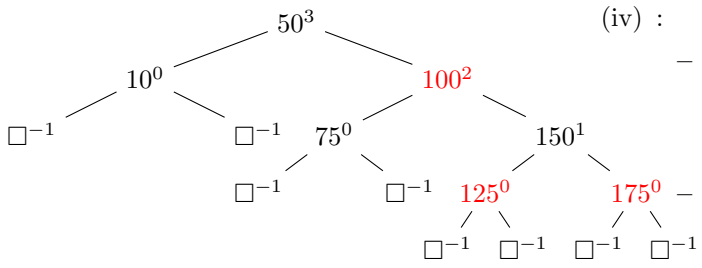
Problem 3: Rot-Schwarz Bäume

Ein rot-schwarz Baum ist ein binärer Suchbaum, den wir auf die folgende Weise erweitern: Jeder Knoten und jeder leere Teilbaum erhält eine Farbe (rot oder schwarz), so dass die folgenden Regeln gelten: (i) die Wurzel ist schwarz; (ii) die leeren Teilbäume sind schwarz; (iii) die Kinder eines roten Knoten sind schwarz; und (iv) die schwarze Tiefe aller leeren Teilbäume ist gleich, d.h., für alle leeren Teilbäume ist die Anzahl der schwarzen Knoten auf dem Pfad von der Wurzel zum jeweiligen Teilbaum gleich.

a) Zeichnen Sie drei Beispiele für rot-schwarz Bäume und erklären Sie, warum diese jeweils die Regeln für einen rot-schwarz Baum erfüllen.

1. Beispiel



(i) Wurzel ist schwarz

(ii) alle leeren Teilbäume sind schwarz

(iii) Kinderknoten von 100 sind schwarz(75,150)

(iv) :

- linker Teilbaum(10): von der Wurzel bis zu den leeren Teilbäumen haben wir 1 schwarzen Knoten (10)
- rechter Teilbaum(100-150-175): von der Wurzel bis zu den leeren Teilbaum haben wir 1 schwarzen Knoten (150)

2. Beispiel

3. Beispiel

b) Sei T ein rot-schwarz Baum, und sei s die schwarze Tiefe der leeren Teilbäume. Zeigen Sie, dass T mindestens $2s-1-1$ schwarze Knoten besitzt. Was folgt daraus über die Mindestanzahl von Knoten in einem rot-schwarz Baum mit Höhe h? Folgern Sie: ein rot-schwarz Baum mit n Knoten hat Höhe $O(logn)$.

Annahme: Anzahl der Schwarzen Knoten in einem red-black-tree: $2^{s-1}-1$, dabei gelten (i)-(iv) aus der Aufgabenstellung.
base-case: red-black-tree, $s=2$ (leerer Baum mit Wurzel):

```
  r    | depth: 1 (root is black)
 / \
bn bn  | depth: 2 (black null leaves)
```

nach (i) gilt: $Num_{blackNodes}=1$
$Num_{blackNodes}=2^{2-1}-1=2^0-1=2-1=1$

Die Annahme gilt im Base-case, Voraussetzung: die Formel $2^{s-1}-1$ gilt für red-black-trees mit $s=k$ schwarzer Tiefe.

I.S red-black-tree, mit $s=k+1$:
nach (i)-(iv) gilt:

- Die Wurzel ist Schwarz
- Jeder Teilbaum der Wurzel hat eine schwarze Tiefe von k
- Jeder Teilbaum hat nach Annahme mindestens $2^{k-1}-1$ schwarze Knoten

$Num_{blackNodes} \geq 1+2(2^{k-1}-1)=1+2^1*2^{k-1}-2=2^k-1 \mid k=s-1$
$Num_{blackNode} \geq 2^{s-1}-1$

Die Formel aus der Annahme gilt auch im Induktionsschritt.

Laut Definition ist die schwarze Tiefe die Anzahl der Schwarzen Knoten von der Wurzel, bis einem Blatt, wobei Blätter ebenfalls schwarz sind.
d.h für die Anzahl schwarzer Knoten im einem red-black-tree gilt: $Num_{blackNode} \geq h/2$, wobei h die Höhe des gesamten Baumes ist.
daraus folgt: $Num_{blackNodes} \leq 2^{h/2-1}-1$ — nach h umformen:
$h \geq 2(log(Num_{blackNodes}+1)+1) \rightarrow h \geq 2(log(n+1)+1)$ die Dominate Komplexitätsklasse ist $O((logn))$
daher gilt für die Höhe eines red-black-tree $h <= O(log(n))$

Quellen:
https://medium.com/data-science/understanding-time-complexity-with-python-examples-2bda6e8158a7
https://www.geeksforgeeks.org/introduction-to-red-black-tree/#interesting-points-about-redblack-t