Tutorial 10

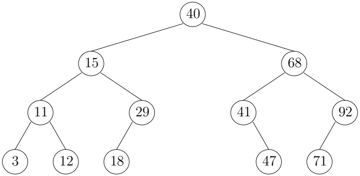
Tutorial 10

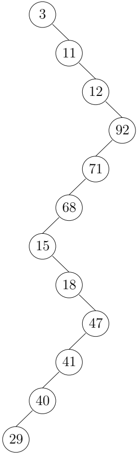
CSCI2100 Teaching Team

Department of Computer Science and Engineering
The Chinese University of Hong Kong

Binary Search Tree Example

Two possible BSTs on $S = \{3, 11, 12, 15, 18, 29, 40, 41, 47, 68, 71, 92\}$:



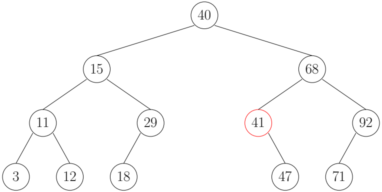


Predecessor Query

Let S be a set of integers. A predecessor query for a given integer q is to find its predecessor in S, which is the largest integer in S that does not exceed q.

Example

Suppose that $S = \{3, 11, 12, 15, 18, 29, 40, 41, 47, 68, 71, 92\}$ and we have a balanced BST T on S:



We want to find the predecessor of $q = 42$ in S.

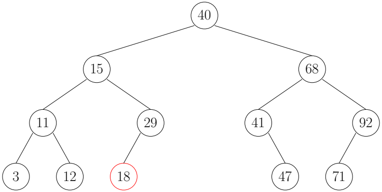
Nodes accessed: 40, 68, 41, and 47.

Successor Query

Let S be a set of integers. A successor query for a given integer q is to find its successor in S, which is the smallest integer in S that is no smaller than q.

Example

We want to find the successor of $q = 17$ in S.



Nodes accessed: 40, 15, 29, and 18.

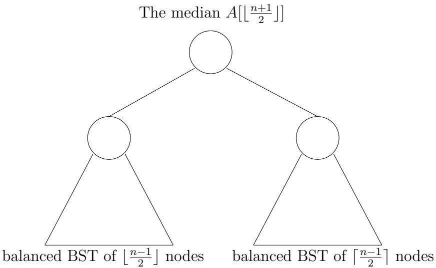
Construction of a Balanced BST

In the following, we will discuss how to construct a balanced BST T on a sorted set S of n integers in $O(n)$ time.

Construction of a Balanced BST

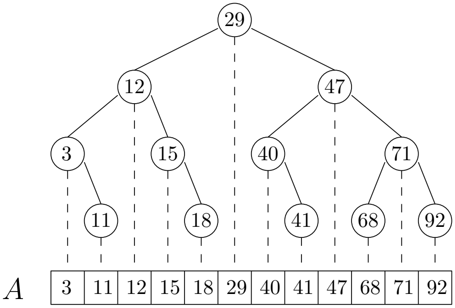
Assume that S is stored in an array A and A is sorted.

- **Observation:** The subtree of any node in a balanced BST is also a balanced BST.
- **Main idea:**



Example

Let us construct a balanced BST T on the following sorted array A.



Construction of a Balanced BST

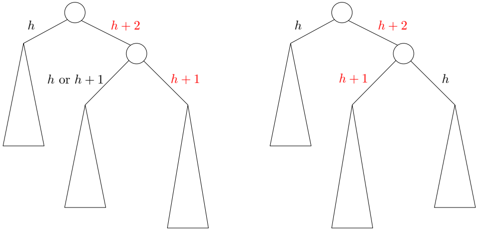
Let $f(n)$ be the maximum running time for constructing a balanced BST from an array of length n. We have:

$$
\begin{aligned}
f(1) &= O(1) \\
f(n) &= O(1) + 2 \cdot f(\lceil n/2 \rceil)
\end{aligned}
$$

Solving the recurrence gives $f(n) = O(n)$.

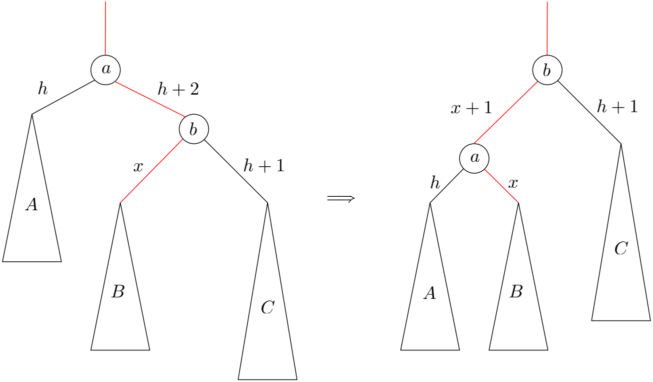
Rebalancing

In lectures we explored the Left-Left and Left-Right cases in detail, so here we will look at Right-Right and Right-Left:



Right-Right

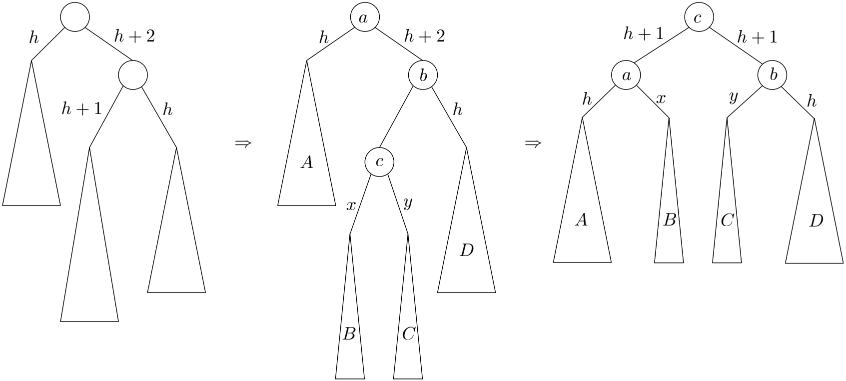
Fix by a rotation (symmetric to left-left):



Note that $x = h$ or $h + 1$, and the ordering from left to right of A, a, B, b, C is preserved after rotation.

Right-Left

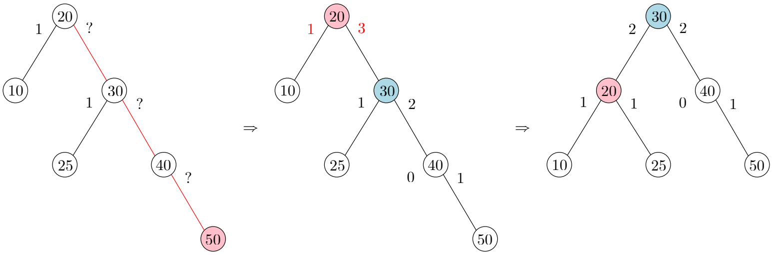
Fix by a double rotation (symmetric to left-right):



Note that x and y must be h or $h - 1$. Futhermore at least one of them must be h.

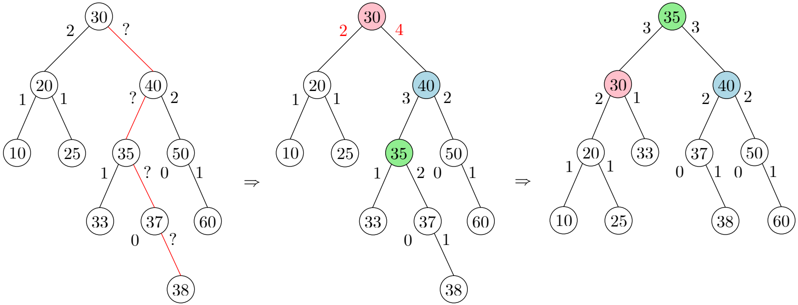
Right-Right Example

Inserting 50:



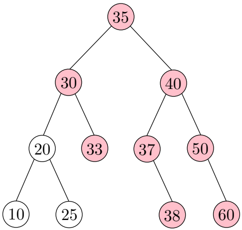
Right-Left Example

Inserting 38:



Range Reporting

Let S be a set of n integers. Given an interval $[q, \infty)$, a range query reports all the integers of S that fall in $[q, \infty)$. Describe an algorithm to use a balanced BST on S to answer a query in $O(\log n + k)$, where k is the number of integers reported.

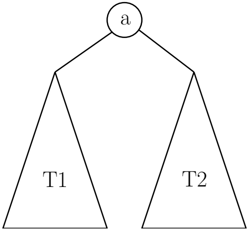


For the query $[27, +\infty)$, we need to report the integers in pink.

Range Reporting

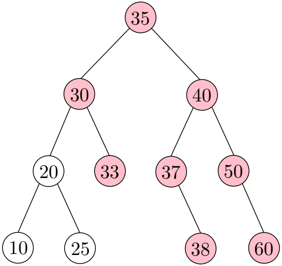
To answer a query $[q, \infty)$, we do the following at the root:

- If $a < q$, recursively report the integers in T_2 that fall in $[q, \infty)$.
- If $a = q$, report a and all the integers in T_2.
- If $a > q$, report a and all the integers in T_2. After that, recursively report the integers in T_1 that fall in $[q, \infty)$.



Range Reporting

The tutor will explain the algorithm using $[27, +\infty)$ as the example query.



In each level of the recursion, we do the following:

- Compare q to the integer stored in the root, the cost of which is $O(1)$.
- (If necessary) report all the integers in the right subtree, the cost of which is proportional to the number of integers in the right subtree.

As the height of the BST is $O(\log n)$, the first bullet costs $O(\log n)$ in total. The second step reports integers from disjoint subtrees and, therefore, incurs cost $O(k)$ in total. The overall cost is $O(\log n + k)$