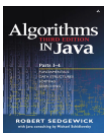
Week 2: references and exercises

- Recursive structures: list and tree
- Linked list / double-linked / circular
- Implementing operations with the linked list: by recursion and by iteration
- Java programming: generic classes, parametrized types, nested classes
- Terminology for tree structures: k-ary tree, height, level, depth. Implementation of a tree.
- Path of a tree: prefix / preorder, infix / in order, postfix / postorder, level order
- Syntactic tree. Conversions of arithmetic expressions: infix, postfix and prefix notations.

References

- 

[Sedgewick & Wayne 2011] §1.3
- 

[Sedgewick 2003] §3.3, §3.4, §4.3, §4.4, §4.6, §4.7, §5.4-5.7
- Java tutorial on generics

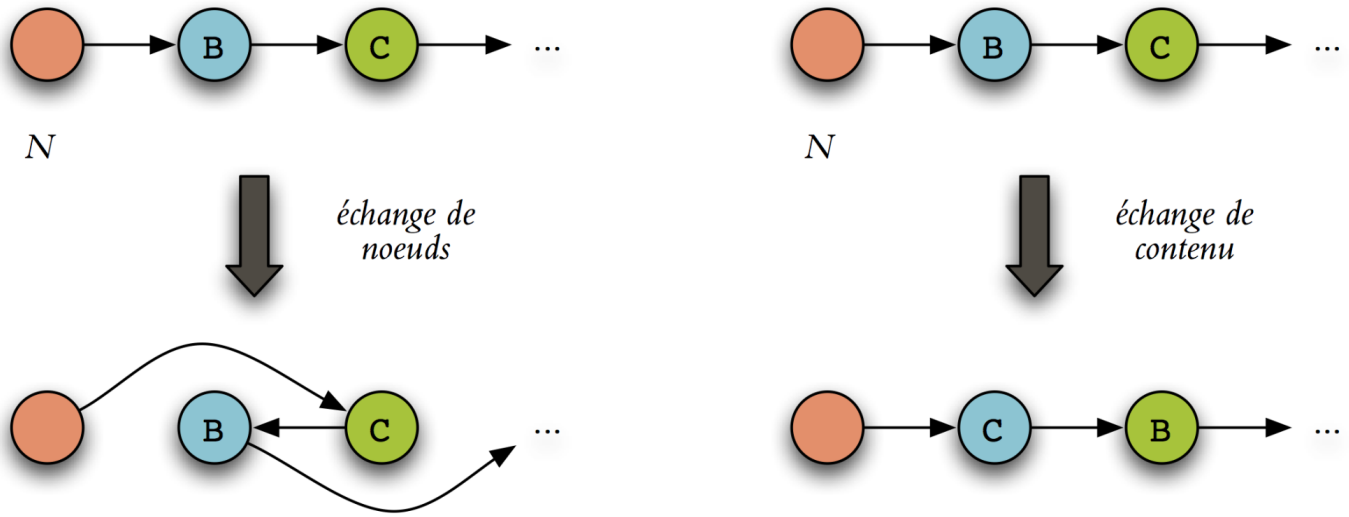
Examples of `java.util.LinkedList`

- openjdk6-b.14 To be observed: sentinel `header`, nested class `Entry<T>`, private methods `insertBefore(Entry)`and`remove`
- gnu classpath . Note: linear list with variables for header and tail (`first`and `last`), nested class, if-then-else in methods for insertion / deletion at ends (`addLastEntry`, `removeFirst`etc.)

1 Exercises: linked list

Unless otherwise specified, the following exercises represent a list as a set of nodes: each node `x` (except the terminal node `x = null`) contains the variables `x . next`(next element) and `x . data`(content: stored item).

1.1 Exchange of Elements



- Develop the code for `exchange( N )` that two nodes exchange following `N` .
- Develop the code for `exchangeData( N )` which exchanges the contents of both nodes following `N` .

1.2 Reversal of List

Propose algorithms to reverse a linked list

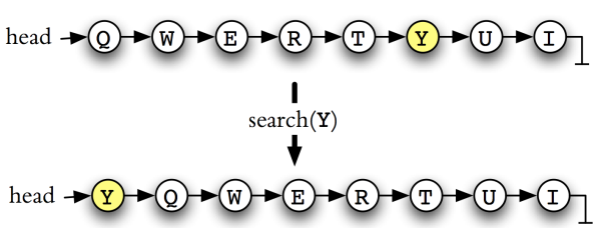
- by iteration (in a single route), or
- by recursion.

1.3 Circular list

- Show how to implement FIFO file interface methods with a circular list (keep a reference to the last node on the list).
- Show how to concatenate two circular lists.

1.4 Move to front

The MTF (*move-to-front*) heuristic moves the found item to the header. When an unsuccessful search is made, the list does not change. Heuristics are useful when looking for elements with different frequencies because the often searched elements are at the top of the list during a series of calls.



Give an operation `search`(head , `y`) that performs the sequential search according to the MTF heuristic.

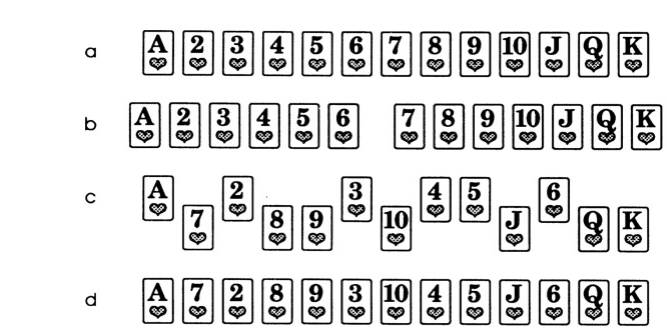
1.5 How to beat the cards?

The goal of beating the cards is to get a random order in the package. Mathematically, we want a random permutation uniformly distributed. The following algorithm constructs such a permutation.

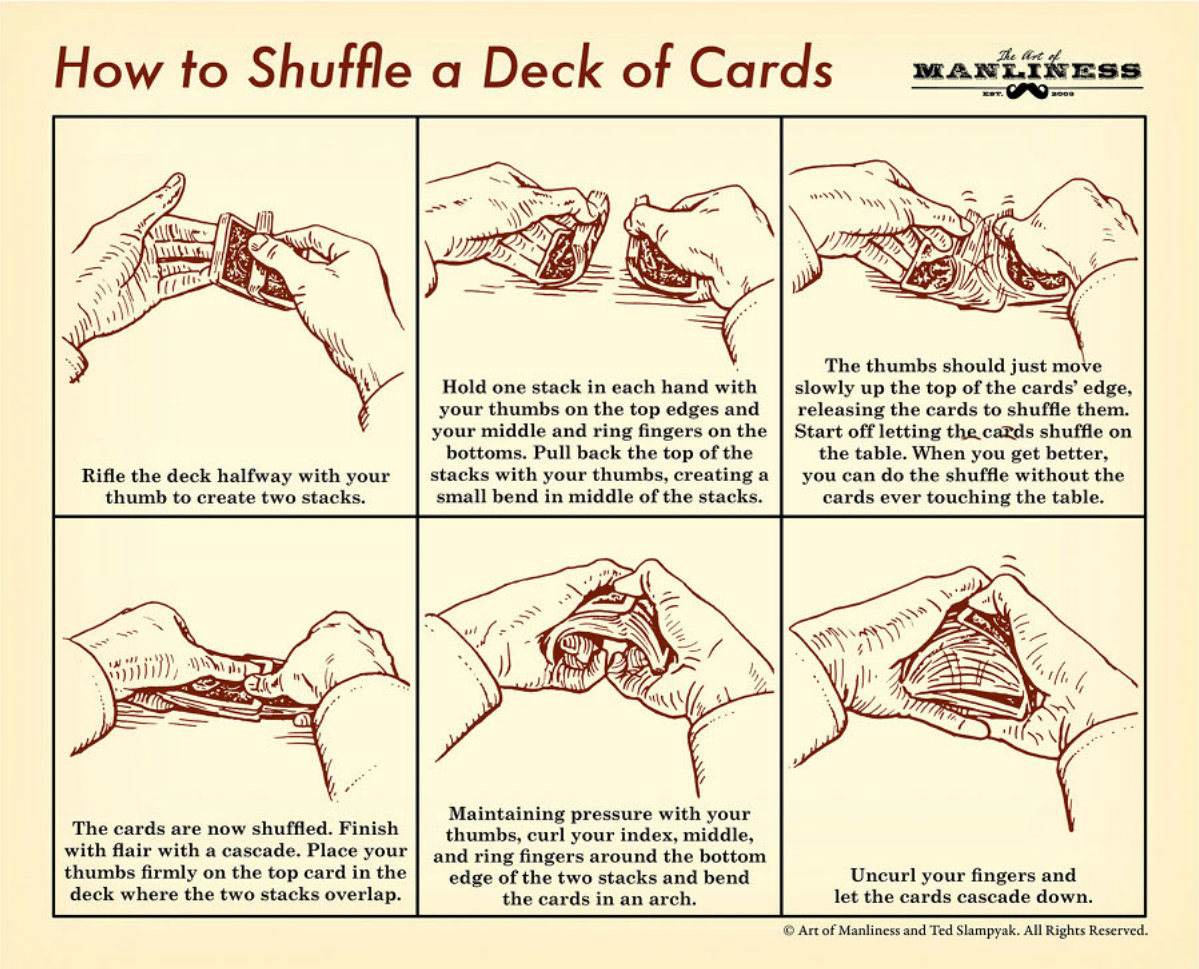
```
RndPerm (n) // constructs random permutation π [0..n-1]
{
    initialize π [0..n-1]
    for (i ⟵ 0,1, ..., n-1) {π [i] ⟵ i}
    for (i ⟵ 0,1, ..., n-2)
    {
        j ⟵ Random (i, i + 1, ..., n - 1) // one of the random i..n-1 val
        exchange π [i] ⟷ π [j]
    }
}
```

The function `Random` is a generator of pseudo-random numbers
to choose the index j uniformly distributed among $i, i+1, \ldots, n-1$.

- Implement the logic of `RndPerm` to calculate the random permutation of a simply chained list. More precisely, give the pseudocode for an operation `ListPerm( x , n )` that returns the head of a list with `n` nodes (for example, `n` *game* cards) after a random permutation of uniform distribution. The argument `x` is the head of the initial list with `n` nodes.
- Propose an algorithm that implements the riffle shuffle (illustrated below) using simply linked lists. Analyze the calculation time of the algorithm. To compare to `ListPerm` , note that it is enough to repeat the beating $c \cdot \lg n$ times for convergence to the uniform distribution [Dave Bayer and Persi Diaconis , 1992]. (In particular, seven iterations are sufficient for 52 cards.)



At input (a), we have a list $\mathtt{x} = (x_1, x_2, \ldots, x_n)$ and an argument $k \in \{0, 1, \ldots, n\}$. The algorithm (b) cutting the list $A = (x_1, \ldots, x_k)$ and $B = (x_{k+1}, \ldots, x_n)$ (lists can be empty), (c) interleaves the two lists randomly, (d) and thus constructed the random merging of two lists, where the original sequence between the elements of the same sublist remains the same.



2 Exercises: tree

2.1 Properties of nodes

- Give a recursive definition of the number of external nodes in a tree. Give a recursive algorithm that computes the number of external nodes in a binary tree. Give an algorithm that traverses the tree and calculates the exposure of all the nodes.
- Give a recursive definition for the *exposure* of a node u, defined as the shortest distance to an external node in the subtree rooted to u. Give a recursive algorithm that computes the exposure. Give a recursive algorithm that traverses the tree and computes the exposure to each node.
- Give lower and upper bounds on the maximum exposure in a tree with n internal nodes.
  Hint: Give proof by induction as has been done for height.

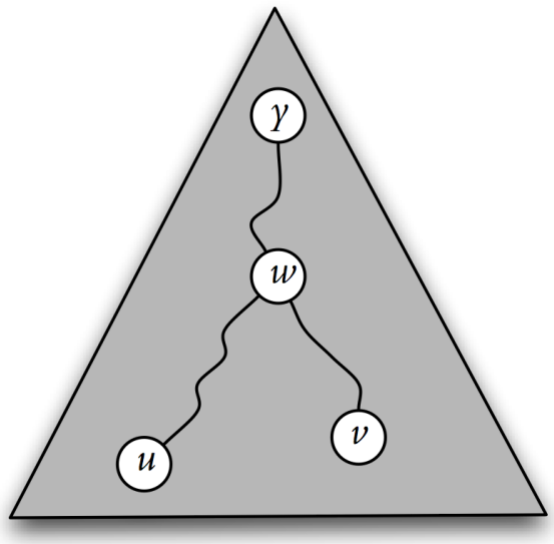
2.2 Polish compiler

The syntactic tree is a binary tree where the internal nodes are operations, and the external nodes are constants.

- Show the syntactic tree for the prefix expression `+ * 4 + 5 3 - 1 7`.

- Give an algorithm to construct a syntactic tree for arithmetic expressions from the prefix notation. The input is a sequence $x_1, x_2, \ldots, x_n$where each $x_i$is a constant or an operation (with two arguments). The algorithm must calculate the references to the left and right children and return the root of the tree. (You can assume that the sequence is of correct syntax.)
  Hint : Use the recursion to build the nodes of the tree.
- Give a recursive algorithm to evaluate the expression represented by a syntactic tree. The entrance is the root of the tree.

2.3 Common Ancestor



Give an algorithm that returns the lowest common ancestor of two nodes (u , v) in a binary tree rooted to y .

Hint : Gather the ancestors of u and v in an appropriate structure and identify those in common.