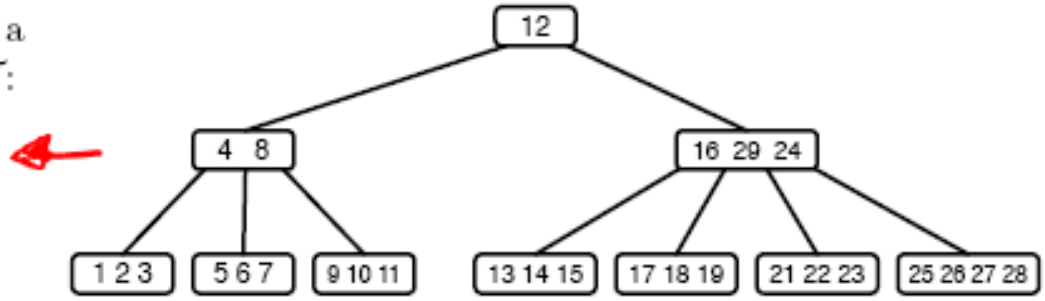
Utilizzando il metodo dell'**aggregazione** e quello del **potenziale**, si determini il costo ammortizzato per operazione di una sequenza di n operazioni, ove il costo c_i dell'i-esima operazione sia dato da

$$c_i = \begin{cases} 2 \cdot i & \text{se } i \text{ è potenza esatta di 6} \\ \frac{3}{5} & \text{altrimenti}\,. \end{cases}$$

(a) Si **definisca** la struttura dati dei B-tree.

(b) Dopo aver **determinato** il grado minimo del B-tree $\mathcal{T}$ a lato si **illustri** l'esecuzione delle seguenti operazioni su $\mathcal{T}$:

(1) DELETE(1)
(2) DELETE(17)
(3) DELETE(5)
(4) DELETE(21)
(5) DELETE(9)
(6) DELETE(25)
(7) DELETE(2)

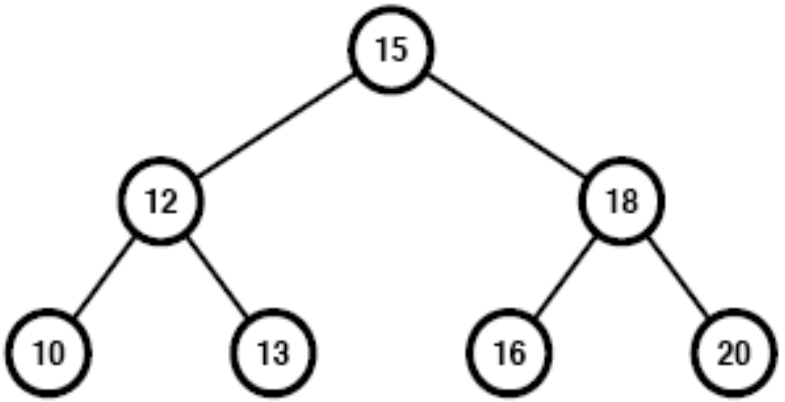


(c) Sia $\mathcal{T}'$ un B-tree con 4500 chiavi, il cui grado minimo è il medesimo di quello in figura. Qual è la massima altezza possibile per $\mathcal{T}'$?

ESERCIZIO 3

Si **definiscano** gli heap binomiali. Quindi si ~~spieghino~~, anche avvalendosi di opportuni esempi, le procedure per l'inserimento di una chiave e per l'estrazione del~~le chiave~~ minim~~o~~, confrontandole con le analoghe ~~operazioni~~ relative agli heap di Fibonacci.

(a) Si **descrivano** le operazioni di *zig-zag*, *zig-zig* e *zig* in uno splay tree di tipo bottom-up. Quindi si **eseguano** nell'ordine dato le seguenti operazioni sullo splay tree a lato:
- SEARCH(16)
- INSERT(11)
- DELETE(13)
- SEARCH(20)

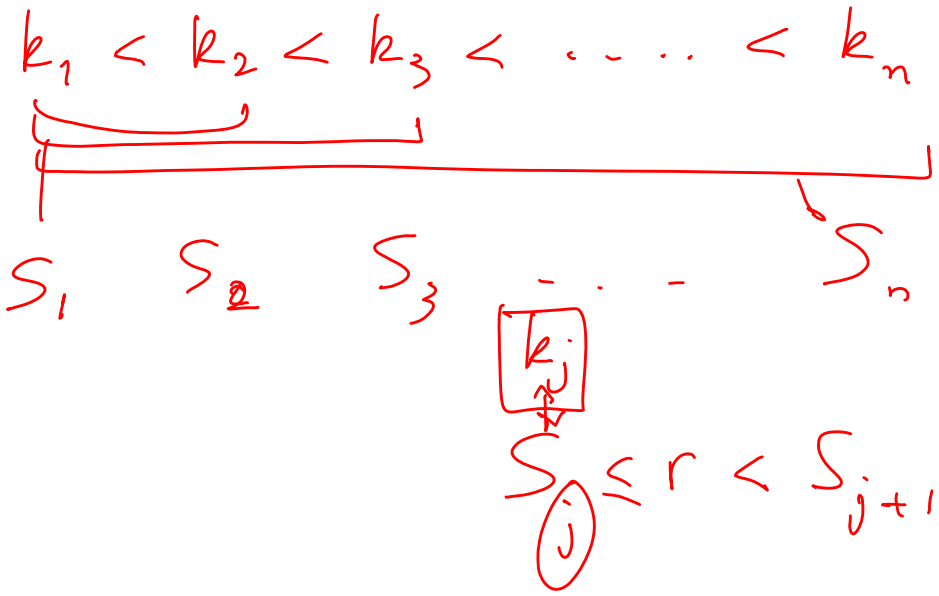


(b) Dopo aver **descritto** le operazioni di *zig-zag*, *zig-zig* e *zig*, nonchè l'operazione di assemblaggio finale in uno splay tree di tipo top-down, si eseguano le medesime operazioni sullo splay tree in figura (nella variante top-down).

(b) Sia $\mathcal{T}$ uno splay tree non vuoto le cui chiavi siano numeri interi a due a due distinti.

Si **descriva** come modificare gli SPLAY TREE affinché possa essere gestita in maniera efficiente anche l'operazione SPLAY-SUM$(\mathcal{T}, r)$, con $r \geq \min \mathcal{T}$, per la ricerca della massima chiave k in $\mathcal{T}$ tale che, detta S la somma di tutte le chiavi in $\mathcal{T}$ minori o uguali a k, si abbia $S \leq r$.

Qual è il costo ammortizzato dell'operazione SPLAY-SUM$(\mathcal{T}, r)$? Perché?



In aggiunta ai campi usuali, ciascun nodo ha anche un campo lsum che mantiene la somma delle chiavi residenti nei nodi del sottoalbero sinistro.
Se ad un nodo manca il figlio sinistro, lsum risulterà indefinito/NIL

Un prototipo della funzione SPLAY_SUM (root, r) è nel lucido seguente, ove viene utilizzata la seguente espressione a ? b con la seguente semantica:

if a è definito then a else b fi

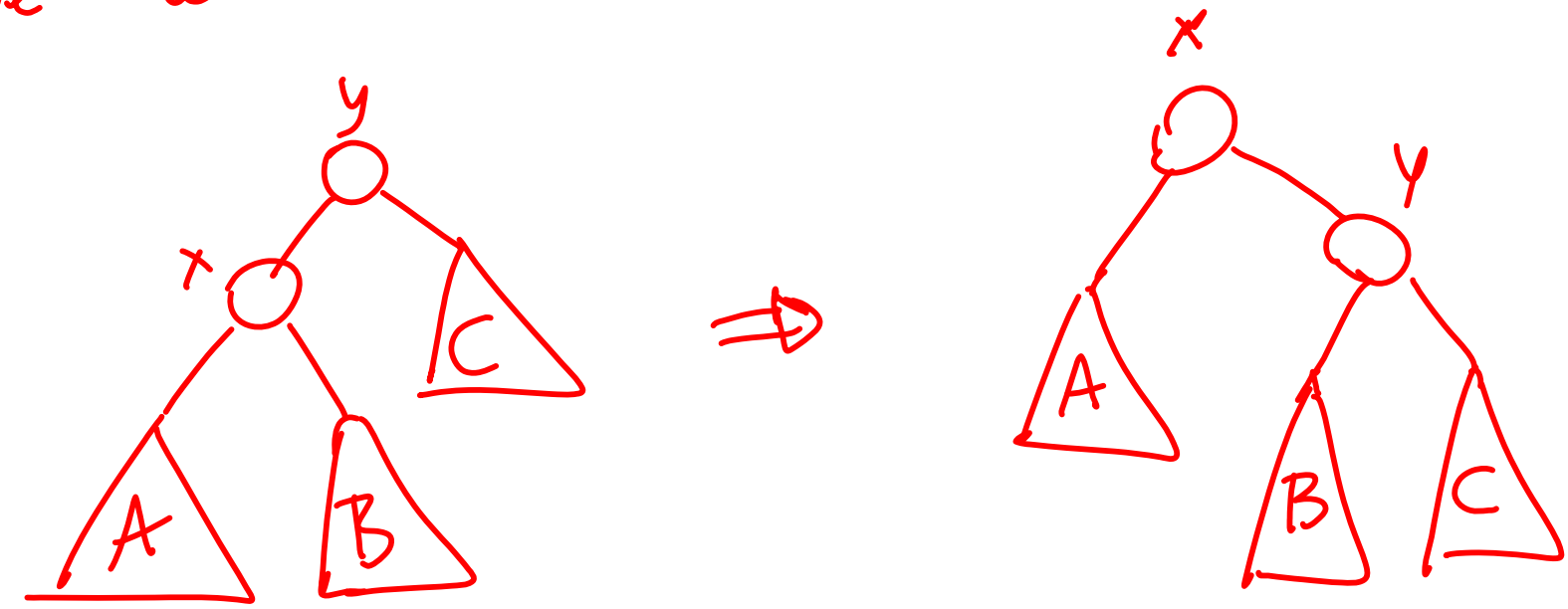
```
SPLAY-SUM(root, r)
  if   root = NIL   then
       return NIL

  elseif  lsum[root]?0 + key[root] = r  then
          return root

  elseif  lsum[root)?0  + key[root) > r  then
          return SPLAY_SUM(left[root], r)

  else
          return SPLAY_SUM(right[root],
                           r - lsum[root)?0 - key[root)) ? root
```

Il campo lsum può essere facilmente mantenuto durante le rotazioni.



$$lsum[y] := lsum[y] - lsum[x] - key[x]$$

Esso può essere facilmente aggiornato durante un inserimento o una cancellazione