Sets

Set ADT

- A collection of unordered distinct objects
  - there is no inherent ordering of elements in a set, but keeping the elements sorted can lead to more efficient set operations

- Main operations
  - union(B): executes $A \leftarrow A \cup B$
  - intersect(B): executes $A \leftarrow A \cap B$
  - subtract(B): executes $A \leftarrow A - B$
  - implemented using a generic version of the merge algorithm

- Running time of an operation should be at most $O(n_A + n_B)$

Storing a Set in a List

- We can implement a set with a list

- Elements are sorted according to some canonical ordering

- Space used is $O(n)$

2	5	6	7	8	9

Generic Merging

- Generalized merge of two sorted lists *A* and *B*
- Auxiliary methods aIsLess, bIsLess, bothAreEqual decide whether to add the element to list *S* based on what main operation is performed

Algorithm *genericMerge(A, B)*
   S ← empty sequence
   while ¬*A.isEmpty*() ∧ ¬*B.isEmpty*()
     a ← *A.first().element(); b* ← *B.first().element*()
     if *a < b*
       aIsLess(a, S); A.remove(A.first())
     else if *b < a*
       bIsLess(b, S); B.remove(B.first())
     else { *b = a* }
       bothAreEqual(a, b, S)
       A.remove(A.first()); B.remove(B.first())
   while ¬*A.isEmpty*()
     aIsLess(a, S); A.remove(A.first())
   while ¬*B.isEmpty*()
     bIsLess(b, S); B.remove(B.first())
   return *S*

4

Example: Union

- if $a < b$, copy a to output sequence and go to next element of A
- if $a = b$, copy a to output sequence and go to next element of A and B
- if $a > b$, copy b to output sequence and go to next element of B

A | 2 | 5 | 6 | 7 | 9 | B | 2 | 7 | 8 | 10 |

$$S = A \cup B$$

Example: Union

- if $a < b$, copy a to output sequence and go to next element of A
- if $a = b$, copy a to output sequence and go to next element of A and B
- if $a > b$, copy b to output sequence and go to next element of B

A | 2 | 5 | 6 | 7 | 9 | B | 2 | 7 | 8 | 10 |

$S = A \cup B$ | 2 |

Example: Union

- if $a < b$, copy a to output sequence and go to next element of A
- if $a = b$, copy a to output sequence and go to next element of A and B
- if $a > b$, copy b to output sequence and go to next element of B

A | 2 | 5 | 6 | 7 | 9 |

B | 2 | 7 | 8 | 10 |

$S = A \cup B$ | 2 | 5 |

Example: Union

- if $a < b$, copy a to output sequence and go to next element of A
- if $a = b$, copy a to output sequence and go to next element of A and B
- if $a > b$, copy b to output sequence and go to next element of B

A | 2 | 5 | 6 | 7 | 9 | B | 2 | 7 | 8 | 10 |

$S = A \cup B$ | 2 | 5 | 6 |

Example: Union

- if $a < b$, copy a to output sequence and go to next element of A
- if $a = b$, copy a to output sequence and go to next element of A and B
- if $a > b$, copy b to output sequence and go to next element of B
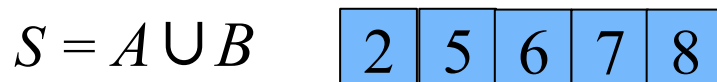
A | 2 | 5 | 6 | 7 | 9 |
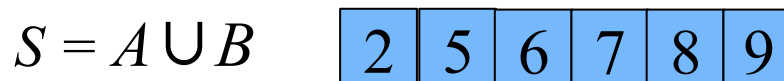
B | 2 | 7 | 8 | 10 |

$S = A \cup B$ | 2 | 5 | 6 | 7 |

Example: Union

- if $a < b$, copy a to output sequence and go to next element of A
- if $a = b$, copy a to output sequence and go to next element of A and B
- if $a > b$, copy b to output sequence and go to next element of B

A | 2 | 5 | 6 | 7 | 9 | B | 2 | 7 | 8 | 10 |

$S = A \cup B$ | 2 | 5 | 6 | 7 | 8 |

Example: Union

- if $a < b$, copy a to output sequence and go to next element of A
- if $a = b$, copy a to output sequence and go to next element of A and B
- if $a > b$, copy b to output sequence and go to next element of B

A | 2 | 5 | 6 | 7 | 9 | B | 2 | 7 | 8 | 10 |

$S = A \cup B$ | 2 | 5 | 6 | 7 | 8 | 9 |

Example: Union

- if $a < b$, copy a to output sequence and go to next element of A
- if $a = b$, copy a to output sequence and go to next element of A and B
- if $a > b$, copy b to output sequence and go to next element of B

A | 2 | 5 | 6 | 7 | 9 | B | 2 | 7 | 8 | 10 |

$S = A \cup B$ | 2 | 5 | 6 | 7 | 8 | 9 | 10 |

Using Generic Merge for Set Operations

- Any of the set operations can be implemented using a generic merge

- For example:
  - intersection: only copy elements that are duplicated in both lists
  - subtraction: only copy elements from *A* that are not equal to those in *B*

- All methods run in linear time.