I217E: Functional Programming

Nao Hirokawa

JAIST

Term 2-1, 2022

`http://www.jaist.ac.jp/~hirokawa/lectures/fp/`

Schedule

10/12	introduction	11/9	interpreters
10/14	algebraic data types I	11/11	compilers
10/19	algebraic data types II	11/16	termination
10/21	applications	11/18	confluence
10/26	program reasoning	11/25	verification
10/28	data structures I	11/30	review
11/2	data structures II	12/5	exam
11/4	computational models		

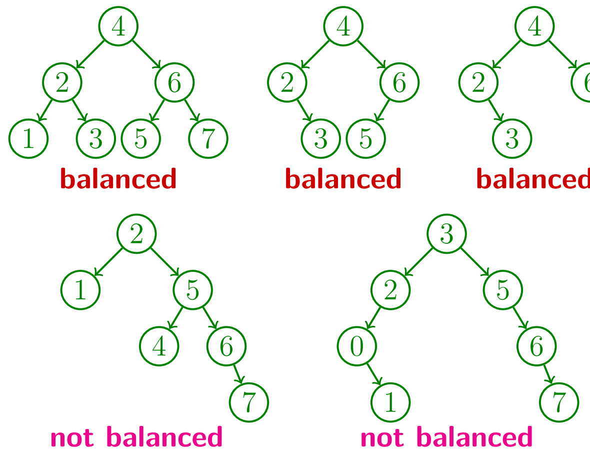
Evaluation

exam (60) + reports (40)

AVL Trees

(Adelson, Velski, and Landis 1962)

Balanced or Not?



Balanced Trees

Definition

tree t is **balanced** if $|\texttt{slope}\ t| \leqslant 1$ for all subtrees t, where

$$\begin{aligned}
\texttt{depth}\ \text{Leaf} &= 0 \\
\texttt{depth}\ (\text{Node}\ \ell\ x\ r) &= 1 + \max\{\texttt{depth}\ \ell, \texttt{depth}\ r\} \\
\texttt{slope}\ \text{Leaf} &= 0 \\
\texttt{slope}\ (\text{Node}\ \ell\ x\ r) &= \texttt{depth}\ \ell - \texttt{depth}\ r
\end{aligned}$$

Observation

for every balanced tree t

$$\begin{aligned}
|\texttt{slope}\ (\text{add}\ x\ t)| &\leqslant 2 \\
|\texttt{slope}\ (\text{delete}\ x\ t)| &\leqslant 2
\end{aligned}$$

Idea: Modify and then Rebalance

Definition

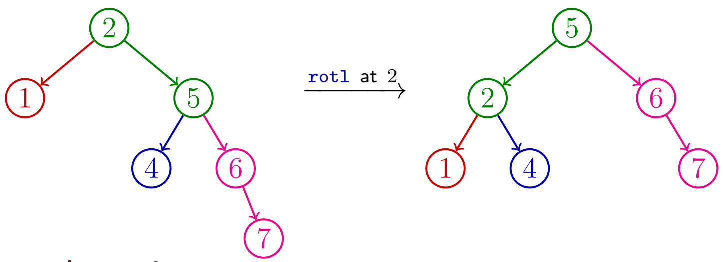
$$\begin{aligned}
\texttt{add}\ x\ \text{Leaf} &= \text{Node Leaf}\ x\ \text{Leaf} \\
\texttt{add}\ x\ (\text{Node}\ \ell\ y\ r) &= \\
&\begin{cases}
\text{rebalance}\ (\text{Node}\ (\texttt{add}\ x\ \ell)\ y\ r) & \text{if } x < y \\
\text{rebalance}\ (\text{Node}\ \ell\ y\ (\texttt{add}\ x\ r)) & \text{if } x > y \\
\text{Node}\ \ell\ y\ r & \text{otherwise}
\end{cases}
\end{aligned}$$

Challenge

implement **constant time** rebalance function

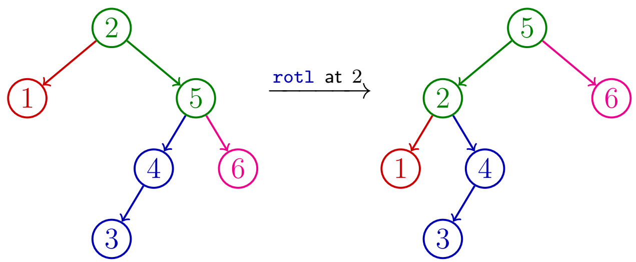
Rebalance by Rotation (1)

$$\begin{aligned}
\texttt{rotl}\ (\text{Node}\ \ell\ x\ (\text{Node}\ m\ y\ r)) &= \text{Node}\ (\text{Node}\ \ell\ x\ m)\ y\ r \\
\texttt{rotr}\ (\text{Node}\ (\text{Node}\ \ell\ x\ m)\ y\ r) &= \text{Node}\ \ell\ x\ (\text{Node}\ m\ y\ r)
\end{aligned}$$

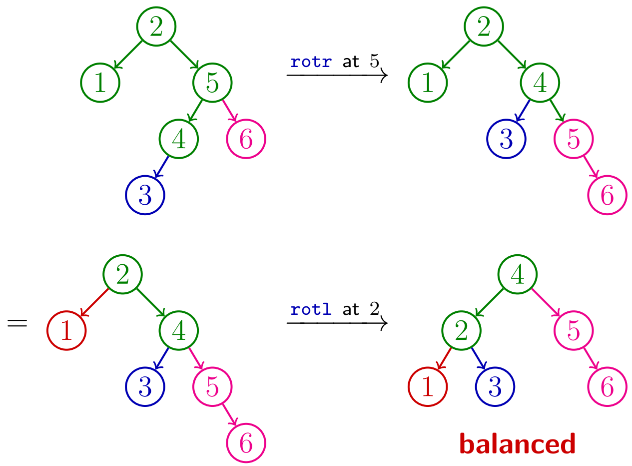


slope -2 **balanced**

Rebalance by Rotation (2)



slope -2 **not balanced**



Constant Time Rebalancing Function (?)

Definition

$$\texttt{rotr}\ (\text{Node}\ (\text{Node}\ \ell\ x\ m)\ y\ r) = \text{Node}\ \ell\ x\ (\text{Node}\ m\ y\ r)$$

$$\text{shiftr}\ (\text{Node}\ \ell\ x\ r) = \begin{cases} \texttt{rotr}\ (\text{Node}\ (\texttt{rotl}\ \ell)\ x\ r) & \text{if } \texttt{slope}\ \ell = -1 \\ \texttt{rotr}\ (\text{Node}\ \ell\ x\ r) & \text{otherwise} \end{cases}$$

$$\text{rebalance}\ t = \begin{cases} \text{shiftr}\ t & \text{if } \texttt{slope}\ t = 2 \\ \text{shiftl}\ t & \text{if } \texttt{slope}\ t = -2 \\ t & \text{otherwise} \end{cases}$$

Complexities of AVL Trees

modify definition of trees to keep depth d in nodes:

$$\begin{aligned} &\text{data}\ \ \texttt{AVLTree}\ a = \\ &\quad \text{Leaf}\ \mid\ \text{Node}\ (\text{Tree}\ a)\ \text{Int}\ \text{Int}\ (\text{Tree}\ a) \end{aligned}$$

$$\begin{aligned} \texttt{depth}\ \text{Leaf} &= 0 \\ \texttt{depth}\ (\text{Node}\ \ell\ x\ d\ r) &= d \end{aligned}$$

Complexities

for AVL trees of size n

- slope and rebalance cost $O(1)$
- member, add, and delete cost $O(\log n)$

Lazy Evaluation

Exercises

define next functions:

1. $\texttt{from}\ n = n : n+1 : n+2 : \cdots$ same as $[n\,..]$

2. $\texttt{take}\ n\ (x_1 : x_2 : \cdots) = [x_1, x_2, \ldots, x_n]$

use Haskell to evaluate next terms:

1. $\texttt{take}\ 2\ (\texttt{from}\ 10)$

2. $\texttt{zip}\ [1\,..]\ [10, 20, 30]$

Eager Evalution

Terminology

eager evalution evaluates arguments of function and then compute the function

Example

computation in eager evaluation

$$\begin{aligned}\texttt{take}\ 2\ (\underline{\texttt{from}\ 10}) &= \texttt{take}\ 2\ (10 : \underline{\texttt{from}\ 11})\\ &= \texttt{take}\ 2\ (10 : 11 : \underline{\texttt{from}\ 12})\\ &= \texttt{take}\ 2\ (10 : 11 : 12 : \underline{\texttt{from}\ 13})\\ &= \cdots \quad \text{(non-terminating)}\end{aligned}$$

Lazy Evaluation which Haskell Adopts

Terminology

lazy evalution evaluates argument of function only if it is necessary to compute

Example

computation in lazy evaluation

$$\begin{aligned}\texttt{take}\ 2\ (\underline{\texttt{from}\ 10}) &= \underline{\texttt{take}\ 2\ (10 : \texttt{from}\ 11)}\\ &= 10 : \texttt{take}\ 1\ (\underline{\texttt{from}\ 11})\\ &= 10 : \underline{\texttt{take}\ 1\ (11 : \texttt{from}\ 12)}\\ &= 10 : 11 : \underline{\texttt{take}\ 0\ (\texttt{from}\ 12)} = 10 : 11 : [\,]\end{aligned}$$

Homework: Sieve of Eratosthenes

2	3	4	5	6	7	8	9	10	11	$\cdots$
2	3	–	5	–	7	–	9	–	11	$\cdots$
2	**3**	–	5	–	7	–	–	–	11	$\cdots$
2	**3**	–	**5**	–	7	–	–	–	11	$\cdots$

Homework

implement infinite list $\texttt{primes}$ of prime numbers. for instance,

$$\texttt{take}\ 4\ \texttt{primes} = [2, 3, 5, 7]$$

Homework

1. Implement `rebalance` for AVL trees.

2. Implement `add` for AVL trees.

3. Implement `delete` for AVL trees.

4. Implement `primes`.