

SPLAY TREES

• GLI SPLAY TREES IMPLEMENTANO M OPERAZIONI
CONSECUTIVE DI

- RICERCA
- INSERIMENTO
- CANCELLAZIONE

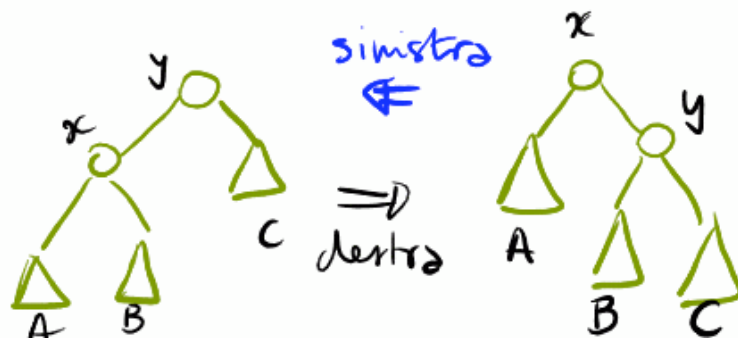
IN TEMPO $O(M \log N)$, DOVE N È IL NUMERO
DI INSERIMENTI

• IN ALTRE PAROLE CIASCUNA DELLE M OPERAZIONI
VIENE ESEGUITA IN TEMPO AMMORTIZZATO $O(\log N)$

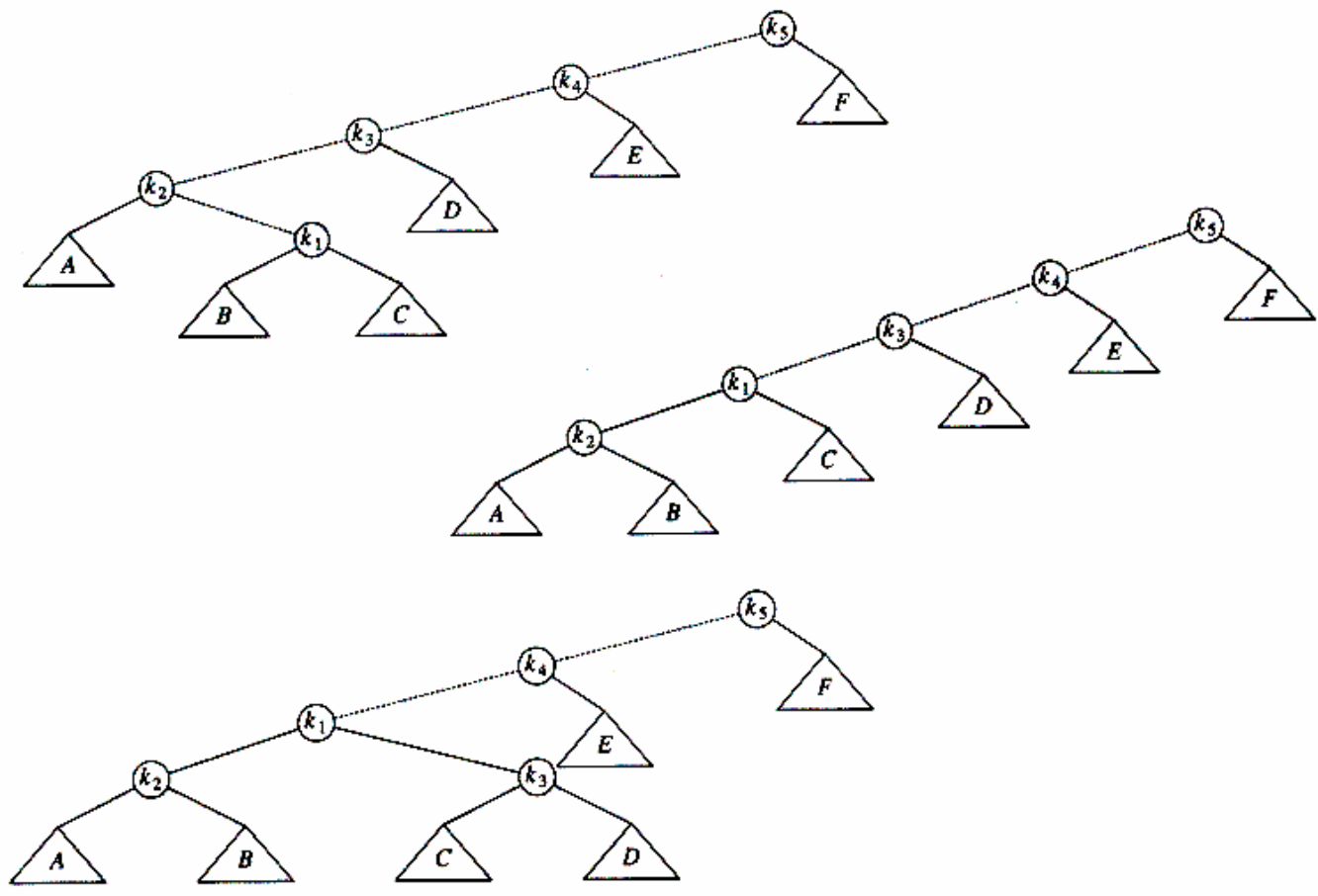
IDEA: PER QUANTO POSSIBILE AVVICINARE ALLA RADICE I NODI CHE SI INCONTRANO DURANTE UN ACCESSO

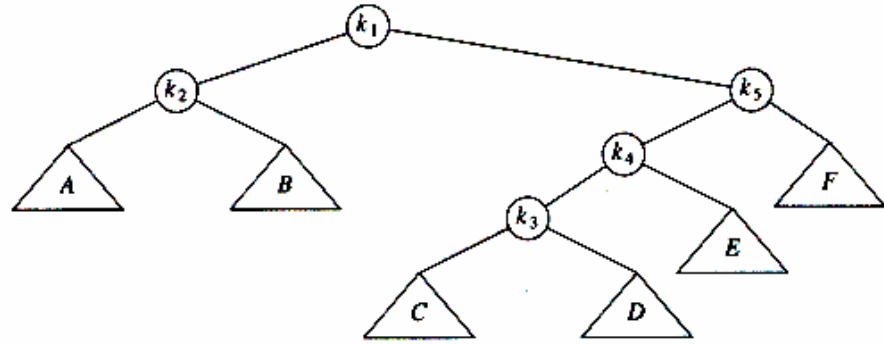
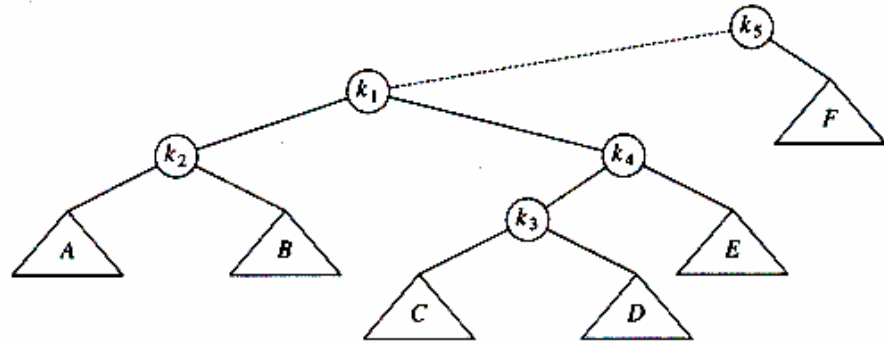
UNA SEMPLICE IDEA (CHE NON FUNZIONA)

EFFETTUARE TUTTE LE POSSIBILI ROTAZIONI, DAL BASSO VERSO L'ALTO, LUNGO IL CAMMINO DI ACCESSO



ESEMPIO (ACCESSO AL NODO k_1)





ESEMPIO

SI CONSIDERI LA SEQUENZA DELLE SEGUENTI OPERAZIONI:

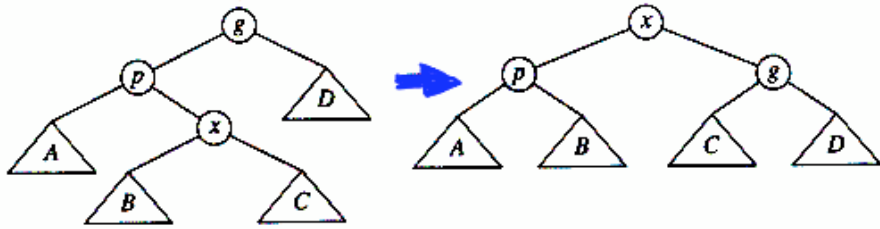
INSERT (1)
INSERT (2)
⋮
INSERT (N)

} $\Theta(N)$

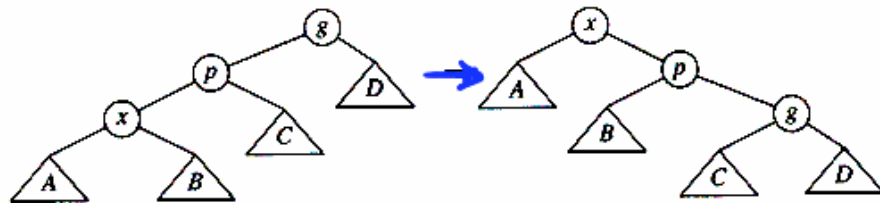
SEARCH (1)	N-1	ROTAZ.
SEARCH (2)	N-1	"
SEARCH (3)	N-2	"
SEARCH (4)	N-3	"
⋮		
SEARCH (N)	1	"

} $\Theta(N^2)$

SPLAY TREES



ZIG-ZAG



ZIG-ZIG



ZIG (x NON HA NONNO)

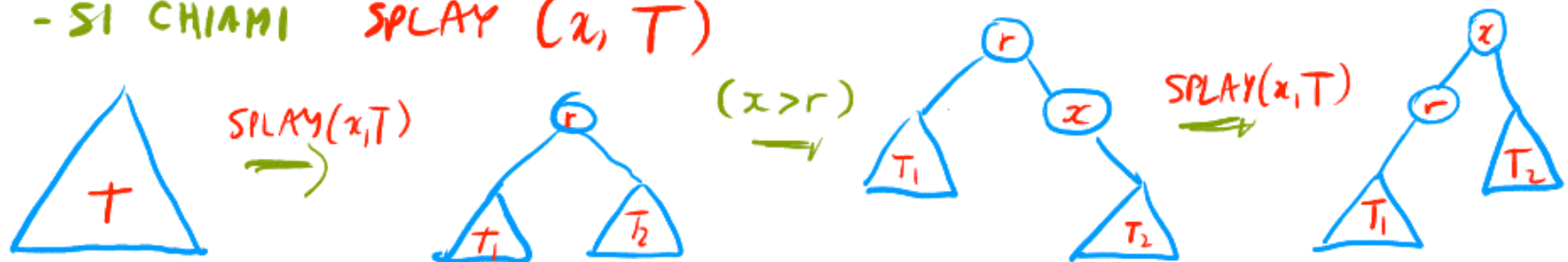
+ SIMMETRICHE

SPLAY(x, T)

- SI CERCHI x SU T MEDIANTE RICERCA BINARIA
- A PARTIRE DAL NODO OVE LA RICERCA SI E' FERMATA E PROCEDENDO VERSO LA RADICE, SI APPLICHIAMO LE ROTAZIONI DI TIPO ZIG, ZIG-ZAG O ZIG-ZIG NECESSARIE

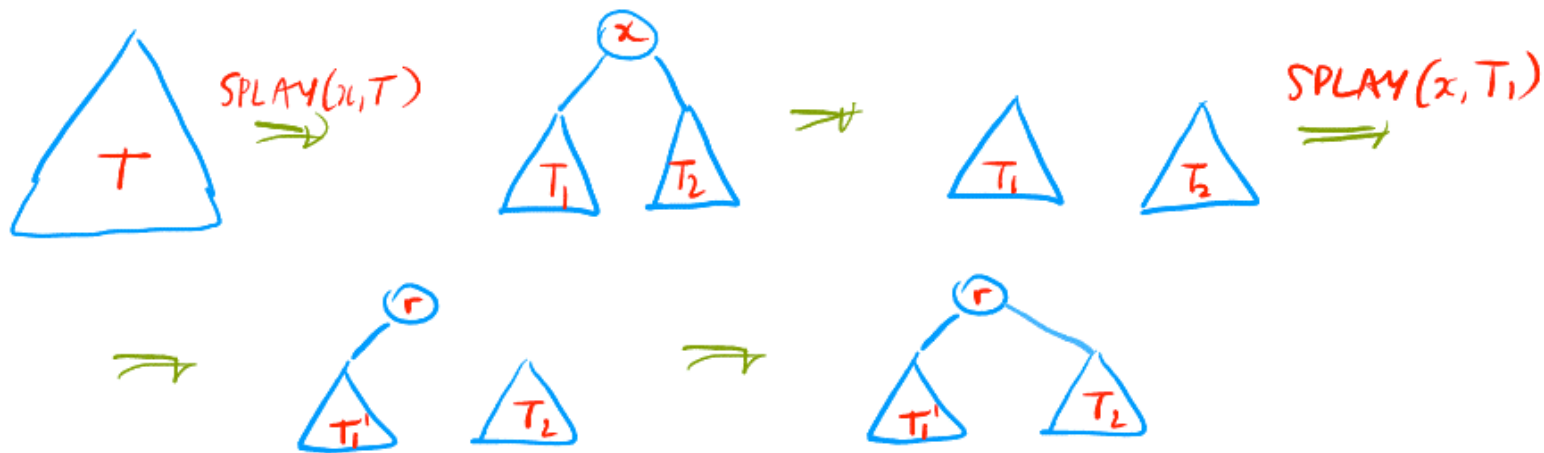
INSERT(x, T)

- SI CHIAMO SPLAY(x, T)
- SI INSERISCA IL NUOVO NODO x
- SI CHIAMO SPLAY(x, T)



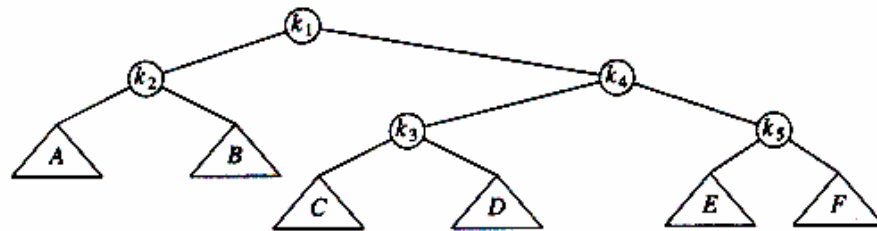
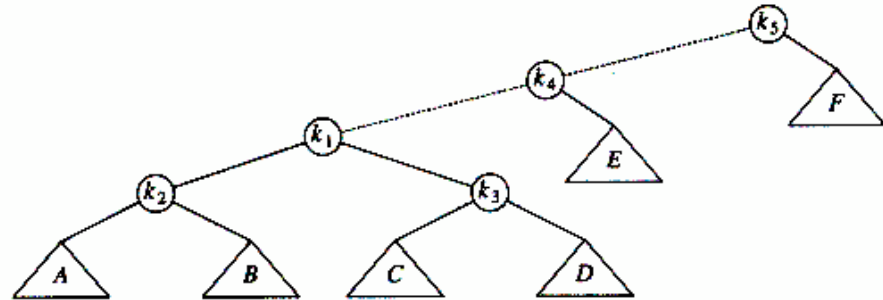
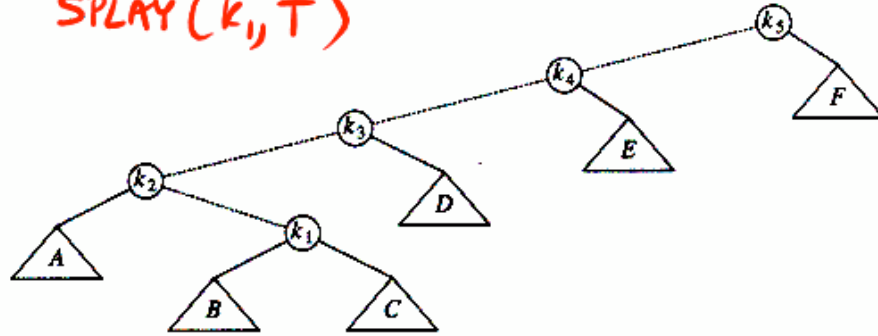
DELETE(x, T)

- SPLAY(x, T)
- SI CANCELLI IL NODO x OTTENENDO DUE ALBERI T_1 E T_2
- SPLAY(x, T_1); SIA r LA NUOVA RADICE
- SI PONGA LA RADICE DI T_2 COME FIGLIO DESTRO DEL NODO r



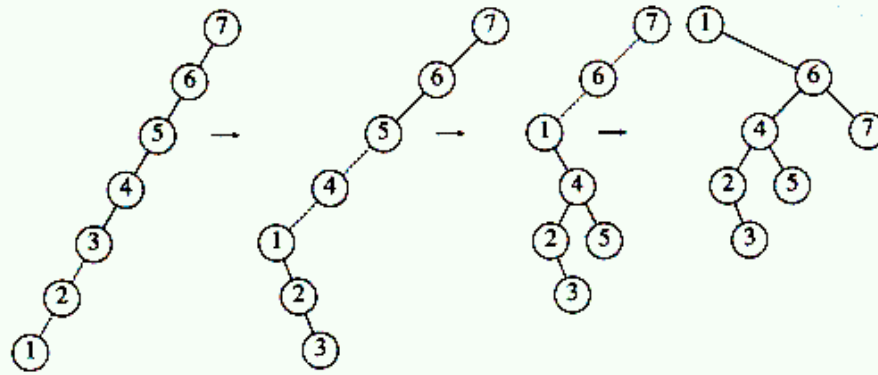
ESEMPIO

SPLAY(k_1, T)

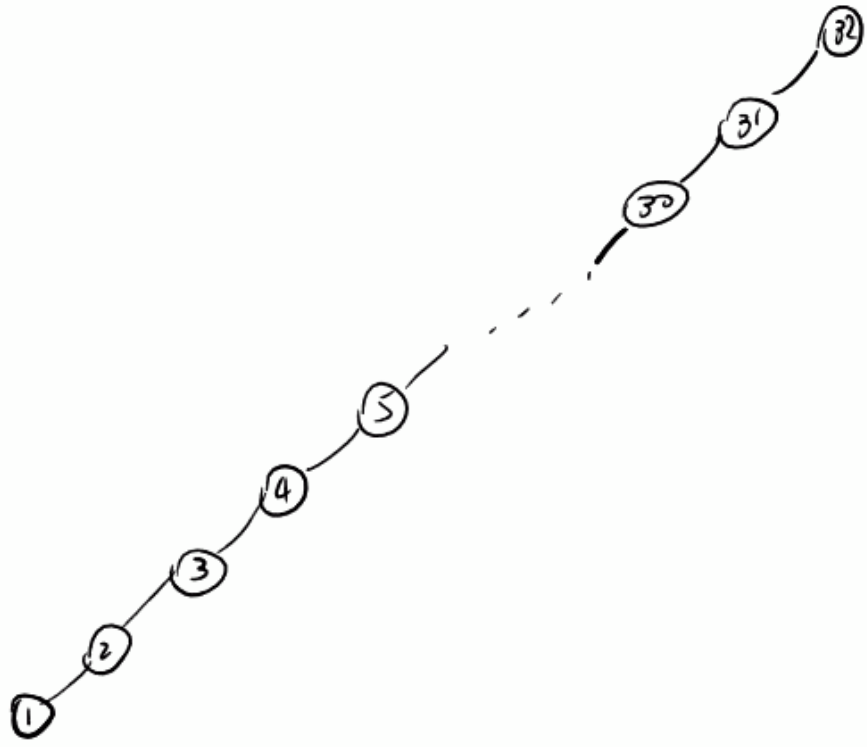


ESEMPLO

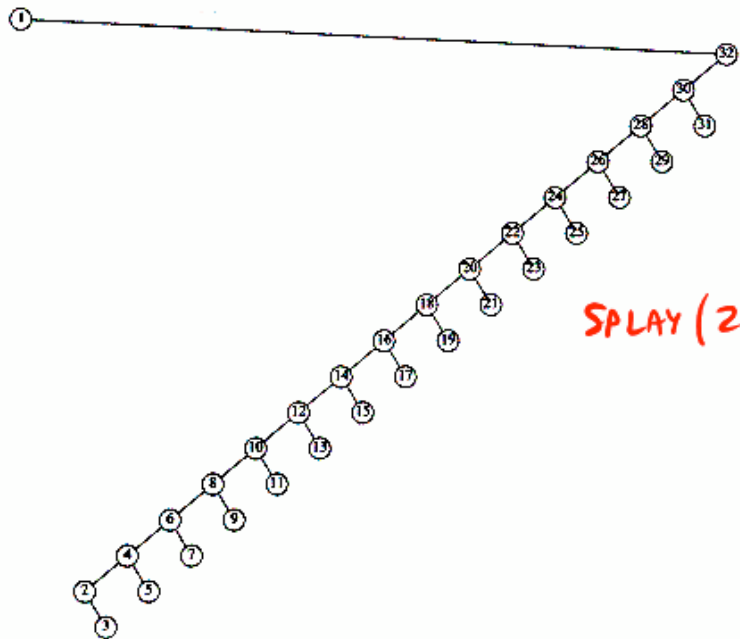
SPLAY (1, T)



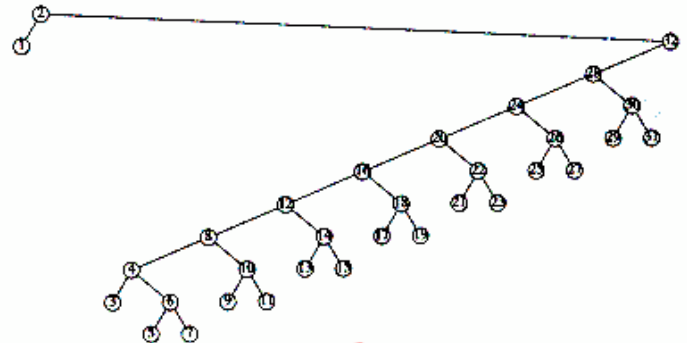
ESEMPPIO



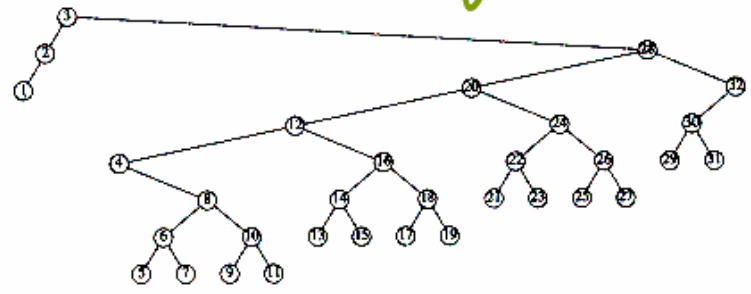
SPLAY(i, T) ⇒



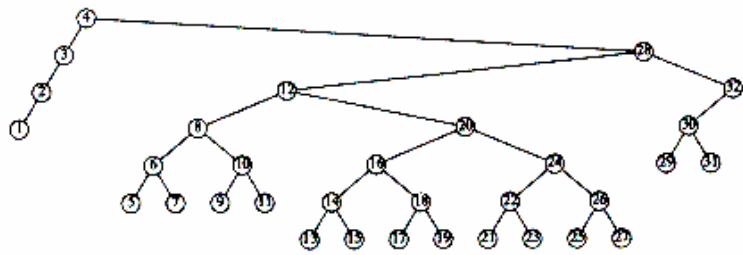
SPLAY(2,T) →



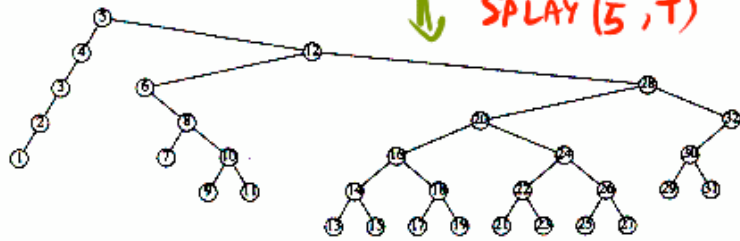
SPLAY(3,T)



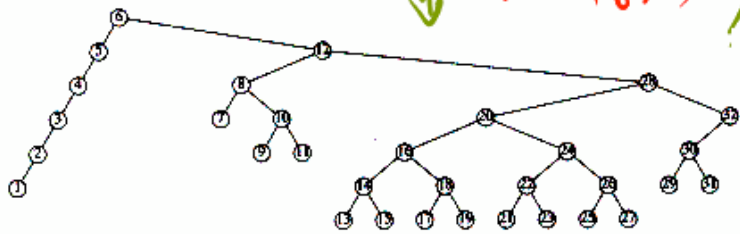
SPLAY(4,T) →



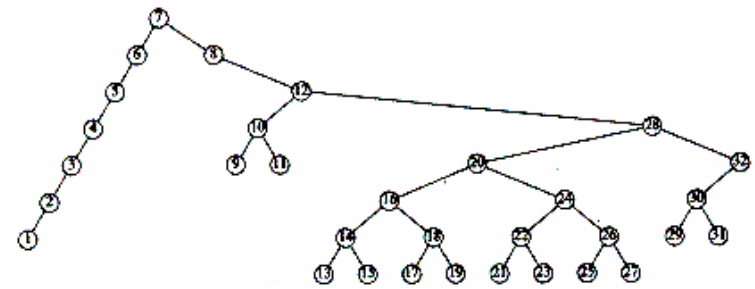
↓ SPLAY(5, T)



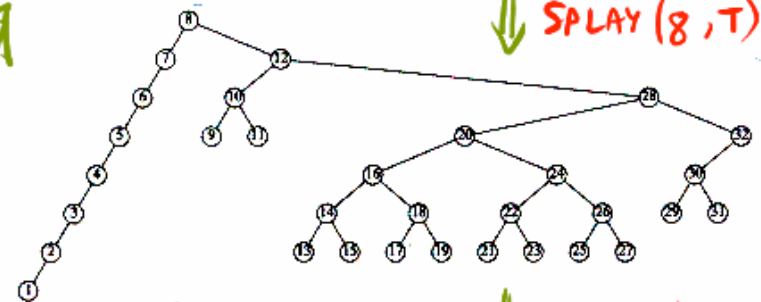
↓ SPLAY(6, T)



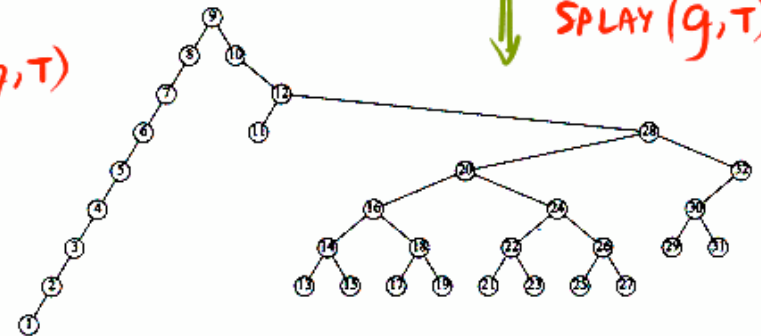
↗ 7
SPLAY(7, T)



↓ SPLAY(8, T)



↓ SPLAY(9, T)



ANALISI AMMORTIZZATA DELL'OPERAZIONE SPLAY

COSTI DELLE OPERAZIONI ELEMENTARI

ZIG	1	ROTAZIONE
ZIG-ZAG	2	ROTAZIONI
ZIG-ZIG	2	ROTAZIONI

PONIAMO:

$S(v) =_{\text{def}} \# \text{ NODI DEL SOTTOALBERO CON RADICE } v$

$R(v) =_{\text{def}} \lg S(v)$ (rango di v)

$\Phi(T) =_{\text{def}} \sum_{v \in T} R(v)$

- SIA T_0 UN ALBERO VUOTO, SI HA

$$\Phi(T_0) = \sum_{v \in T_0} R(v) = 0$$

- SIA T UN ALBERO QUALSIASI

$$\Phi(T) = \sum_{v \in T} R(v) = \sum_{v \in T} l_p S(v) \geq 0 = \Phi(T_0)$$

- PERTANTO POSSIAMO UTILIZZARE IL POTENZIALE

$\Phi(T)$ PER CALCOLARE UN UPPER BOUND AL COSTO

REALE DI M OPERAZIONI (SPLAY, INSERT, DELETE) DI

CUI N SONO INSERT

LEMMA

SIANO $a, b \in \mathbb{N}^+$ E SIA $c \geq a + b$.

ALLORA $\log a + \log b \leq 2 \log c - 2$.

DIM.

SI OSSERVI

$$(a-b)^2 \geq 0$$

$$a^2 - 2ab + b^2 \geq 0$$

$$a^2 + 2ab + b^2 \geq 4ab$$

$$(a+b)^2 \geq 4ab$$

$$a+b \geq 2\sqrt{ab}$$

$$c \geq 2\sqrt{ab} \quad (\text{POICHÉ } c \geq a+b)$$

$$\log c \geq 1 + \frac{1}{2}(\log a + \log b)$$

$$2 \log c - 2 \geq \log a + \log b. \quad \blacksquare$$

(PRENDENDO I LOGARITMI
DI AMBO I MEMBRI)

TEOREMA IL COSTO AMMORTIZZATO DELL'OPERAZIONE $SPLAY(x, T)$
È AL PIÙ $3(R(\text{root}(T)) - R(x)) + 1$.

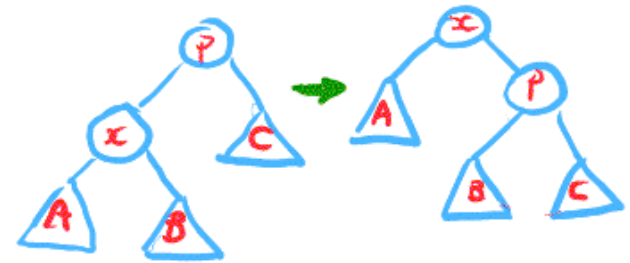
DIM. CALCOLIAMO PER INIZIARE IL COSTO AMMORTIZZATO
DI UNA OPERAZIONE DI TIPO ZIG, ZIG-ZAG, ZIG-ZIG.

NOTAZIONE

INDICHIAMO CON $S_i(x)$ E $S_f(x)$ IL NUMERO DI NODI
NEL SOTTOALBERO DI RADICE x PRIMA E DOPO L'OPERAZIONE
IN ESAME.

ANALOGAMENTE PER $R_i(x)$ E $R_f(x)$.

CASO ZIG



COSTO REALE = 1

$$\Delta\Phi = R_f(x) + R_f(p) - R_i(x) - R_i(p)$$

SI OSSERVA CHE

$$S_i(p) \geq S_f(p) \rightarrow R_i(p) \geq R_f(p) \rightarrow \Delta\Phi \leq R_f(x) - R_i(x)$$

$$S_f(x) \geq S_i(x) \rightarrow R_f(x) - R_i(x) \geq 0 \rightarrow \Delta\Phi \leq 3(R_f(x) - R_i(x))$$

$$\hat{C}_{zig} = 1 + \Delta\Phi \leq 1 + 3(R_f(x) - R_i(x))$$

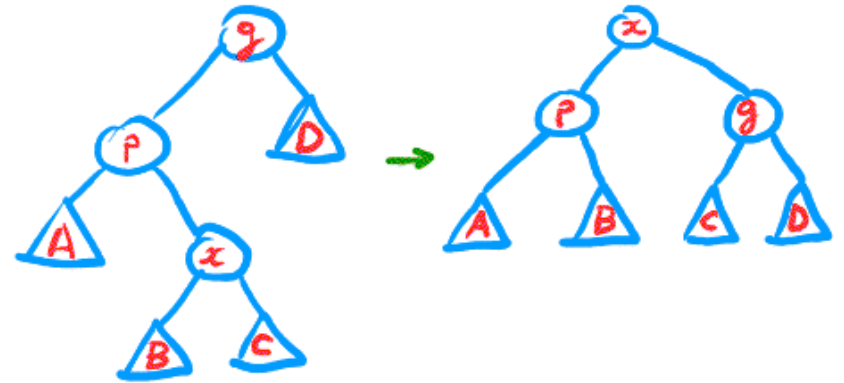
CASO ZIG-ZAG

COSTO REALE = 2

$$\Delta\Phi = R_f(x, p, g) - R_i(x, p, g)$$

SI OSSERVI QUANTO SEGUE:

- $S_f(x) = S_i(g) \rightarrow R_f(x) = R_i(g) \rightarrow \Delta\Phi = R_f(p, g) - R_i(x, p)$
 - $S_i(p) \geq S_i(x) \rightarrow R_i(p) \geq R_i(x) \rightarrow \Delta\Phi \leq R_f(p, g) - 2R_i(x)$
 - $S_f(p) + S_f(g) \leq S_f(x) \rightarrow R_f(p) + R_f(g) \leq 2R_f(x) - 2$
 $\rightarrow \Delta\Phi \leq 2(R_f(x) - R_i(x)) - 2$
 - $S_f(x) \geq S_i(x) \rightarrow R_f(x) - R_i(x) \geq 0 \rightarrow \Delta\Phi \leq 3(R_f(x) - R_i(x)) - 2$
- $$\hat{C}_{ZIG-ZAG} = 2 + \Delta\Phi \leq 2 + 3(R_f(x) - R_i(x)) - 2 = 3(R_f(x) - R_i(x)).$$



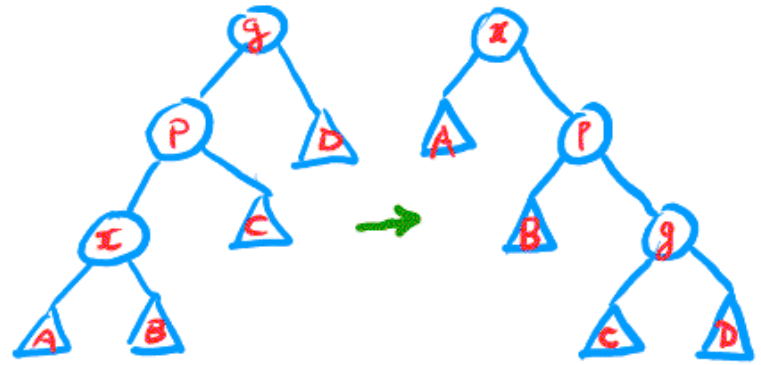
CASO ZIG-ZIG

COSTO REALE = 2

$$\Delta\Phi = R_f(x, p, y) - R_i(x, p, y)$$

SI OSSERVA CHE:

$S_f(x) = S_i(y) \rightarrow R_f(x) = R_i(y)$



$\Delta\Phi = R_f(p, y) - R_i(x, p)$

$S_f(x) \geq S_f(p) \rightarrow R_f(x) \geq R_f(p) \rightarrow \Delta\Phi \leq R_f(x, y) - R_i(x, p)$

$S_i(x) + S_f(y) \leq S_f(x) \rightarrow R_i(x) + R_f(y) \leq 2R_f(x) - 2$

$\rightarrow \Delta\Phi \leq R_f(x) + R_f(y) - R_i(x) - R_i(p) + R_i(x) \leq 3R_f(x) - 2 - 2R_i(x) - R_i(y)$

$S_i(x) \leq S_i(p) \rightarrow R_i(x) \leq R_i(p) \rightarrow \Delta\Phi \leq 3(R_f(x) - R_i(x)) - 2$

$\hat{C}_{zig-zig} = 2 + \Delta\Phi \leq 3(R_f(x) - R_i(x))$

RIASSUMENDO

$$\hat{C}_{21G} \leq 3(R_f(x) - R_i(x)) + 1$$

$$\hat{C}_{21G-2AG} > \hat{C}_{21G-21G} \leq 3(R_f(x) - R_i(x))$$

$$\hat{C}_{\text{SPLAY}} = \sum_{j=1}^k \hat{C}_j \leq \sum_{j=1}^k 3(R_f^{(j)}(x) - R_i^{(j)}(x)) + 1$$

MA $R_f^{(j)}(x) = R_i^{(j+1)}(x)$ QUINDI

$$\begin{aligned} \hat{C}_{\text{SPLAY}} &\leq \sum_{j=1}^k 3(R_i^{(j+1)}(x) - R_i^{(j)}(x)) + 1 \\ &\leq 3(R_i^{(k+1)}(x) - R_i^{(1)}(x)) + 1 \\ &= 3(R(\text{root}(T)) - R(x)) + 1 \quad \blacksquare \end{aligned}$$

SI OSSERVA CHE

$$3 (R(\text{root}(T)) - R(x)) + 1 = O(\log |T|)$$

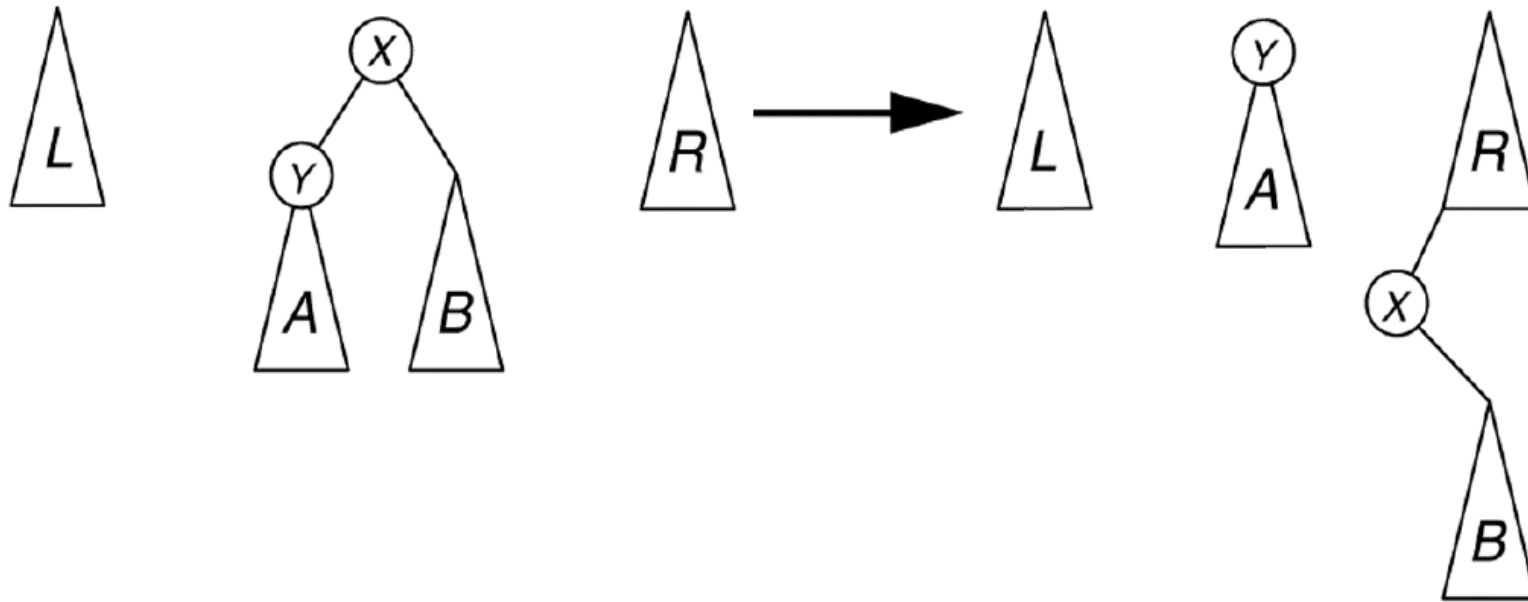
TOP-DOWN SPLAY TREES

- NEI LUCIDI PRECEDENTI ABBIAMO ILLUSTRATO I COSIDDETTI **BOTTOM-UP SPLAY TREES**
- I **BOTTOM-UP SPLAY TREES** SONO PIÙ FACILI DA ANALIZZARE, MA DAL PUNTO DI VISTA PRATICO PRESENTANO ALCUNI PROBLEMI IMPLEMENTATIVI (È QUINDI DI EFFICIENZA)
- INFATTI LE ROTAZIONI **ZIG**, **ZIG-ZIG** E **ZIG-ZAG** HANNO LUOGO **BOTTOM-UP** E QUINDI OCCORRE
 - MEMORIZZARE IL CAMMINO DALLA RADICE AL NODO CERCATO, OPPURE
 - MANTENERE PER CIASCUN NODO UN PUNTATORE AL PADRE

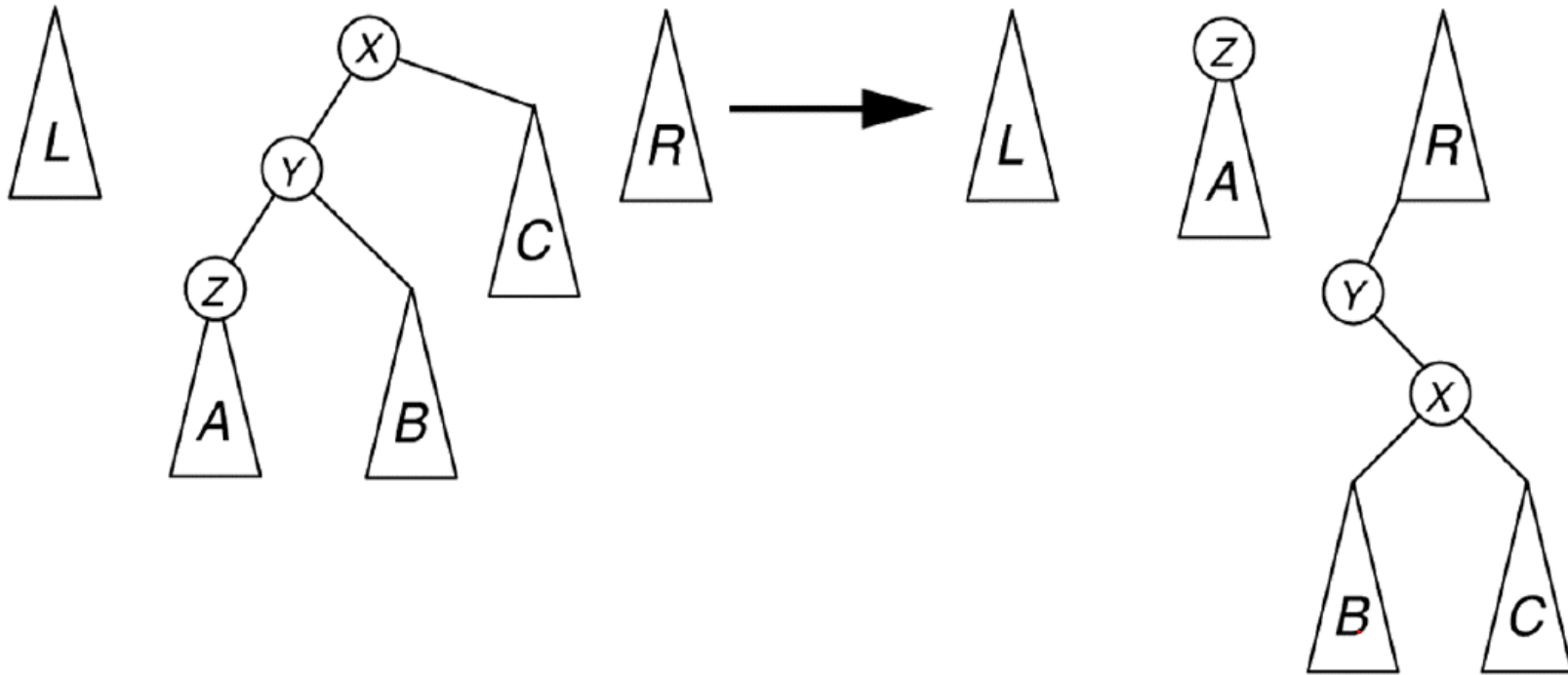
- LA VARIANTE **TOP-DOWN** CONSENTE DI RISOLVERE EGREGIAMENTE IL PRECEDENTE PROBLEMA, PUR MANTENENDO UN COSTO AMMORTIZZATO **LOGARITMICO** PER SPLAY
- L'IDEA DI BASE E' LA SEGUENTE:
 - MENTRE SI PROCEDE NELLA RICERCA DI UN DATO NODO, SI TRATTANO OPPORTUNAMENTE I NODI INCONTRATI NONCHE' I LORO SOTTOALBERI.
 - NEL CORSO DELL'OPERAZIONE DI **SPLAY**, L'ALBERO RISULTERA' SPEZZATO IN **TRE** PARTI:
 - o UN ALBERO **L**
 - o UN ALBERO **T'** DI RADICE UN CERTO NODO **x**
 - o UN ALBERO **R**
 TALI CHE $L \leq T' \leq R$ (NEL SENSO DELLE CHIAVI CONTENUTE IN ESSI)
- INIZIALMENTE **x** E' LA RADICE DEL NOSTRO ALBERO **T** ED **L** E **R** SONO **VUOTI**

- SUCCESSIVAMENTE, SCENDIAMO DI DUE LIVELLI PER VOLTA, SE POSSIBILE, ALLA RICERCA DEL NOSTRO NODO ED APPLICHIAMO L'OPERAZIONE DI ZIG-ZAG O ZIG-ZIC (DESCRITTE SOTTO) A SECONDA DELLA SITUAZIONE.
NEL CASO SI POSSA SCENDERE DI UN SOLO LIVELLO, SI APPLICHERA' L'OPERAZIONE ZIG
- LE OPERAZIONI ZIG, ZIG-ZIC E ZIG-ZAG HANNO L'EFFETTO DI AGGIORNARE I TRE ALBERI L, T' E R MANTENENDO L'INVARIANTE $L \leq T' \leq R$
- ALLA FINE, I TRE ALBERI VENGONO RIASSEMBLATI IN UN UNICO ALBERO

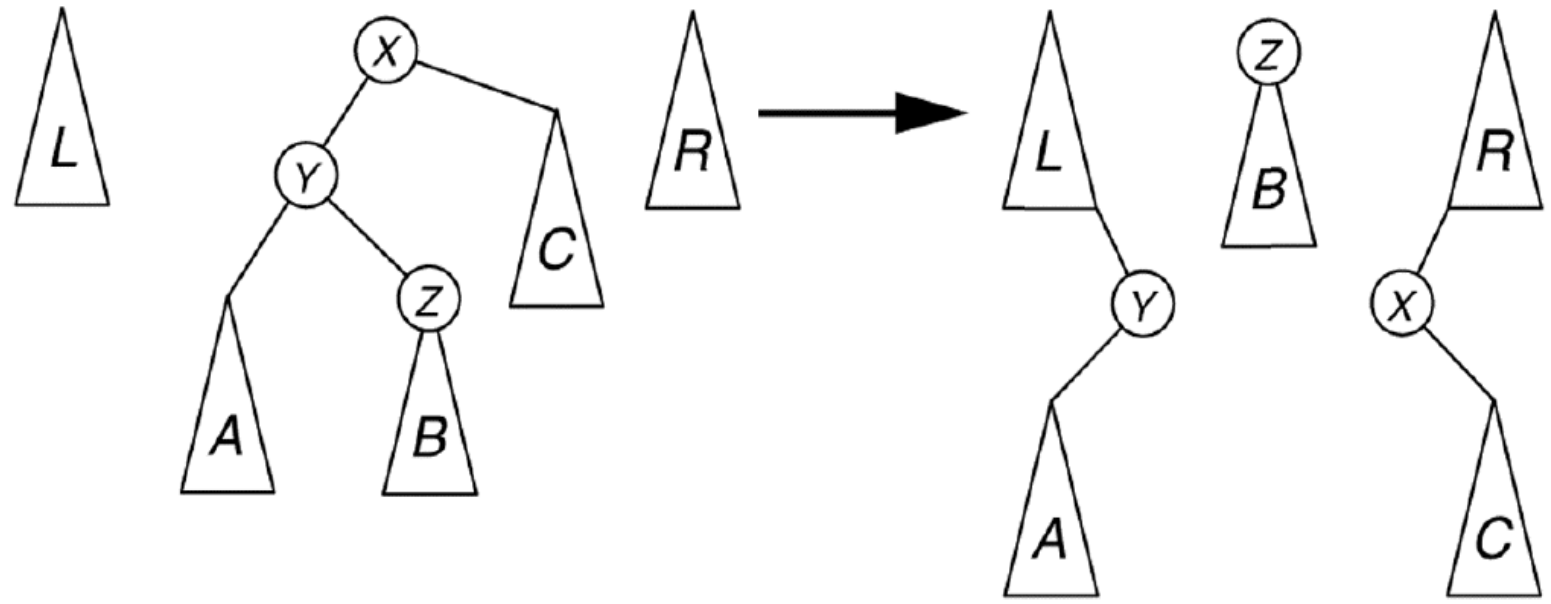
OPERAZIONE ZIG



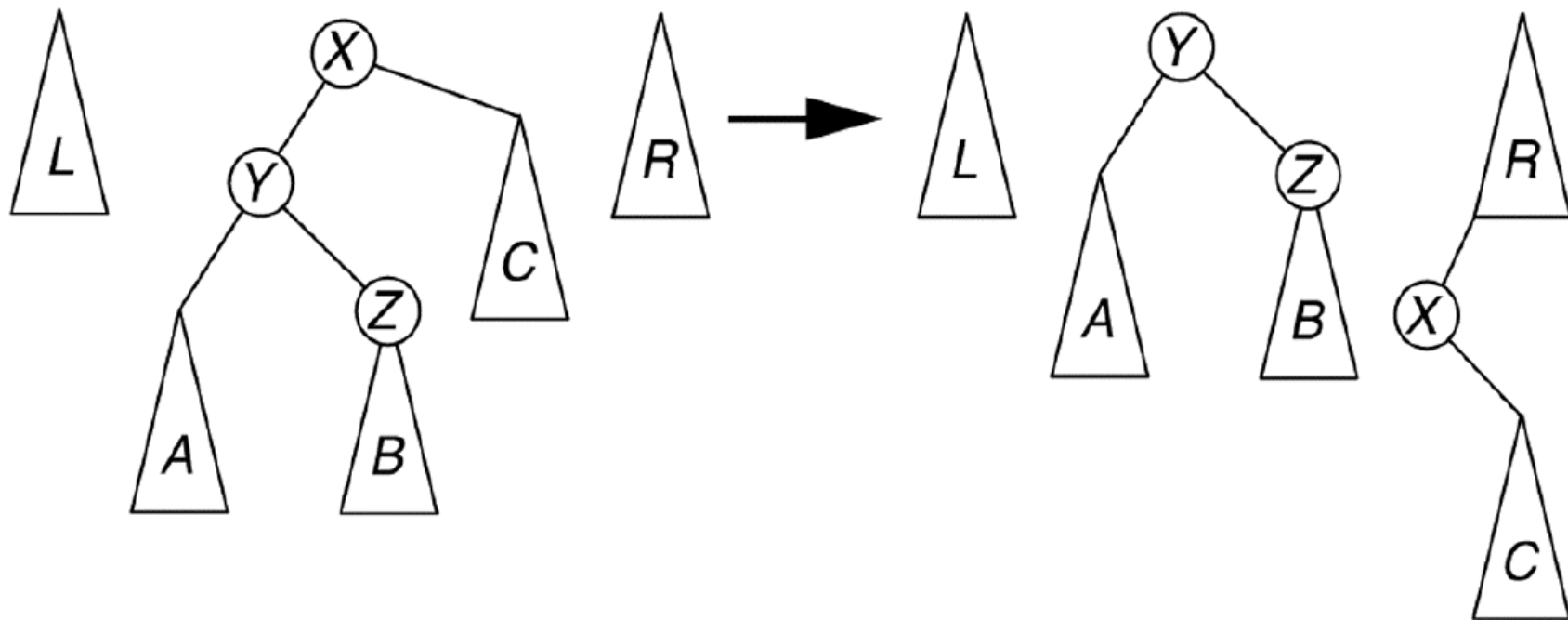
OPERAZIONE ZIG-ZIG



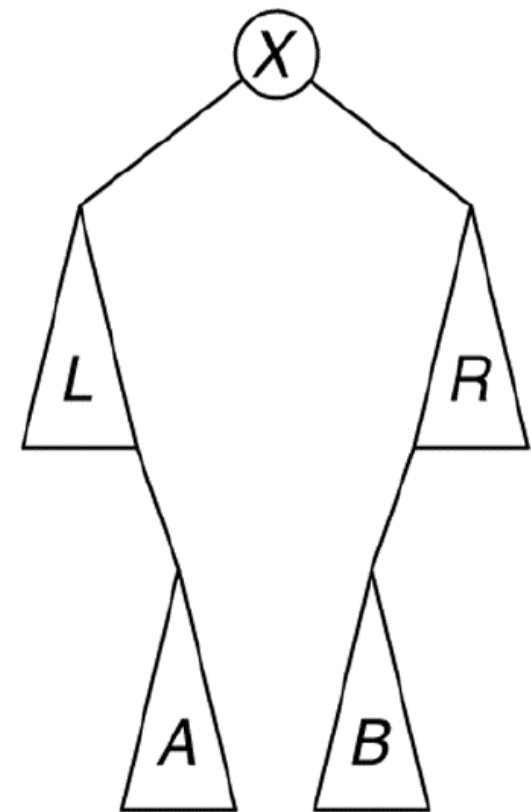
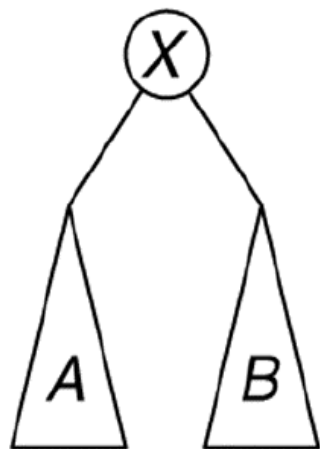
OPERAZIONE ZIG-ZAG



OPERAZIONE ZIG-ZAG (SEMPLIFICATA)



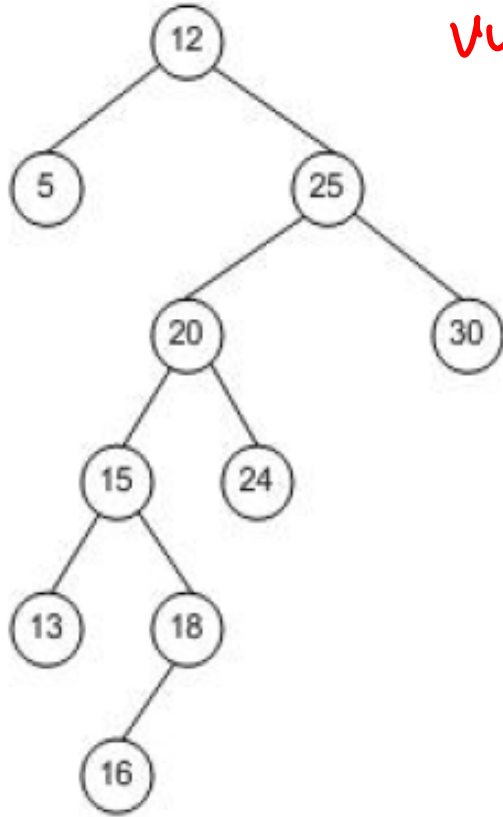
RIASSEMBLAGGIO FINALE



ESEMPIO

SPLAY (19)

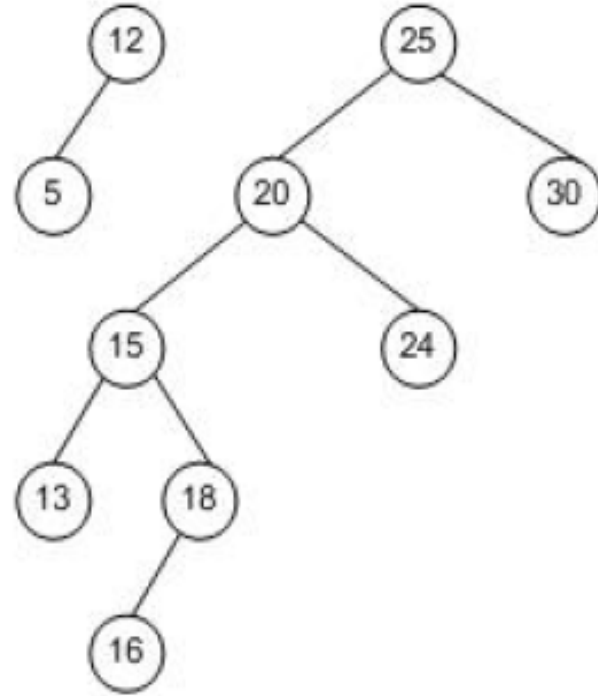
VUOTO



VUOTO



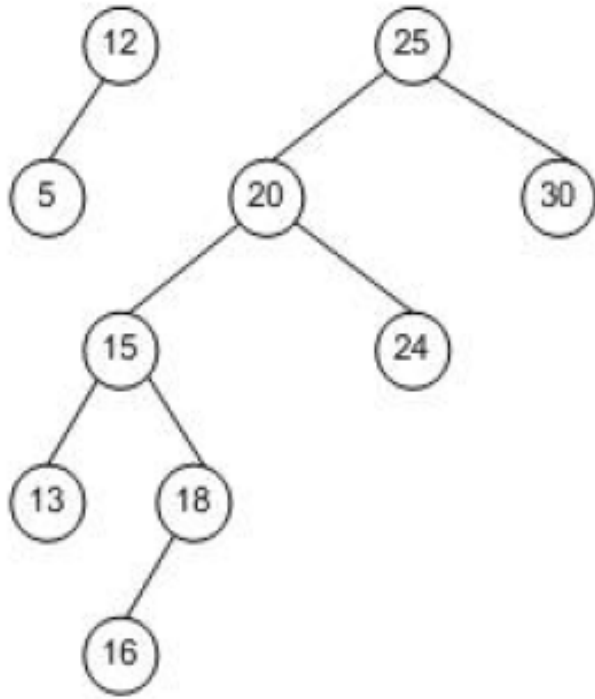
ZIG-ZAG
(SEMPLIFICATO)



VUOTO

ESEMPIO

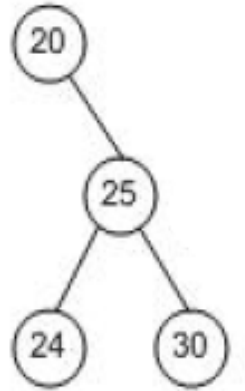
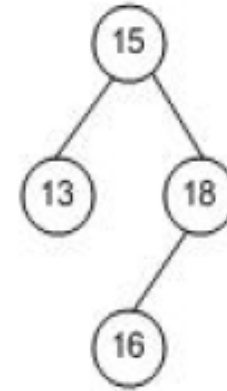
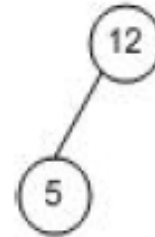
SPLAY (19)



VUOTO

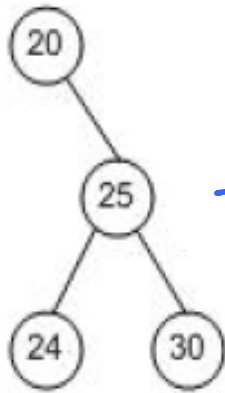
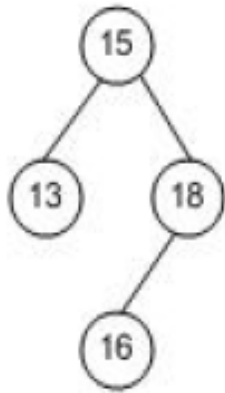
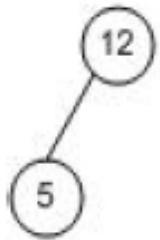


ZIG-ZIG

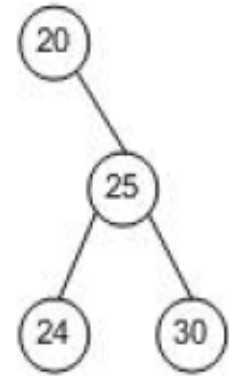
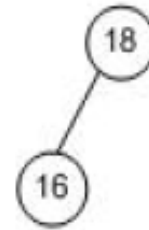
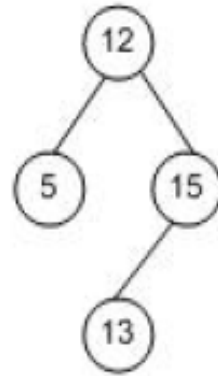


ESEMPIO

SPLAY (19)

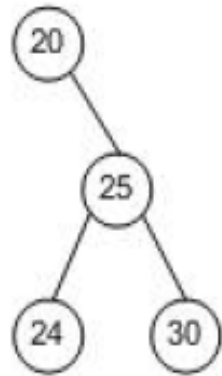
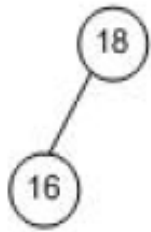
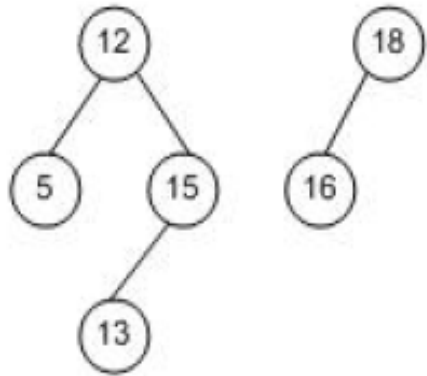


⇒
2LG

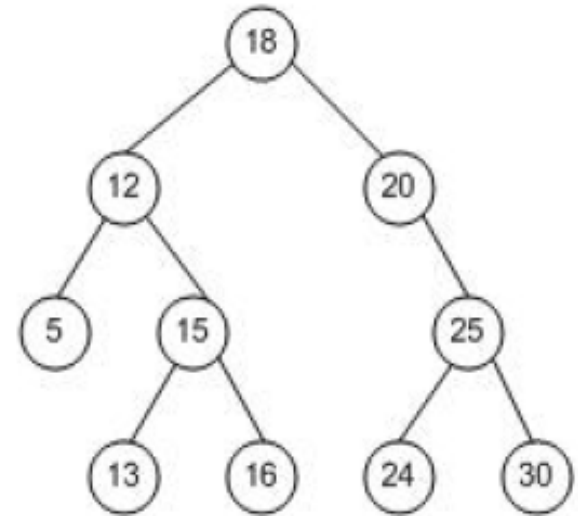


ESEMPIO

SPLAY (19)

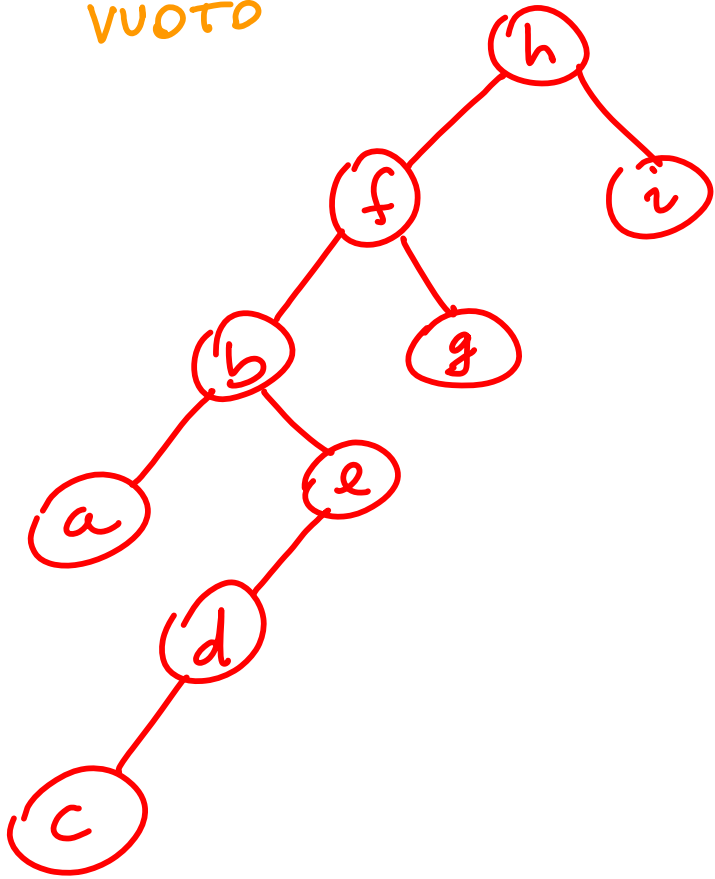


RIASSEMBLAGGIO



ESEMPIO 2: SPLAY (c)

VUOTO

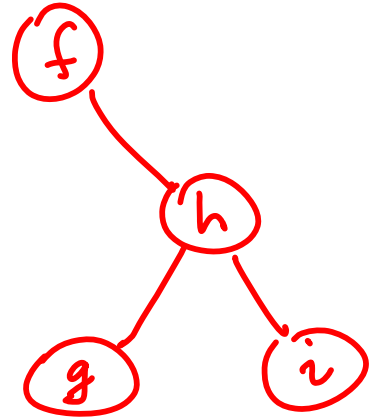
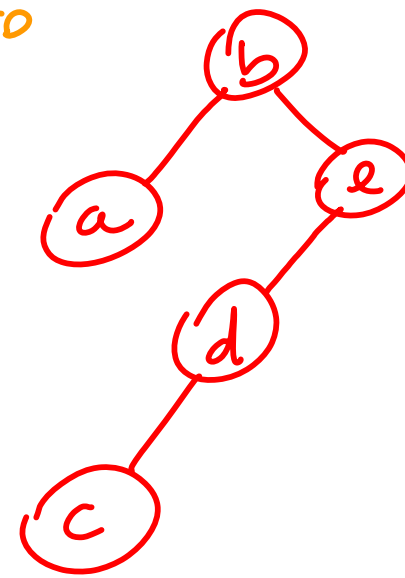


VUOTO



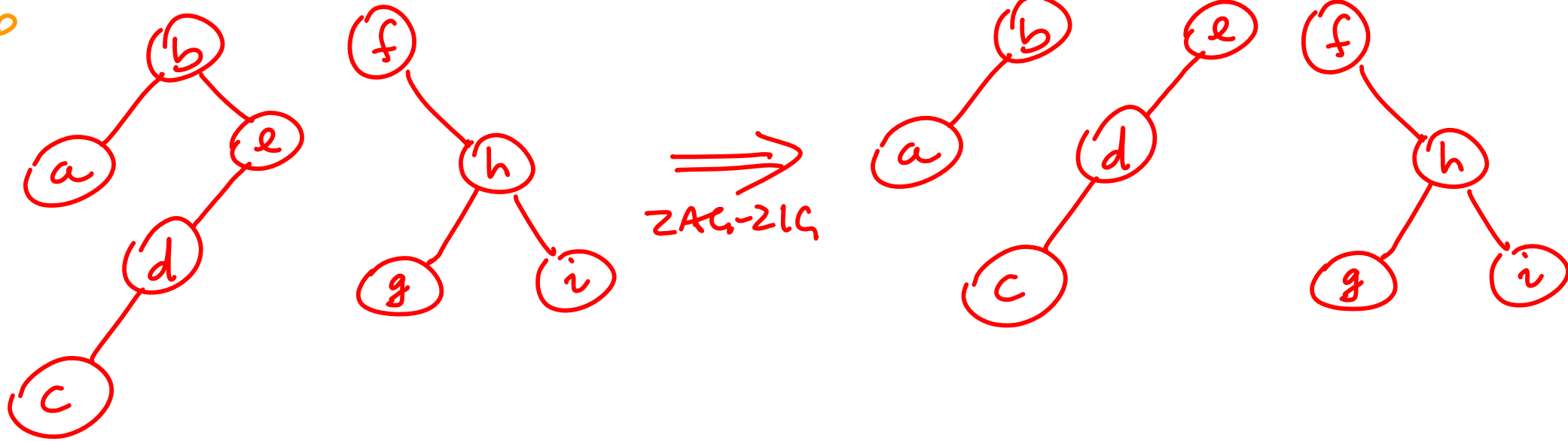
ZIG-ZIG

VUOTO

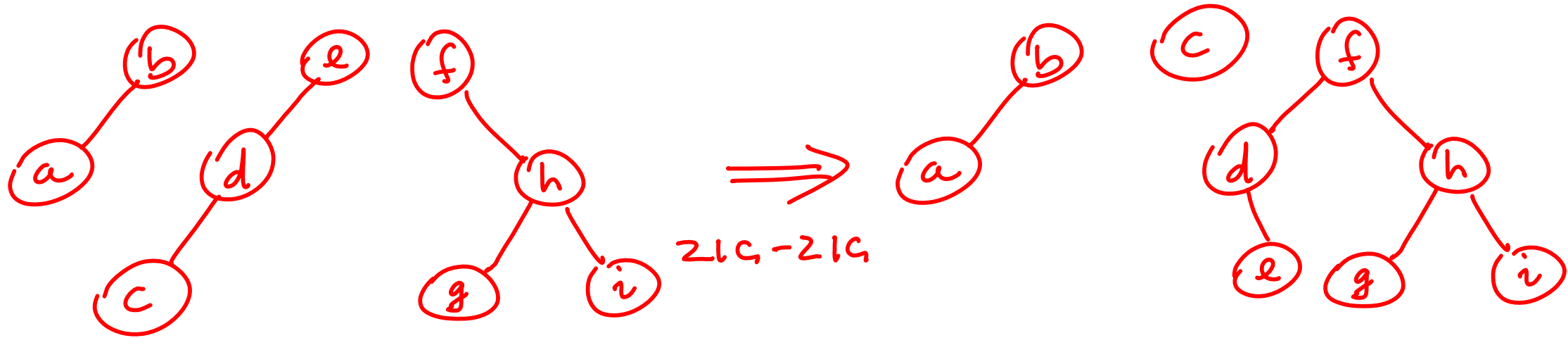


ESEMPIO 2: SPLAY (c)

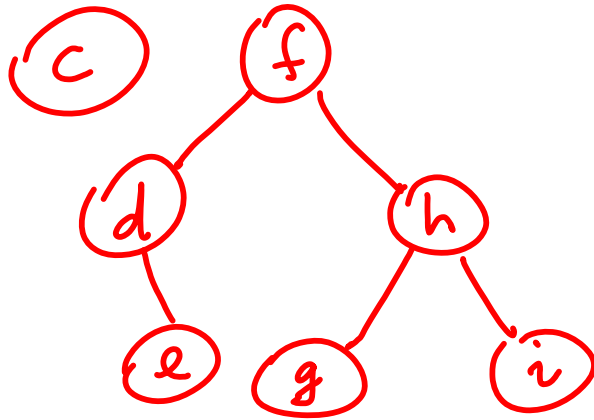
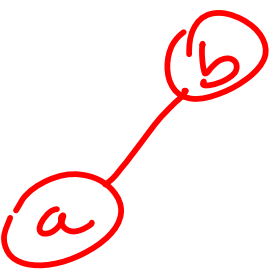
VUOTO



ESEMPIO 2: SPLAY (c)



ESEMPIO 2: SPLAY (c)



⇒
RIASSEMBLAGGIO

