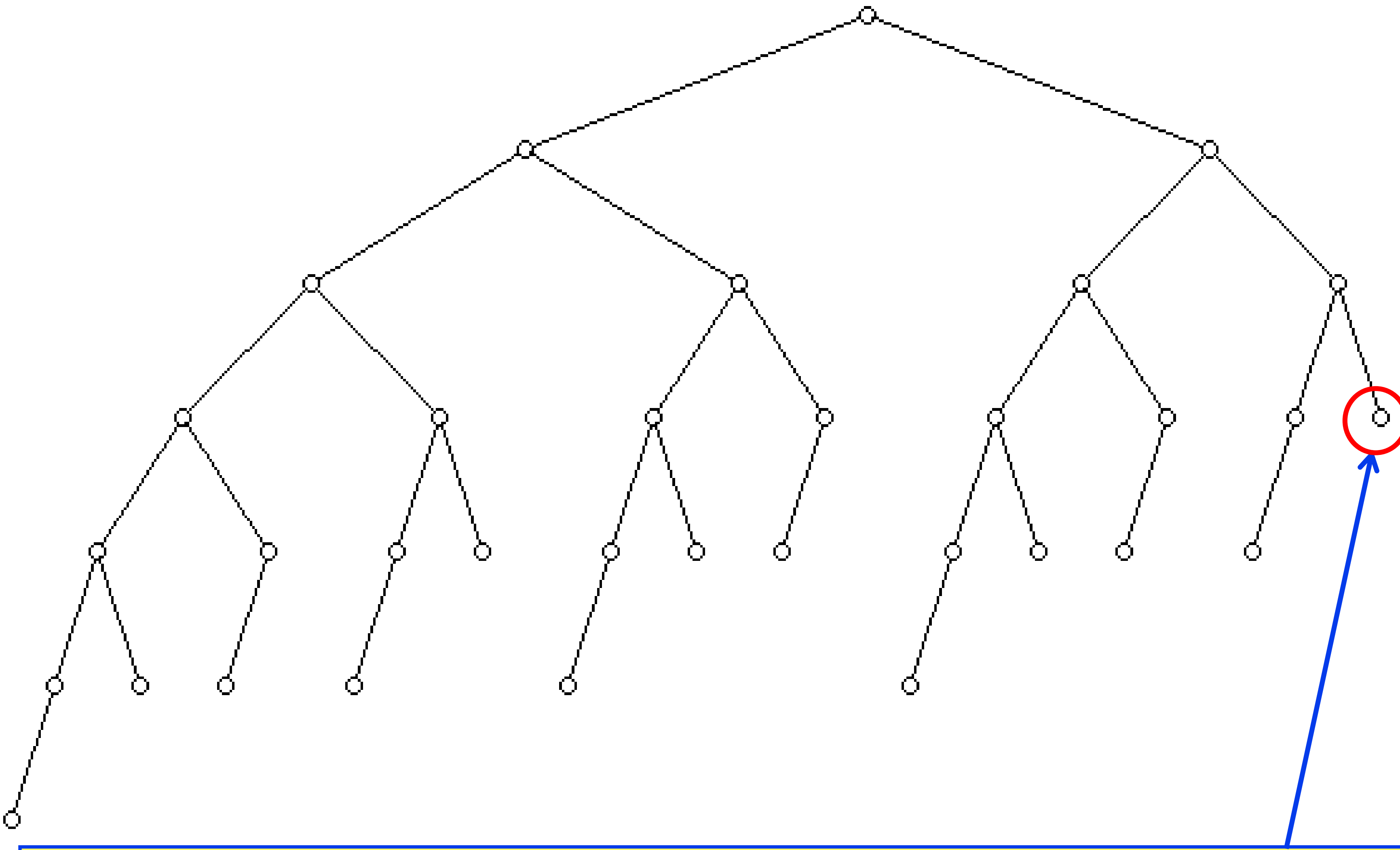
- **Dall'esempio appena visto si conclude che *una cancellazione può dar luogo a più operazioni di bilanciamento***

- ***L'algoritmo di cancellazione* è più complesso perché deve tener conto di questa possibilità**

- **Può *anche* verificarsi la necessità di *una operazione di bilanciamento per ogni livello* (*in quale caso?*)**

- **Il numero di operazioni bilanciamento è nel caso peggiore *O(h)* (h *altezza dell'albero*)**

Cancellazione in alberi AVL



Necessità di una operazione di bilanciamento per ogni livello se si cancella l'elemento più a destra!