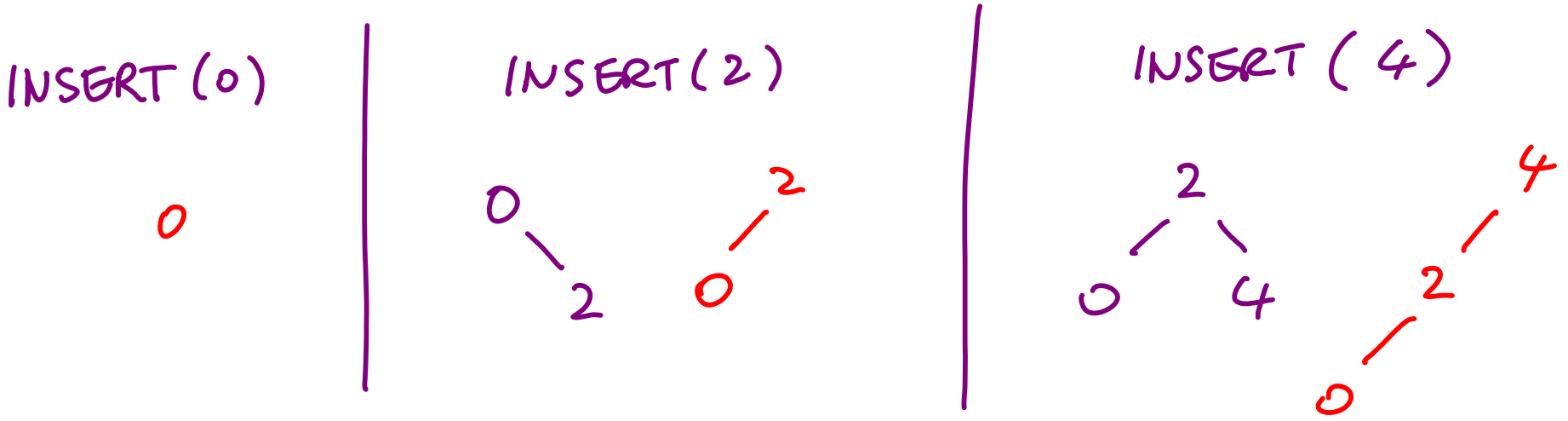
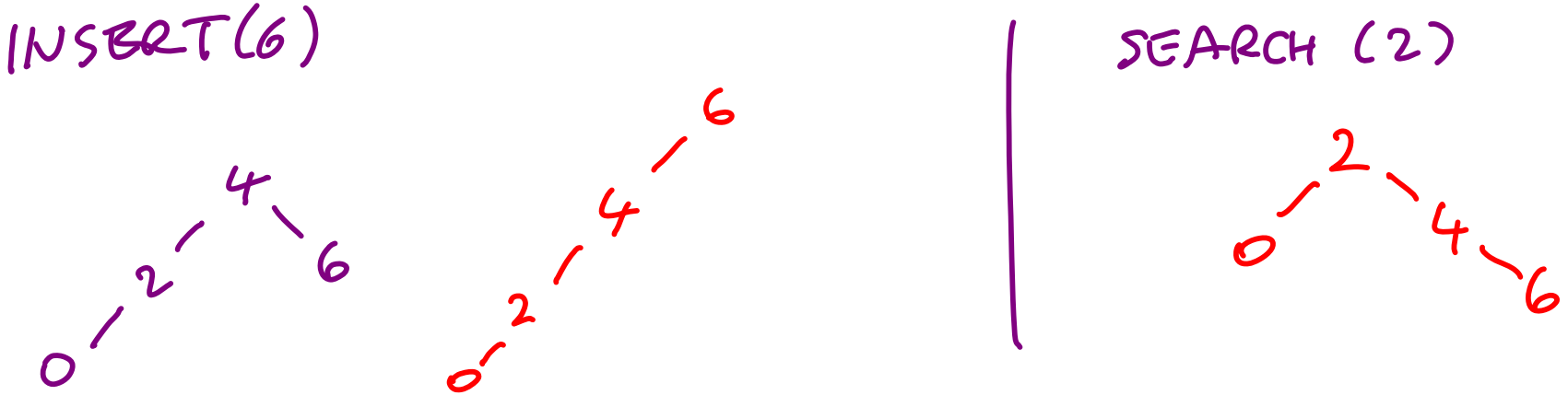
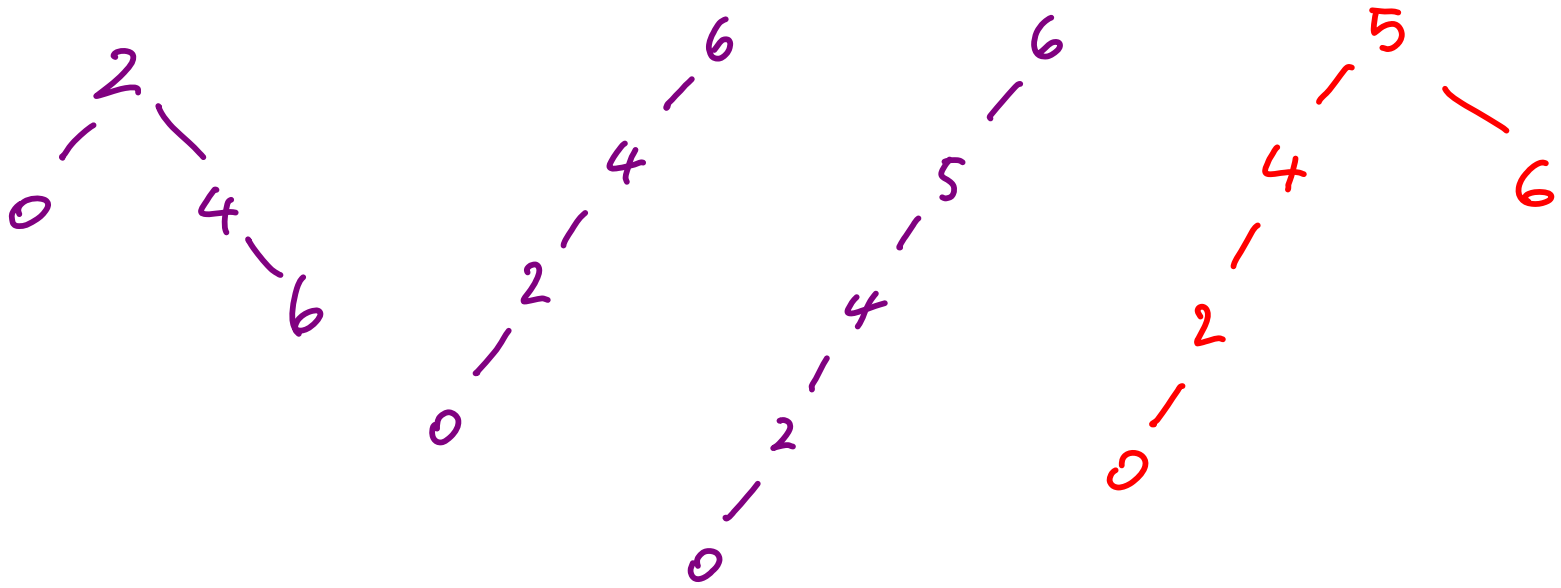
(a) Si eseguano su uno splay tree inizialmente vuoto le seguenti operazioni, nell'ordine dato:
- INSERT 0, 2, 4, 6
- SEARCH 2
- INSERT 5
- SEARCH 0
- DELETE 4

(b) Si descriva come modificare gli SPLAY TREE affinché possa essere gestita in maniera efficiente anche l'operazione SELECT$(k, T)$ per la ricerca del $k$-esimo elemento nello splay tree $T$. Si discuta inoltre la complessità ammortizzata dell'implementazione proposta per l'operazione SELECT$(k, T)$.

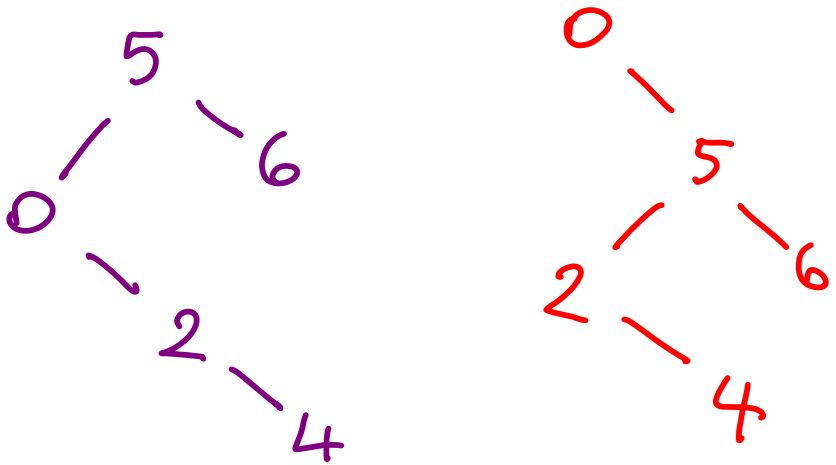


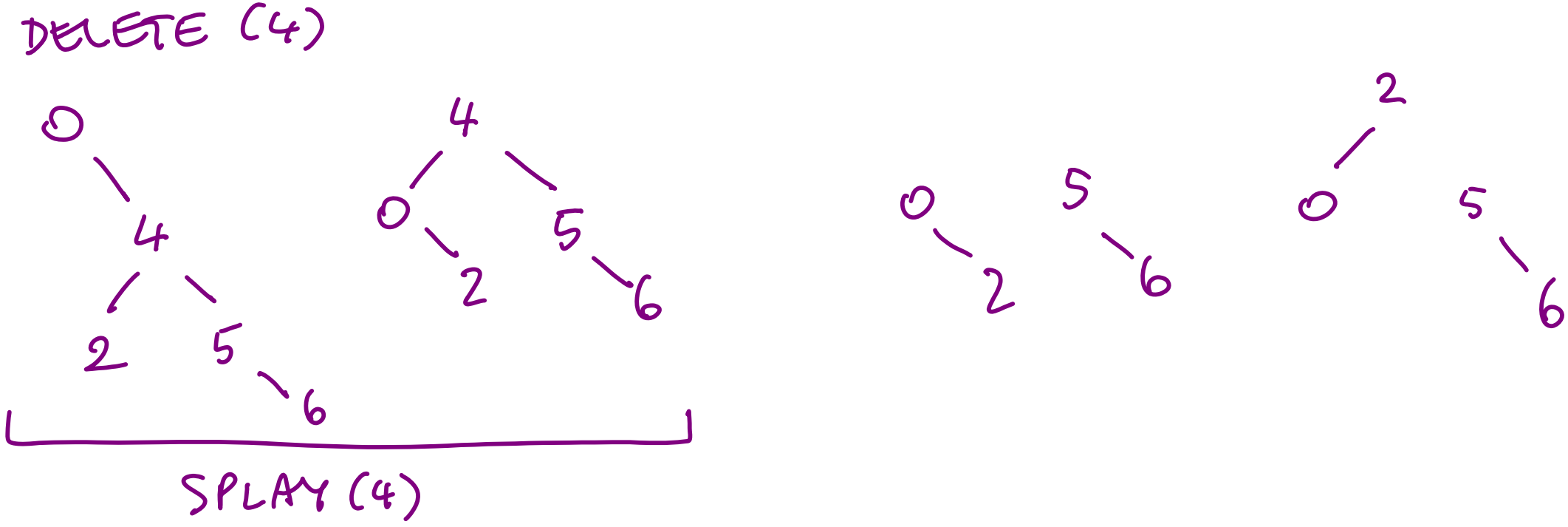



INSERT (5)

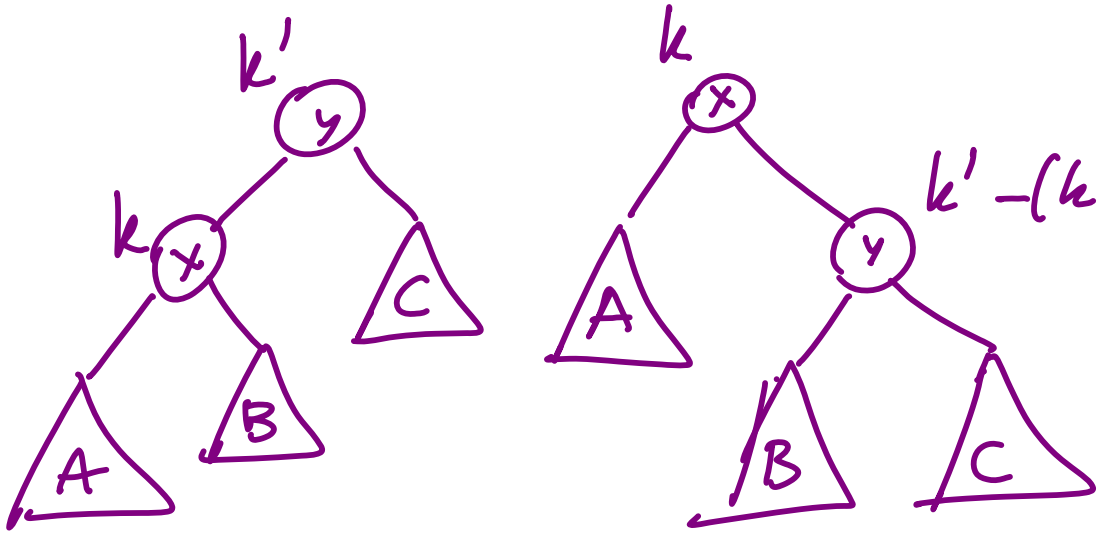



SEARCH (0)




DELETE (4)
0
4
2
5
6
4
0
5
2
6
SPLAY (4)
0
5
2
6
2
0
5
6
2
0
5
6

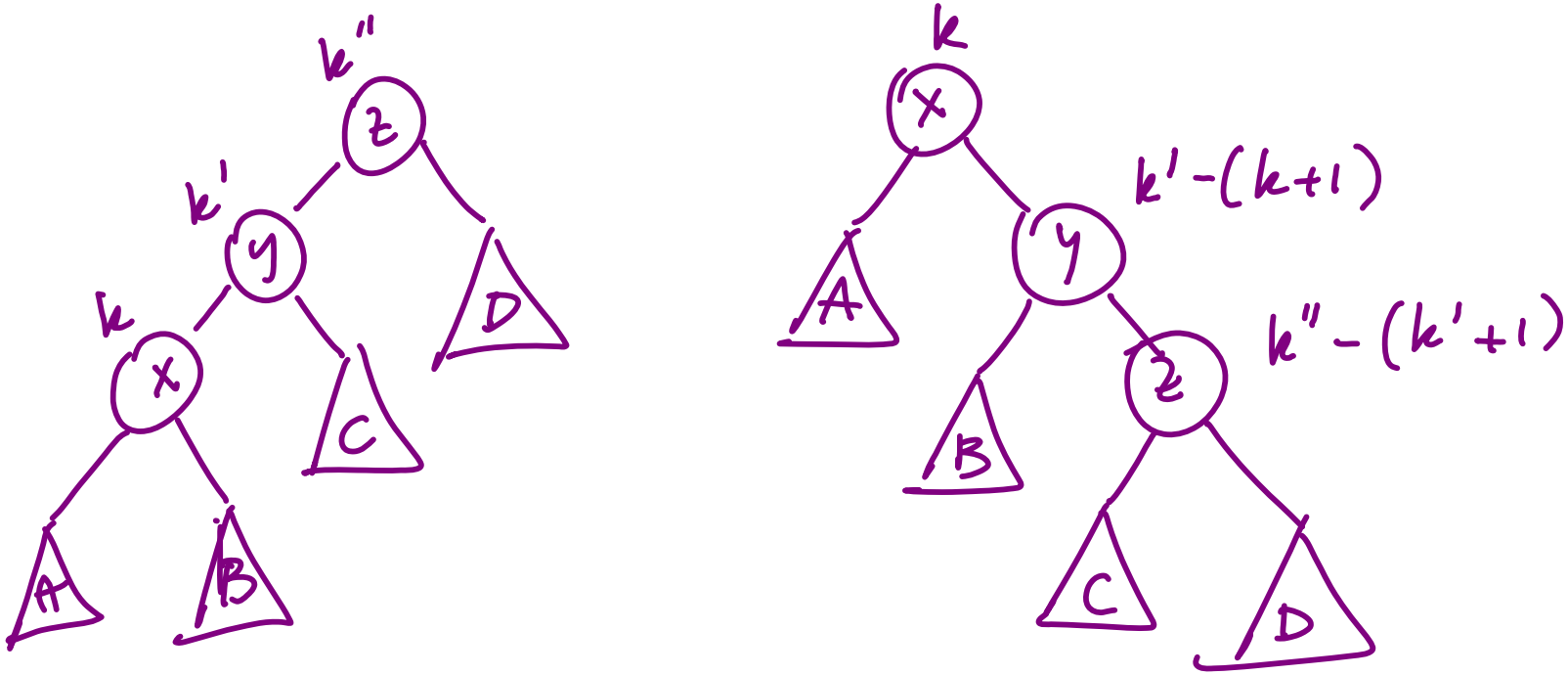
## 21G




$k = |\text{sottoalbero sinistro}|$

$k' - (k+1)$

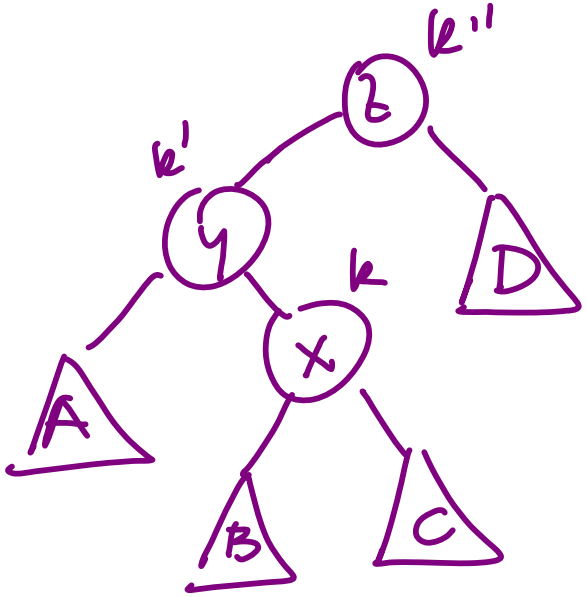
## 21G-21G

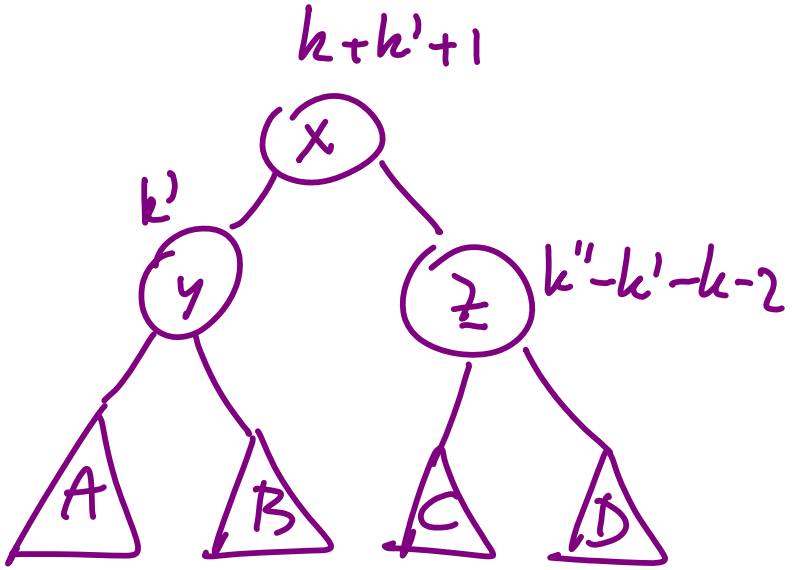



$\#B = k' - (k+1)$

$\#C = k'' - (k'+1)$

## ZIG - ZAG




$\#A = k'$

$\#B = k$

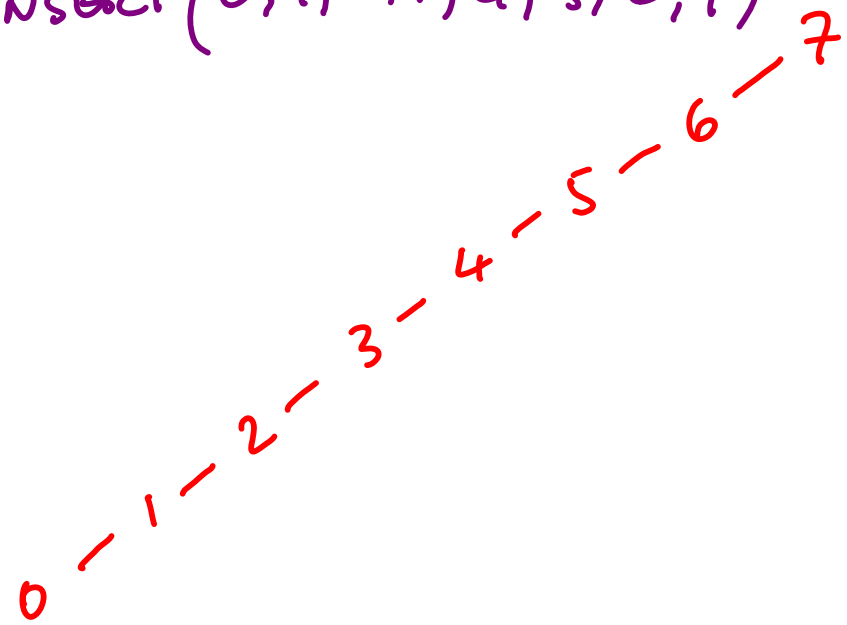
$\#C = k'' - 1 - \#A - 1 - \#B$

$= k'' - k' - k - 2$

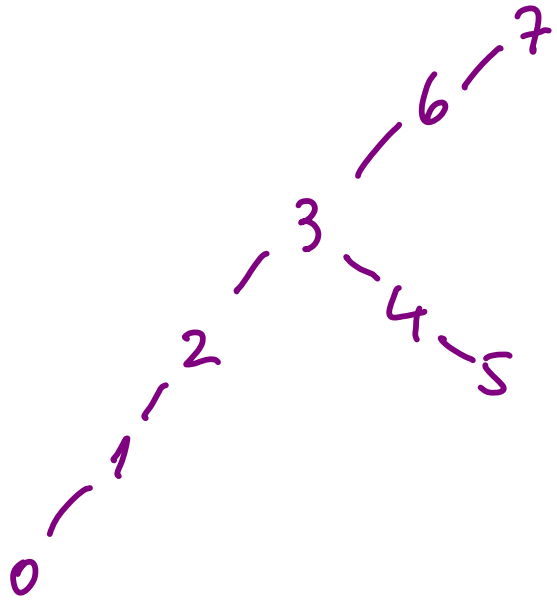
(a) Si eseguano nell'ordine dato le seguenti operazioni su uno splay tree inizialmente vuoto:
- INSERT 0, 1, 2, 3, 4, 5, 6, 7
- SEARCH 3
- INSERT 8
- SEARCH 5
- DELETE 3
- SEARCH 7

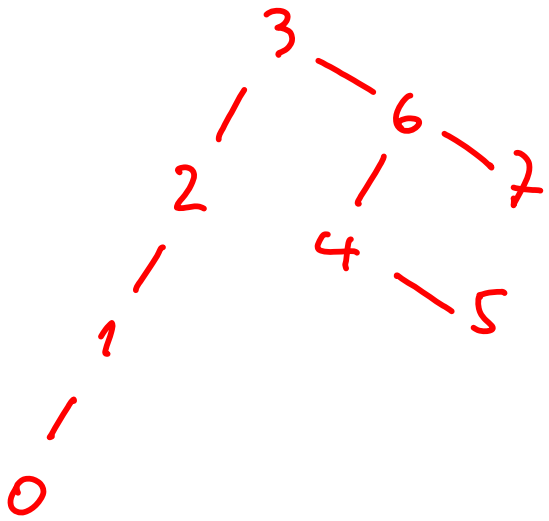
(b) Si descriva come modificare gli SPLAY TREE affinché possa essere gestita in maniera efficiente anche l'operazione SPLAY-L-D($T$) per la ricerca della minima chiave residente in un nodo di profondità massima.
In particolare, si descrivano un'implementazione della procedura SPLAY-L-D($T$) e le modifiche da apportare alle operazioni *zig-zag*, *zig-zig* e *zig*.
Qual è il costo ammortizzato dell'operazione SPLAY-L-D($T$) ? Perché?

INSERT(0,1,2,3,4,5,6,7)

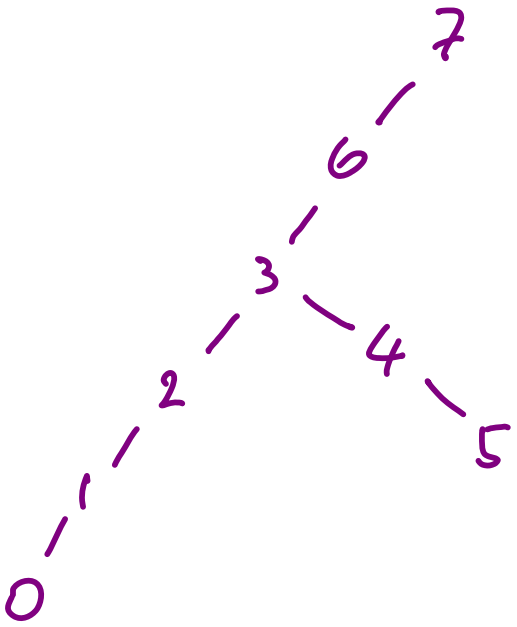


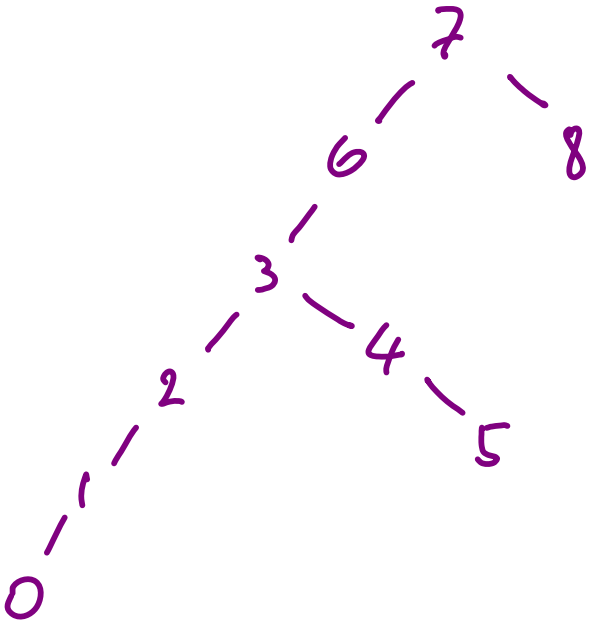
SEARCH (3)

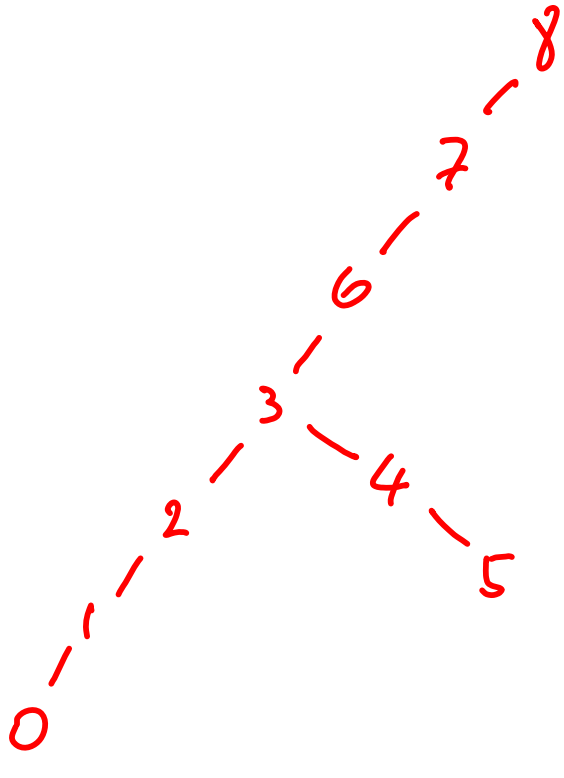


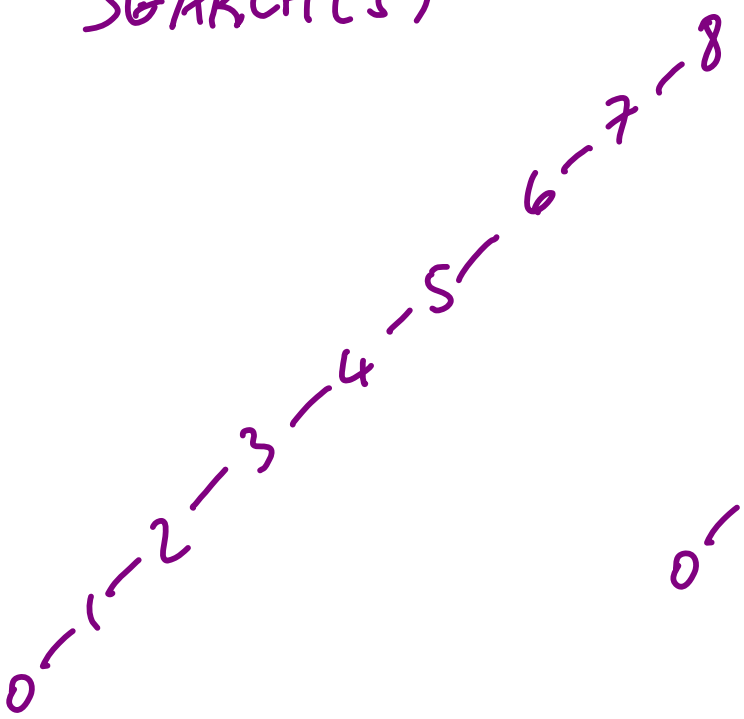

INSERT (8)

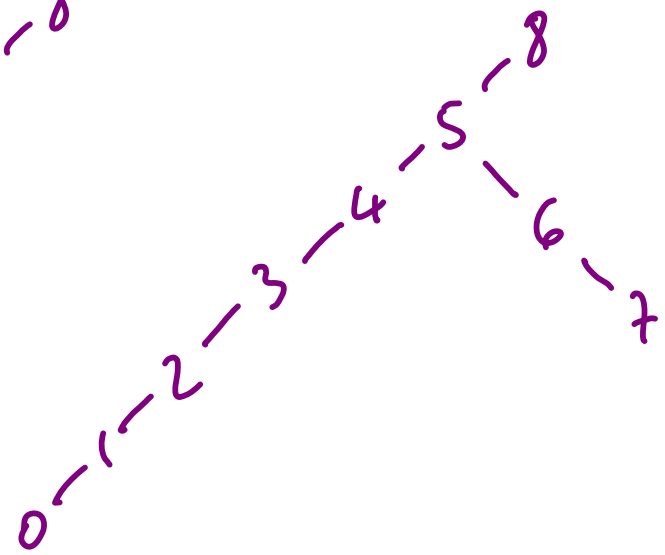


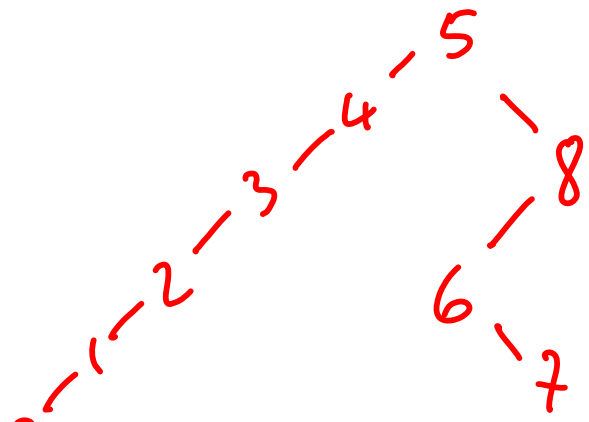



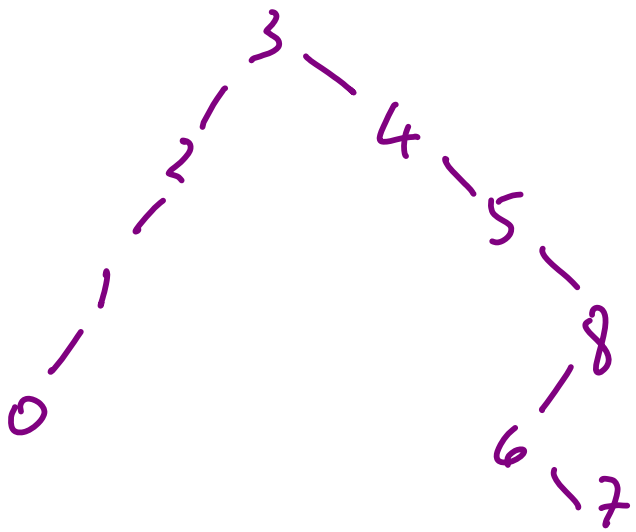
SEARCH(5)

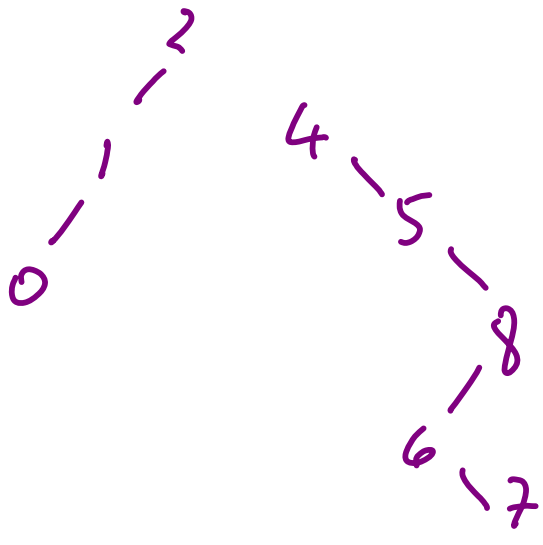


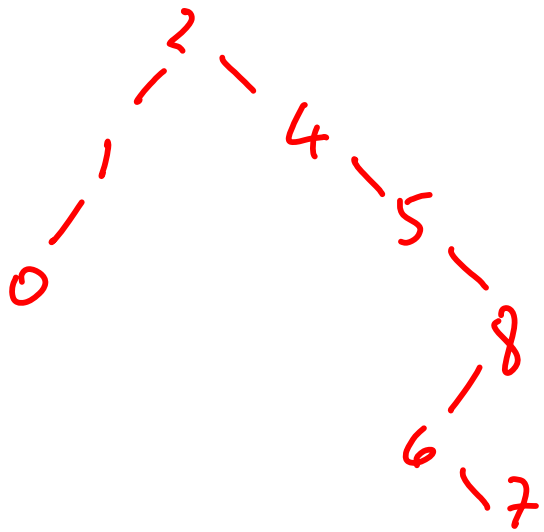



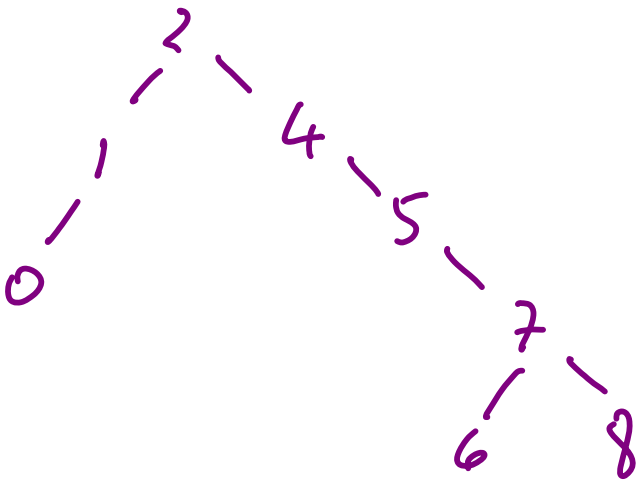

DELETE(3)

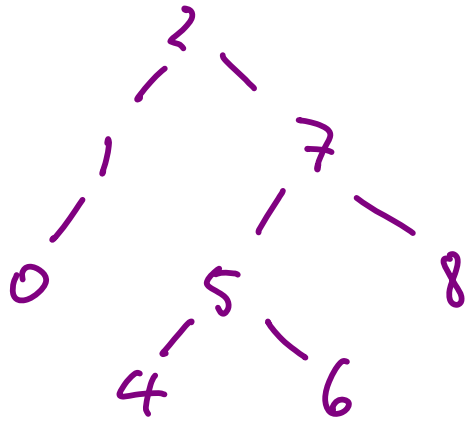


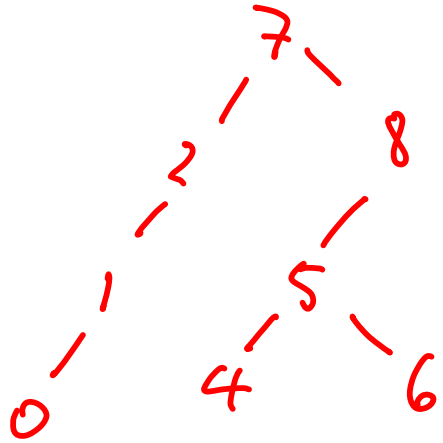



SEARCH (7)

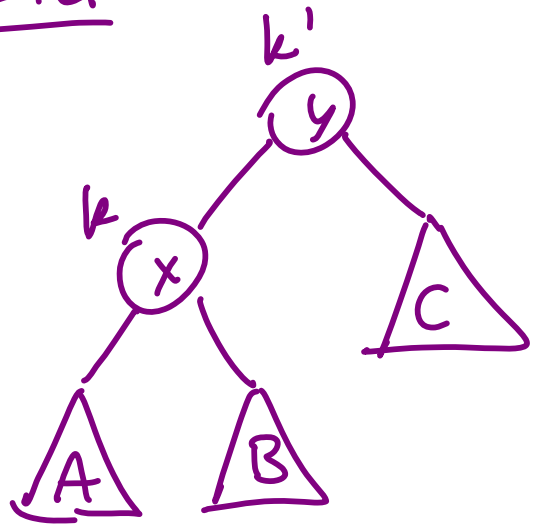


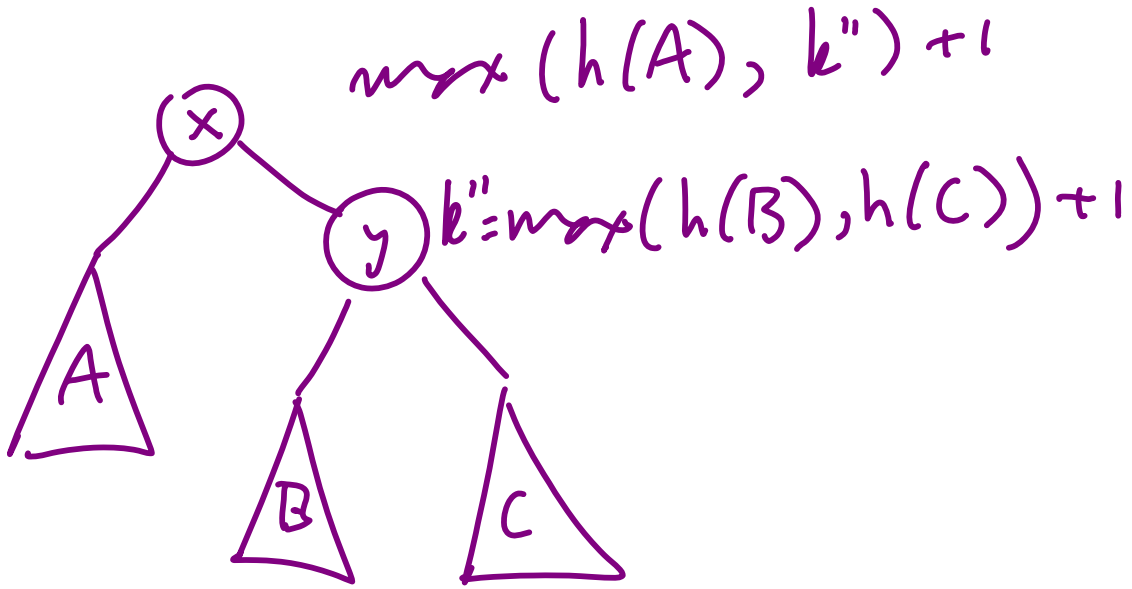



(k) $k$ = profondità

21G

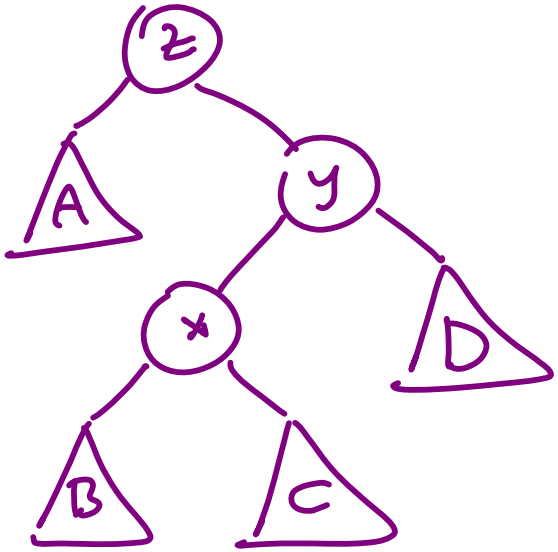


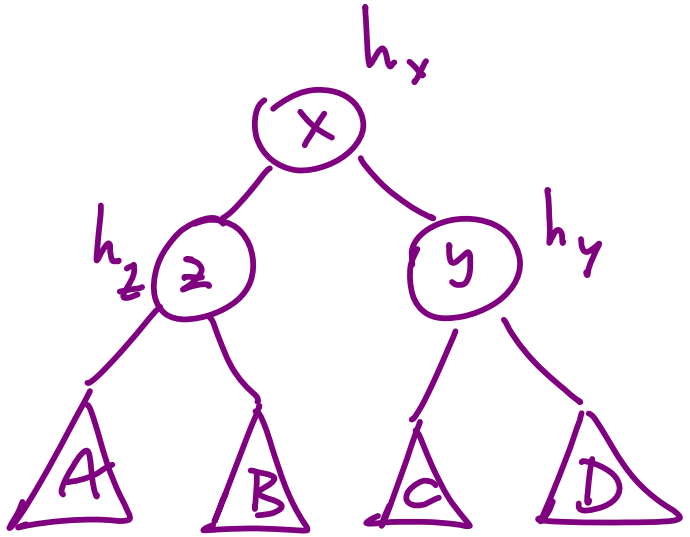




$k = \max(h(A), h(B)) + 1$

$k' = \max(k, h(C)) + 1$

21G - 2AG




$$h_z = \max(h(A), h(B)) + 1$$

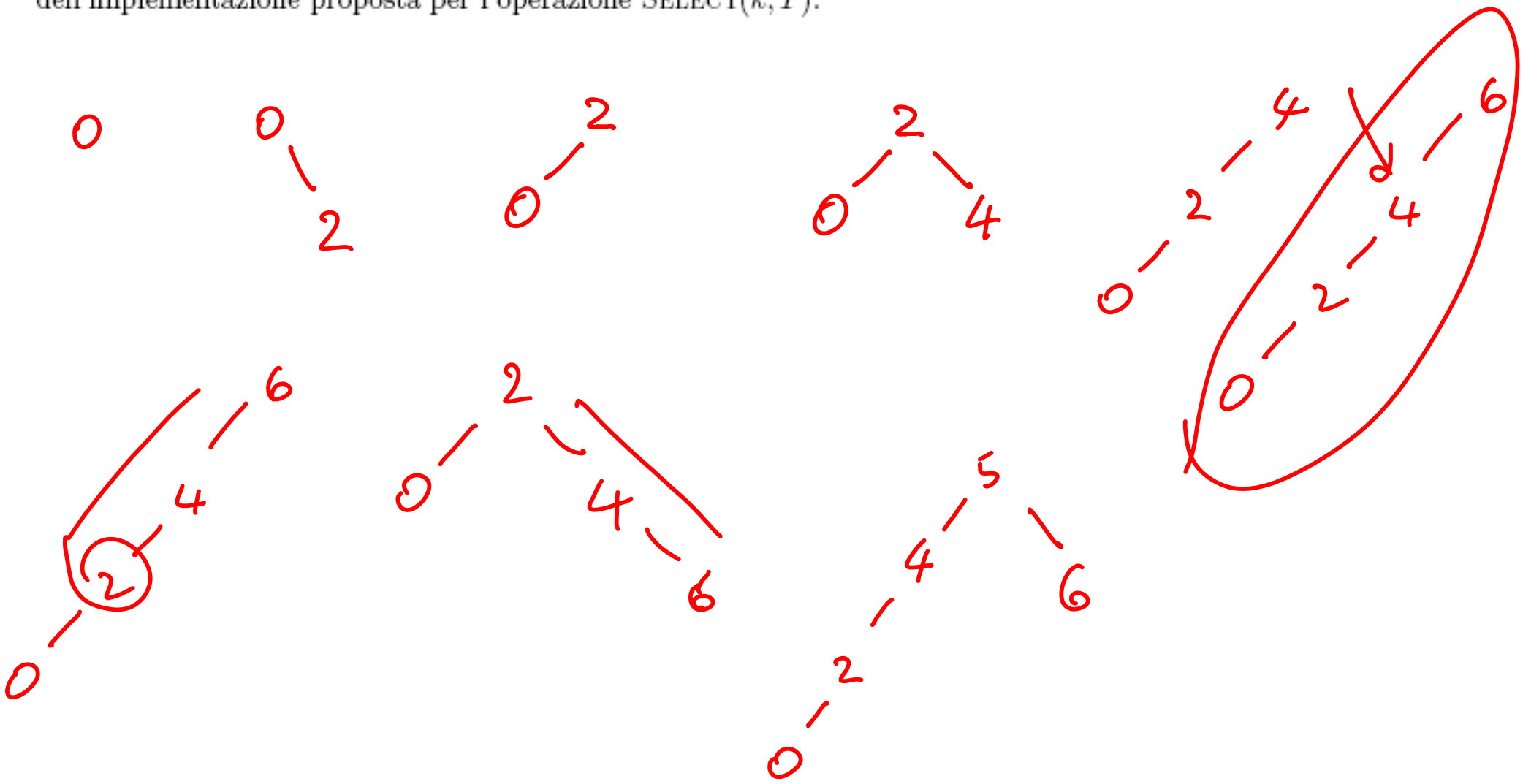
$$h_y = \max(h(C), h(D)) + 1$$

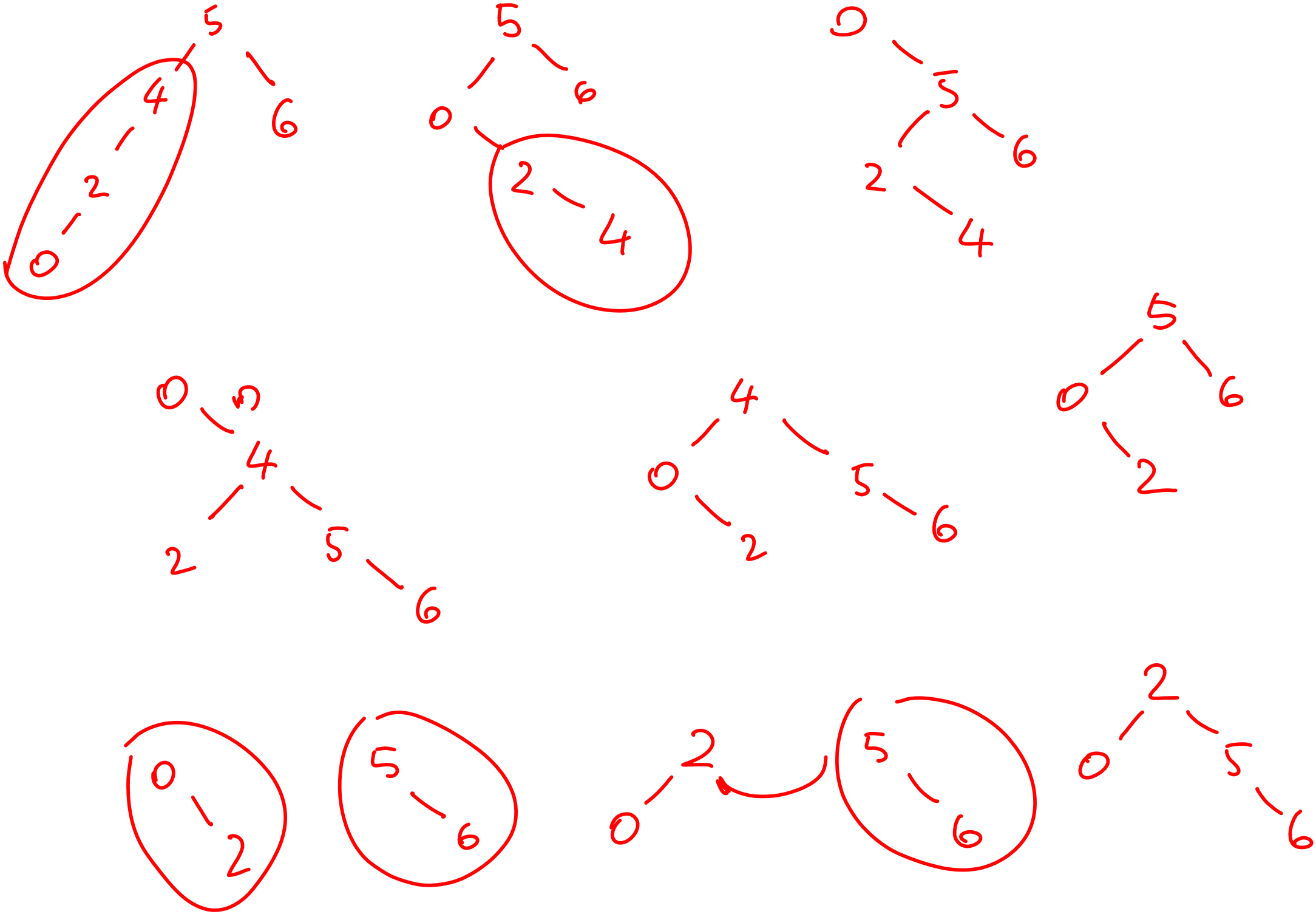
$$h_x = \max(h_z, h_y) + 1$$

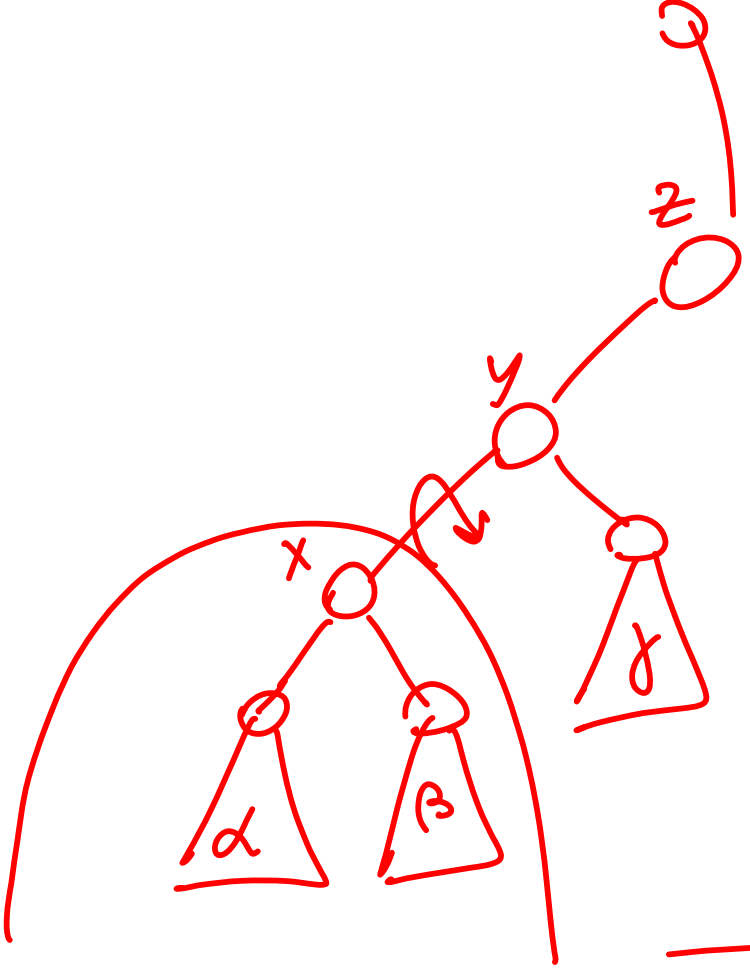
ecc.

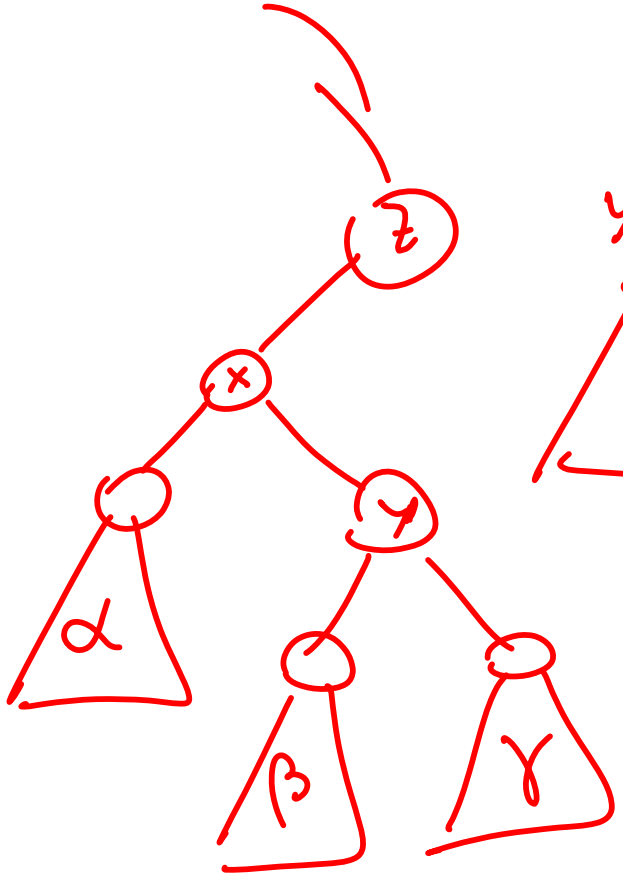
(a) Si eseguano su uno splay tree inizialmente vuoto le seguenti operazioni, nell'ordine dato:
- INSERT 0, 2, 4, 6
- SEARCH 2
- INSERT 5
- SEARCH 0
- DELETE 4

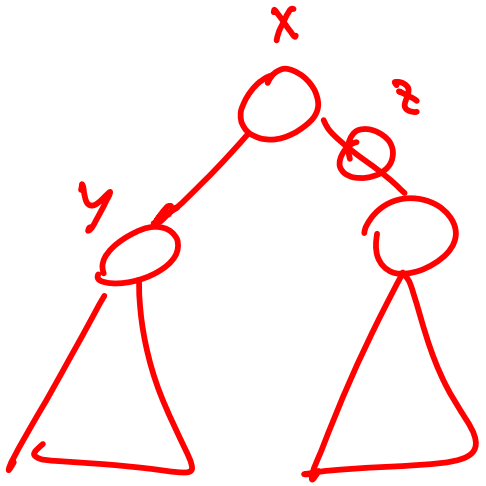
(b) Si descriva come modificare gli SPLAY TREE affinché possa essere gestita in maniera efficiente anche l'operazione SELECT$(k, T)$ per la ricerca del $k$-esimo elemento nello splay tree $T$. Si discuta inoltre la complessità ammortizzata dell'implementazione proposta per l'operazione SELECT$(k, T)$.




5
6
4
2
0
5
6
0
2
4
0
5
6
2
4
0
5
6
0
5
4
2
5
6
4
0
5
6
2
5
0
6
2
0
2
5
6
2
0
2
5
6
2
0
5
6




$c(z) := c(x)$

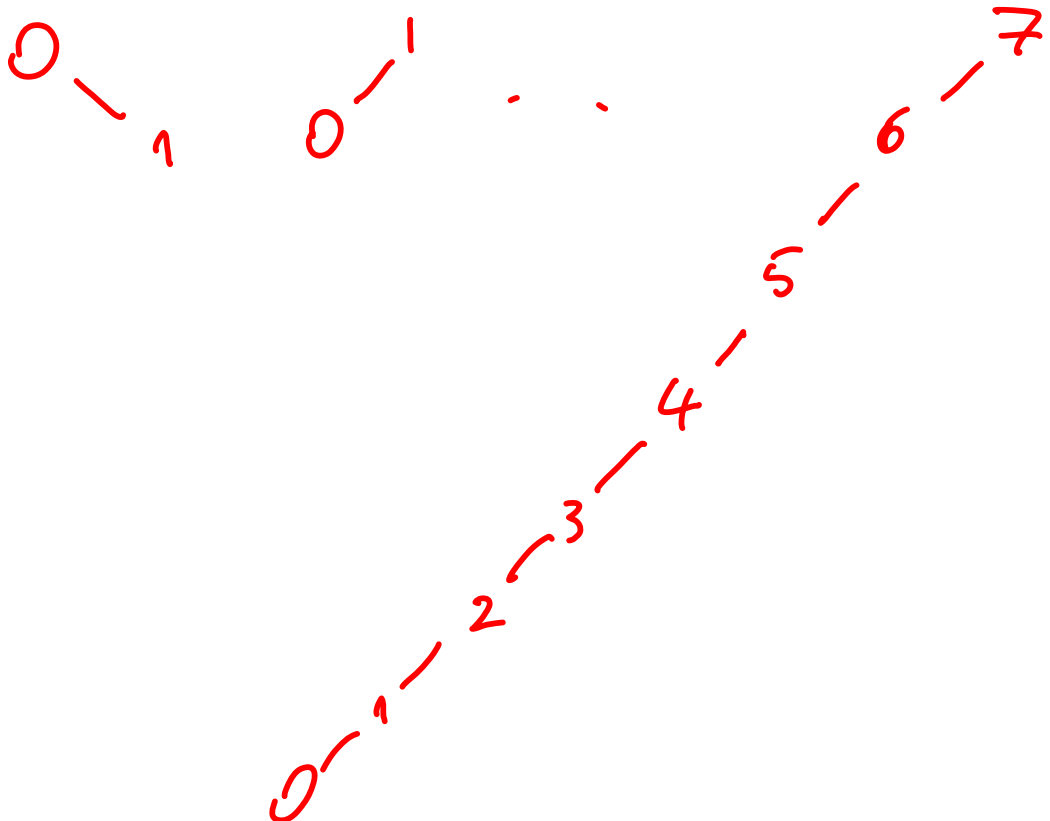
$c(x) := c(N) + 1$

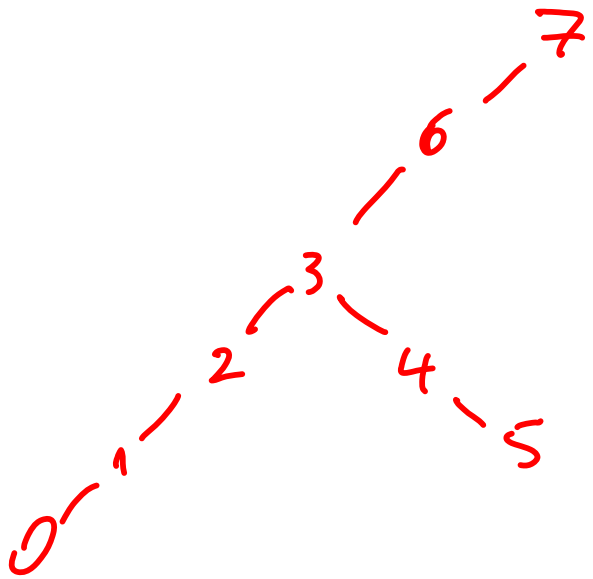
$c(z) := 0$

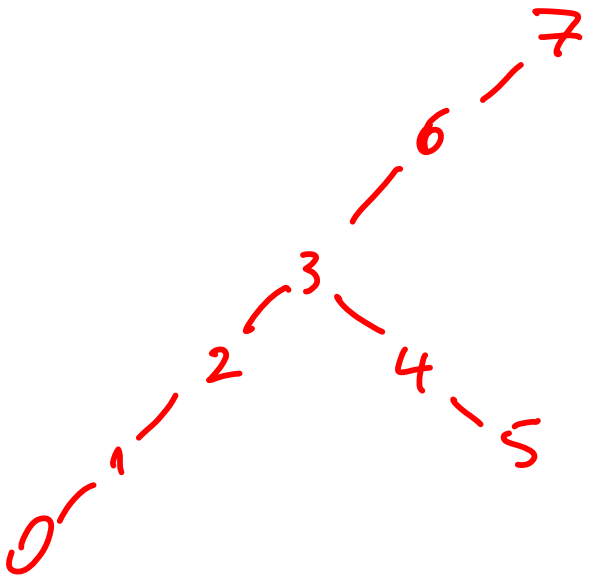
$$c(y) := c(y) - c(x) - 1$$

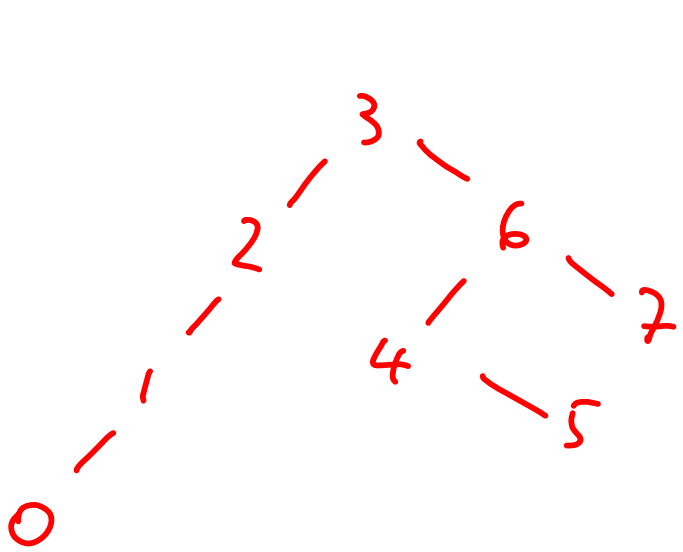
(a) Si eseguano nell'ordine dato le seguenti operazioni su uno splay tree inizialmente vuoto:
- INSERT 0, 1, 2, 3, 4, 5, 6, 7
- SEARCH 3
- INSERT 8
- SEARCH 5
- DELETE 3
- SEARCH 7

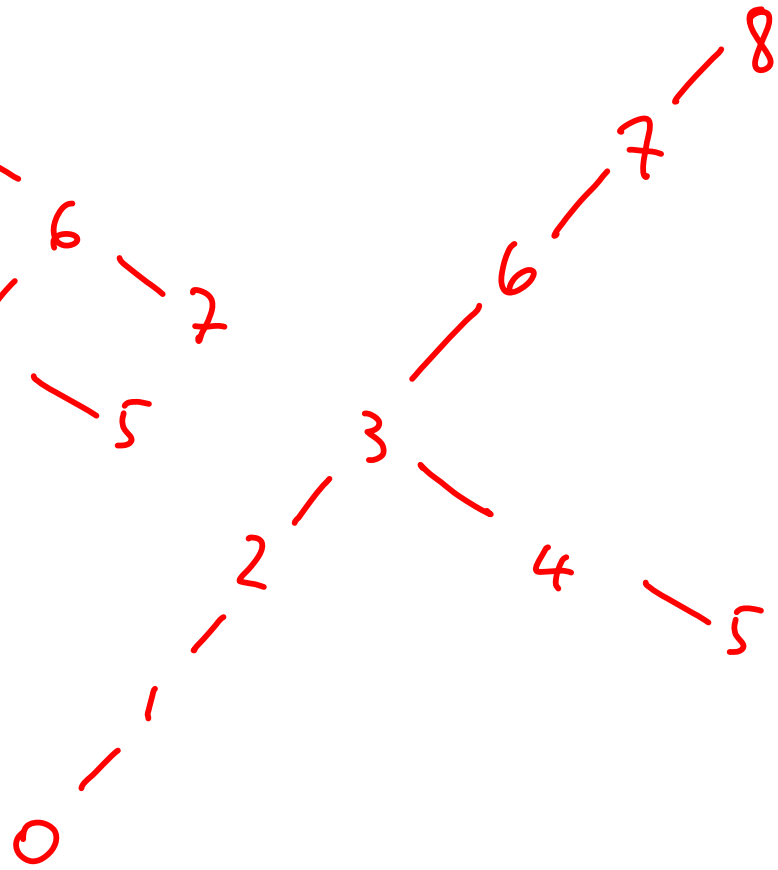
(b) Si descriva come modificare gli SPLAY TREE affinché possa essere gestita in maniera efficiente anche l'operazione SPLAY-L-D($T$) per la ricerca della minima chiave residente in un nodo di profondità massima.
In particolare, si descrivano un'implementazione della procedura SPLAY-L-D($T$) e le modifiche da apportare alle operazioni *zig-zag*, *zig-zig* e *zig*.
Qual è il costo ammortizzato dell'operazione SPLAY-L-D($T$) ? Perché?

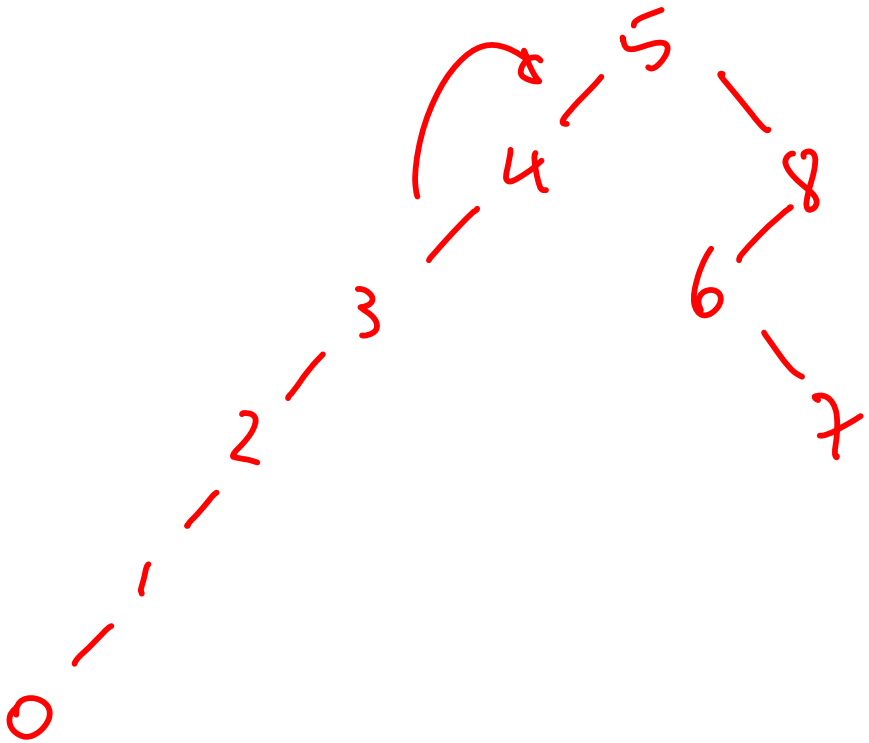


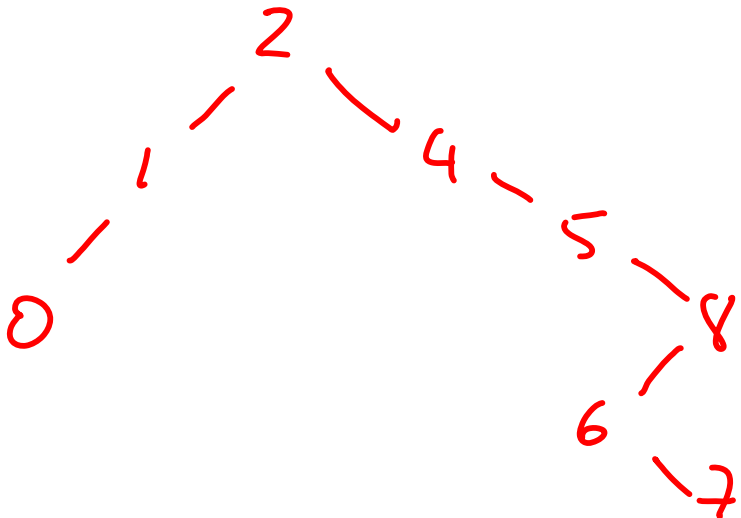



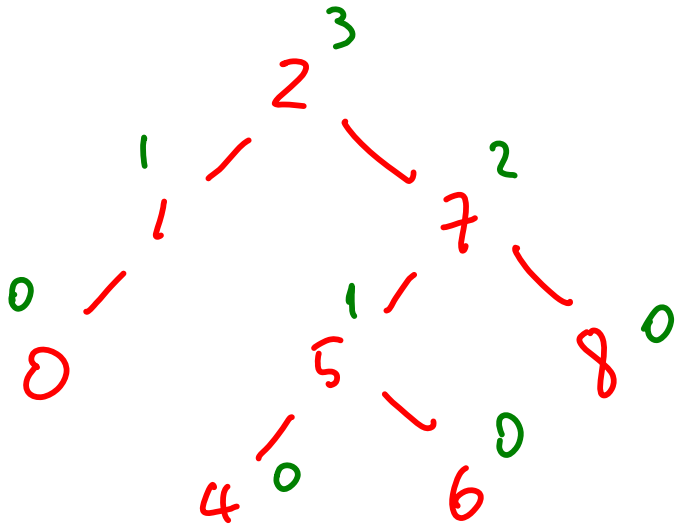

7
6
3
2
4
1
5
0

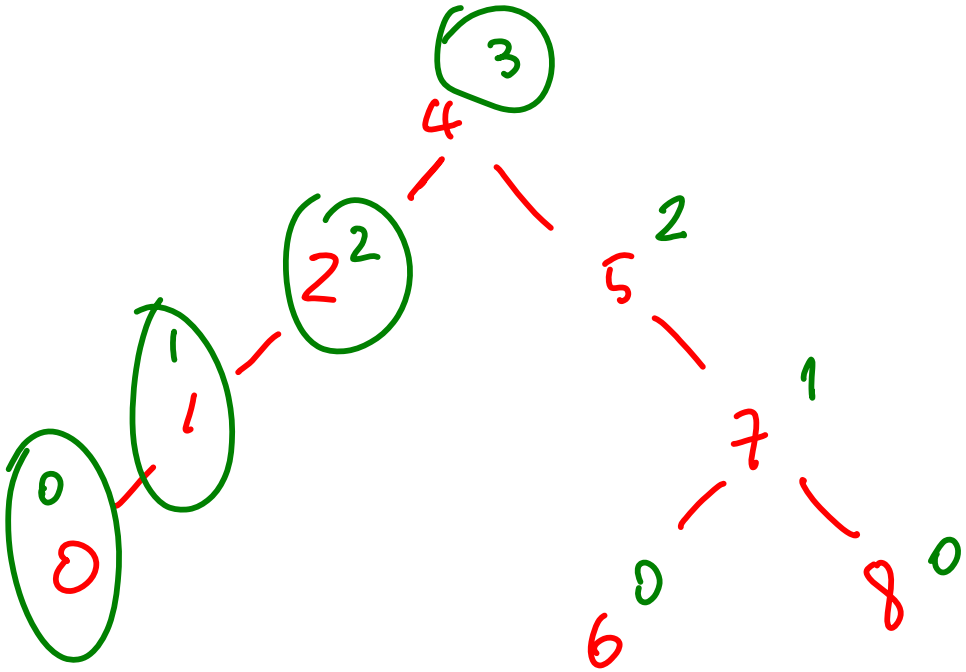

3
6
2
7
4
1
5
0


8
7
6
3
2
4
1
5
0


5
4
8
3
6
2
7
1
0


2
1
4
0
5
8
6
7


3
2
1
1
2
7
0
0
1
5
8
0
4
0
6
0


3
4
2
2
5
1
1
7
0
0
0
6
8
0