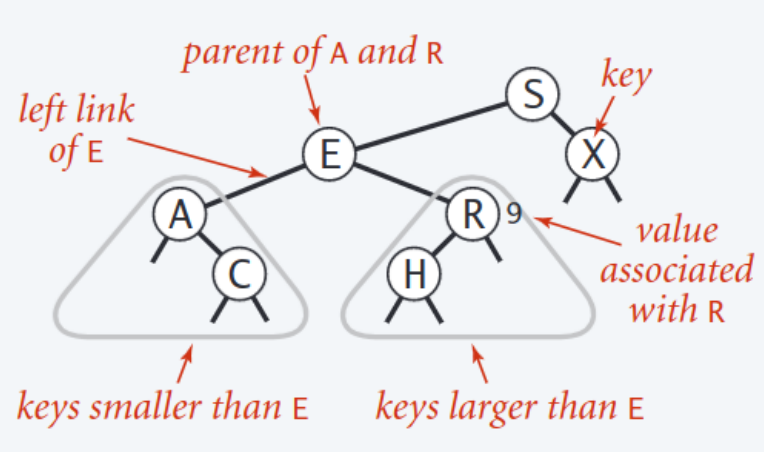
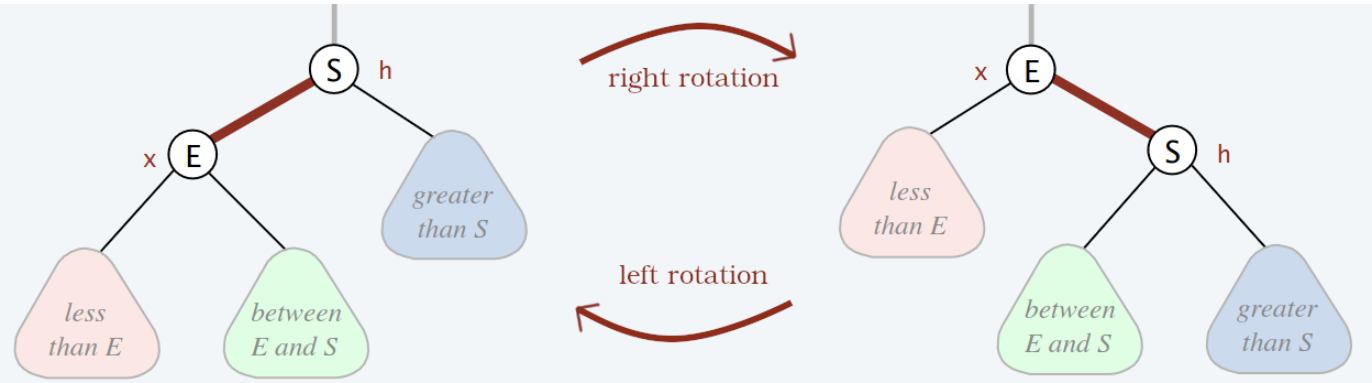
Precept Outline
- Review of Lectures 9 and 10:
  - Binary Search Trees
  - Balanced Binary Search Trees
- Midterm Review

Relevant Book Sections
- Book chapters: 3.1, 3.2 and 3.3

A. Review: Binary Search Trees and Red-Black Trees

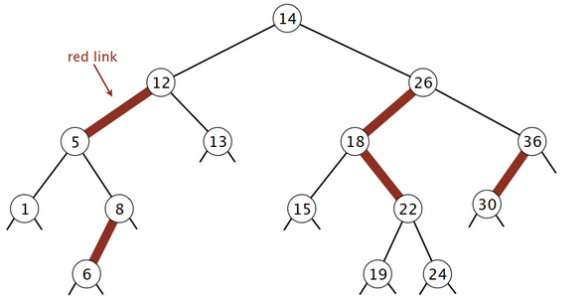
Your preceptor will briefly review key points of this week's lectures. Here are some images reminding you of some of the key definitions from lecture.





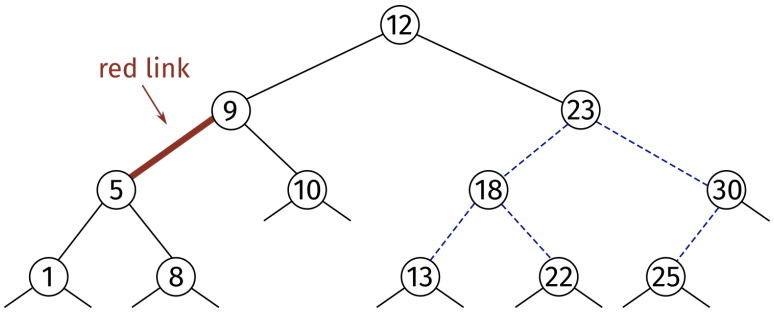
B. Red-Black Trees (Spring'23 Midterm)

The following binary search tree satisfies perfect black balance, but violates color invariants:



Give a sequence of 4 elementary operations (*color flip, rotate left or rotate right*) that restore the color invariants.

Consider the following left-leaning red-black BST with some of the edge colors suppressed:



Which keys in the above tree are red? (Recall that a key is red if the link to its parent is red.)

C. Sorting Compares (Fall'17 Midterm)

Consider an array that contains two successive copies of the integers 1 through n, in ascending order. For example, here is the array when $n = 8$:

$$1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8 \mid 1 \quad 2 \quad 3 \quad 4 \quad 5 \quad 6 \quad 7 \quad 8$$

Note that the length of the array is $2n$, not n.

How many compares does *selection sort* make to sort the array as a function of n? Use tilde notation to simplify your answer.

How many compares does *insertion sort* make to sort the array as a function of n? Use tilde notation to simplify your answer.

How many compares does *mergesort* make to sort the array as a function of n? You may assume n is a power of 2. Use tilde notation to simplify your answer.

D. Algorithm Design

A length-n integer array `a[]` is *single-peaked* if there exists $0 \leq k \leq n-1$ such that the subarray from index up to (and including) k is strictly increasing, and the subarray from index k until $n-1$ is strictly decreasing. The entry `a[k]` is called the *peak*. For example, the array `a[] = {3, 6, 7, 10, 4, 1}` is a single-peaked with peak `10`, but the array `a[] = {3, 6, 7, 10, 4, 5}` is not.

Design an algorithm that receives as input a single-peaked array with n *distinct elements*, and outputs the peak of the array. *Specify the running time of your solution.*

Full credit: The running time of the algorithm must be $O(\log n)$. **Partial credit**: The running time of the algorithm must be $O(n)$.

In the space provided, give a concise English description of your algorithm for solving the problem. You may use any of the algorithms that we have considered in this course (e.g., lectures, precepts, textbook, assignments) as subroutines. If you modify such an algorithm, be sure to describe the modification. Feel free to use code or pseudocode to improve clarity.

E. Extra Practice

Part 1: Heaps

Consider the following binary heap representation of a maximum-oriented priority queue (with `pq[0]` unused).

pq[]	–	50	40	20	10	30	5
	0	1	2	3	4	5	6

Suppose that you *insert* the key 45 into the binary heap. Which keys would be involved in a *compare*? And which keys would be involved in an *exchange*?

Suppose that you perform a `delMax()` operation in the original binary heap. Which keys would be involved in a *compare*? And which keys would be involved in an *exchange*?

Part 2: Linear or Not?

Which of the following are $O(n)$?

( ) The number of compares needed to apply a `delMin()` operation in a minimum-oriented binary heap with n elements.
( ) The number of compares used by the best non-stable compare-based sorting algorithm that sorts n integers.
( ) The number of times a resizing array resizes in order to perform n operations in a row.
( ) The number of times `hello` is printed by the following code:

```
for (int i = 1; i <= n; i *= 2)
    for (int j = 1; j <= i; j++)
            StdOut.println("hello");
```