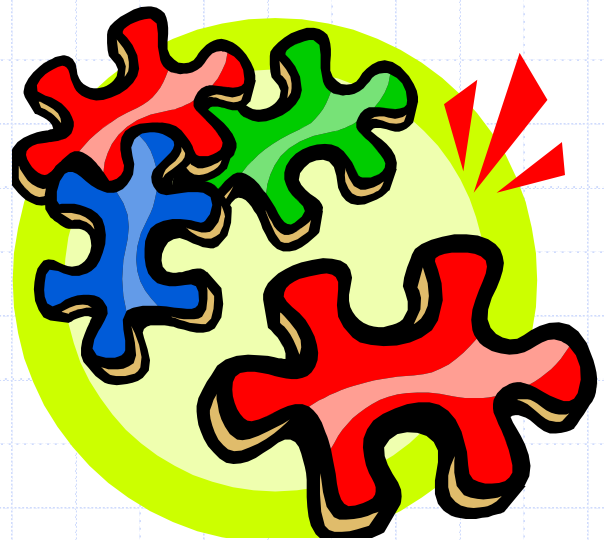
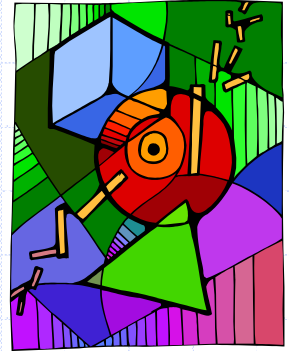


# Sets



# Set Operations

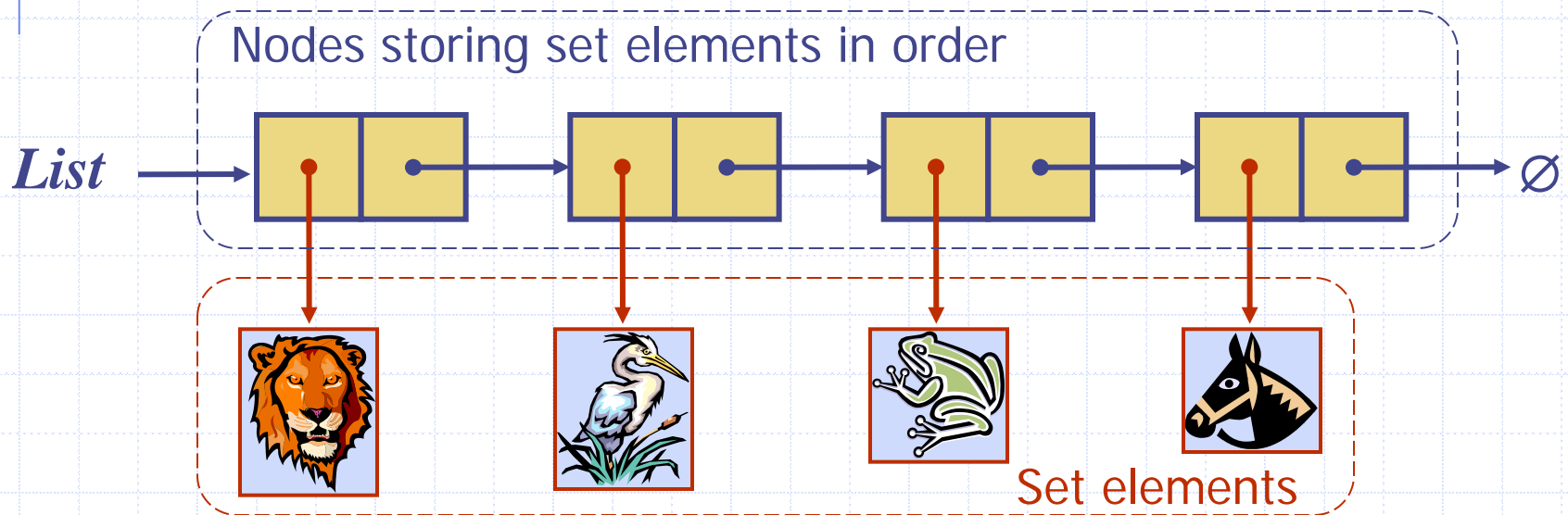


- ◆ We represent a set by the sorted sequence of its elements
- ◆ By specializing the auxiliary methods the generic merge algorithm can be used to perform basic set operations:
  - union
  - intersection
  - subtraction
- ◆ The running time of an operation on sets  $A$  and  $B$  should be at most  $O(n_A + n_B)$

- ◆ Set union:
  - *aIsLess(a, S)*  
*S.insertFront(a)*
  - *bIsLess(b, S)*  
*S.insertBack(b)*
  - *bothAreEqual(a, b, S)*  
*S.insertFront(a)*
- ◆ Set intersection:
  - *aIsLess(a, S)*  
{ *do nothing* }
  - *bIsLess(b, S)*  
{ *do nothing* }
  - *bothAreEqual(a, b, S)*  
*S.insertBack(a)*

# Storing a Set in a List

- ◆ We can implement a set with a list
- ◆ Elements are stored sorted according to some canonical ordering
- ◆ The space used is  $O(n)$



# Generic Merging

- ◆ Generalized merge of two sorted lists  $A$  and  $B$
- ◆ Template method **genericMerge**
- ◆ Auxiliary methods
  - **alsLess**
  - **blsLess**
  - **bothAreEqual**
- ◆ Runs in  $O(n_A + n_B)$  time provided the auxiliary methods run in  $O(1)$  time

```
Algorithm genericMerge( $A, B$ )  
   $S \leftarrow$  empty sequence  
  while  $\neg A.empty() \wedge \neg B.empty()$   
     $a \leftarrow A.front(); b \leftarrow B.front()$   
    if  $a < b$   
      alsLess( $a, S$ );  $A.eraseFront()$   
    else if  $b < a$   
      blsLess( $b, S$ );  $B.eraseFront()$   
    else {  $b = a$  }  
      bothAreEqual( $a, b, S$ )  
       $A.eraseFront(); B.eraseFront()$   
  while  $\neg A.empty()$   
    alsLess( $a, S$ );  $A.eraseFront()$   
  while  $\neg B.empty()$   
    blsLess( $b, S$ );  $B.eraseFront()$   
  return  $S$ 
```

# Using Generic Merge for Set Operations



- ◆ Any of the set operations can be implemented using a generic merge
- ◆ For example:
  - For **intersection**: only copy elements that are duplicated in both list
  - For **union**: copy every element from both lists except for the duplicates
- ◆ All methods run in linear time