Binary Search Trees

335 Fall 2022 – Recitation 8

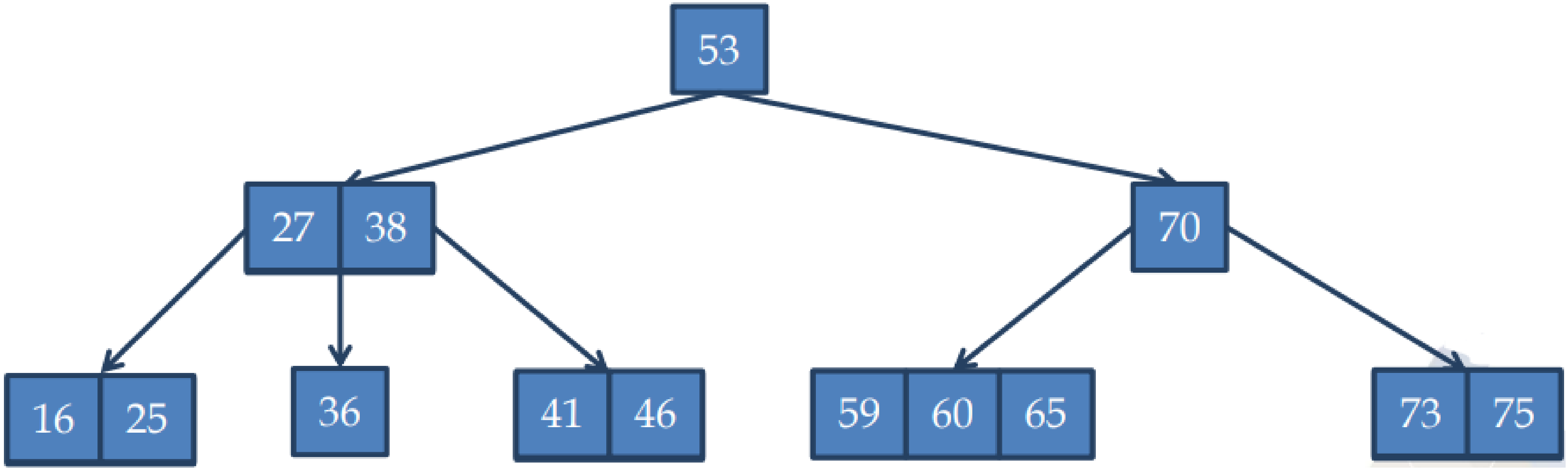
First: Some useful links

- https://www.cs.usfca.edu/~galles/visualization/Algorithms.html
  - https://www.cs.usfca.edu/~galles/visualization/RedBlack.html
  - https://www.cs.usfca.edu/~galles/visualization/BTree.html

Q1-) 2-3-4 Tree

- Insert the following sequence of numbers into a 2-3-4 tree:

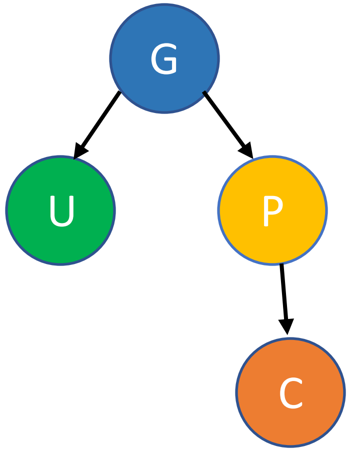
53, 27, 75, 25, 70, 41, 38, 16, 59, 36, 73, 65, 60, 46



Remainders About Red-Black Trees

- The root and NILs are always black
- **Black condition**: every path from the root to a leaf node has the same number of black nodes
- **Red condition**: every red node has a black parent and black children

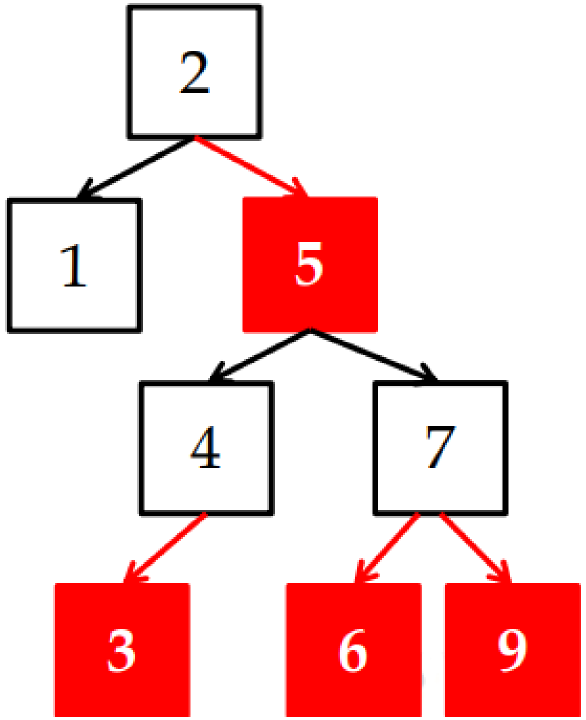
- Case 1: Uncle is red -> Recolor
- Case 2: Uncle is black:
  - Case 2-a: grandparent-parent-child forms a line -> Rotate & Recolor
  - Case 2-b: grandparent-parent-child forms a triangle-> Transform it to case 2-a



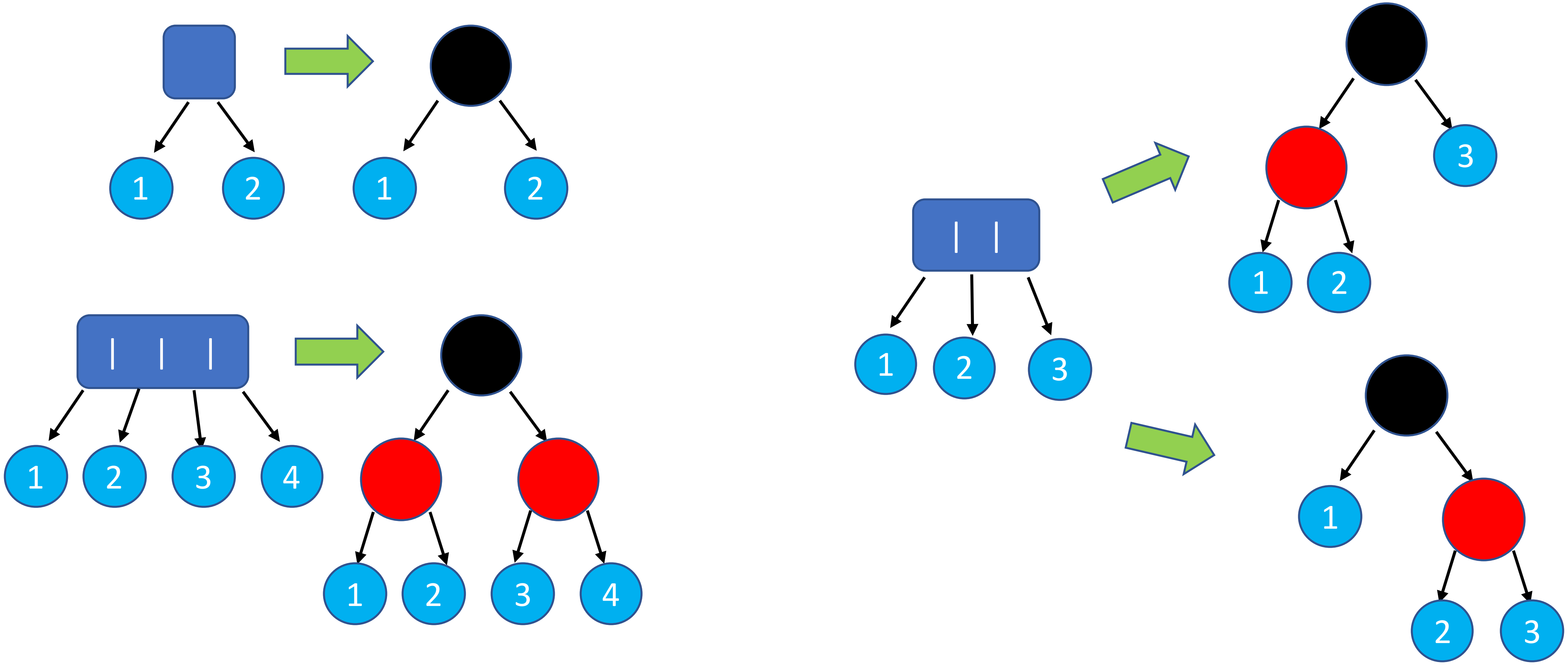
Q2-) Red-Black Tree

- Insert the following sequence of numbers into a red-black tree:

2, 1, 4, 5, 9, 3, 6, 7

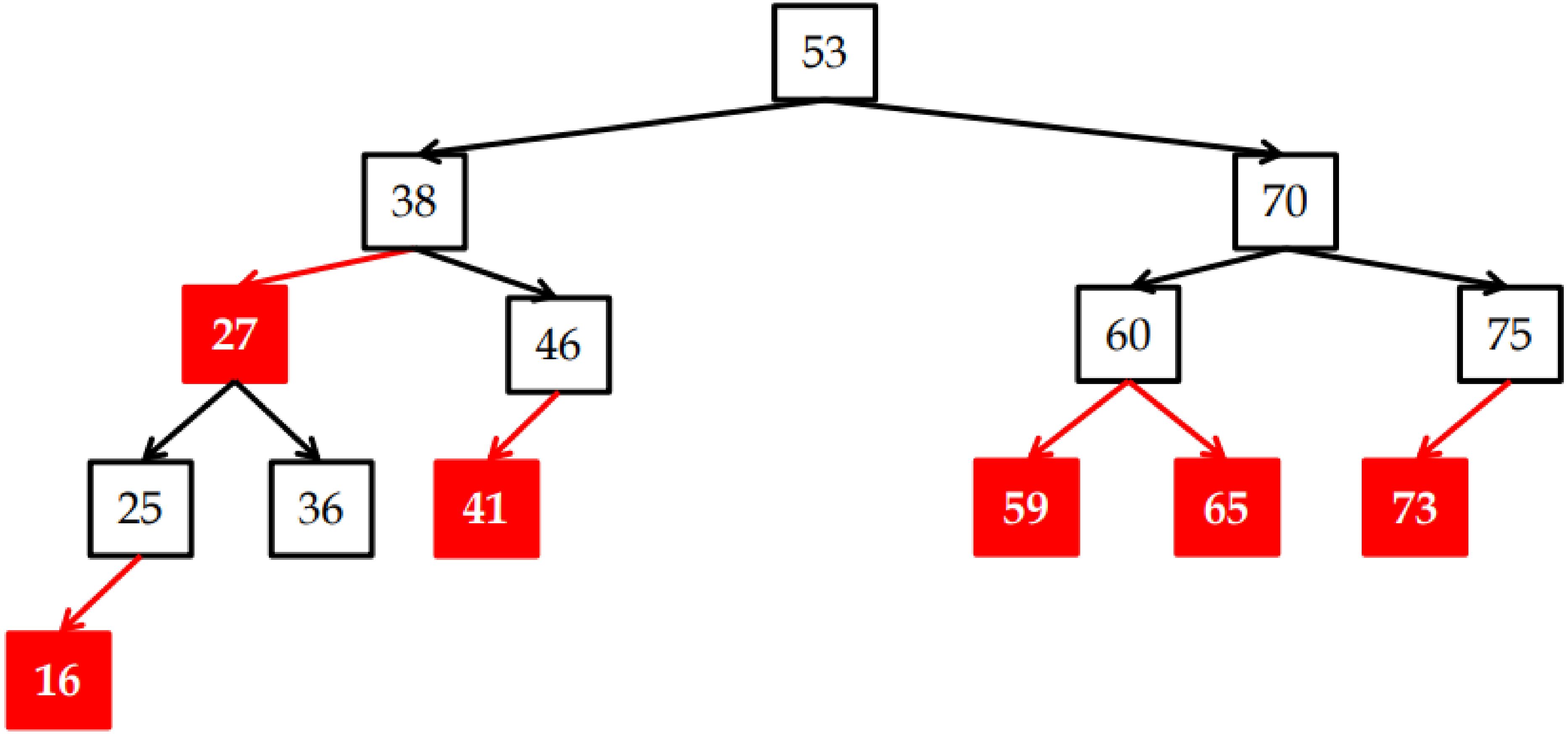


Remainder About Conversions



Q3-) Convert 2-3-4 Tree to Red-Black Tree

- Convert the result tree from the q2 to red-black tree



Q4-) General Questions

1. What is the difference between the binary-search-tree property and the min-heap property? Can the min-heap property be used to print out the keys of an n-node tree in sorted order in O(n) time?

2. Suppose that we have numbers between 1 and 1000 in a binary search tree, and we want to search for the number 363. Could 925, 202, 911, 240, 912, 245, 363 search sequence occur?

3. What is the largest possible number of internal nodes in a red-black tree with black height k? What is the smallest possible number? (exclude the NIL nodes)

Q4 Answers

1. In the min-heap, children's keys are always smaller than the parents. In the binary search tree, keys are sorted such that the left child is smaller than the parent node, and the right child is greater. We need to use lg(n) operations to sort a single element in a heap. Which gives O(n lg n) complexity for sorting all elements which is the running time of heap sort.
2. The traversal sequence is impossible because, we check with 911 in the 3rd step, and then traverse towards to left sub-tree of 911 while looking for a value smaller (363). Then in the 5th step, there is 912 but there is no possibility to encounter 912 at the left subtree since 912 > 911 (from the 3rd step).
3. For the largest case, let's say nodes are ordered as B – R – B – R … B for all branches. Then the height of the tree would be 2k which results in 2^{2k}-1 nodes. For the smallest case, let's say nodes are ordered as B – B – B – B… B for all branches. Then the height of the tree would be k (excluding the NIL nodes) which results in 2^k-1 nodes.