CS 310
The end
# Interfaces and Classes

<table>
<thead>
<tr>
<th>Interface</th>
<th>Hash Table</th>
<th>Resizable Array</th>
<th>Balanced Tree</th>
<th>Linked List</th>
<th>Hash Table + Linked List</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set</td>
<td>HashSet</td>
<td></td>
<td>TreeSet</td>
<td>LinkedHashMap</td>
<td>LinkedHashSet</td>
</tr>
<tr>
<td>List</td>
<td></td>
<td>ArrayList</td>
<td></td>
<td>LinkedList</td>
<td></td>
</tr>
<tr>
<td>Deque</td>
<td></td>
<td>ArrayDeque</td>
<td></td>
<td>LinkedList</td>
<td></td>
</tr>
<tr>
<td>Map</td>
<td>HashMap</td>
<td></td>
<td>TreeMap</td>
<td>LinkedHashMap</td>
<td>linkedHashSet</td>
</tr>
</tbody>
</table>

Deque: double ended queue

Also: java.util.SortedMap, java.util.NavigableMap interfaces
Other

• Concurrent Collections
  • Interfaces:
    • BlockingQueue
    • TransferQueue
    • BlockingDeque
    • ConcurrentHashMap
    • ConcurrentNavigableMap
  • Classes
    • LinkedBlockingQueue
    • ArrayBlockingQueue
    • PriorityBlockingQueue
    • DelayQueue
    • ....
Python

• Python’s built-in containers, dict, list, set, and tuple.
• Python’s collections (from Lib/collections.py and Lib/__abcoll.py)
  • Deque: generalization of stacks and queues
    • list-like container with fast appends and pops on either end
  • Counter: support convenient and rapid tallies
    • dict subclass for counting hashable objects
  • OrderedDict: remember the order that items were inserted
    • dict subclass that remembers the order entries were added
  • Defaultdict (add __missing__(key))
    • dict subclass that calls a factory function to supply missing values

See more (e.g., ChainMap, UserList, UserDict, etc) at https://docs.python.org/3/library/collections.html
C++

• STL (Standard Template Library)
  • Algorithms (Sort, search, partition, permutation, etc)
  • Containers (Data structures)
  • Functions (function objects, e.g. comparator)
  • Iterators
The C++ standard library generic containers

- **Elements keep insertion ordering?**
  - Yes → Sequential access
    - Fast insert and remove?
      - Yes → Variable size
        - Access first only → std::vector
        - Access last only → std::stack
      - No → Accessing elements?
        - Restricted → std::list
        - Bidirectional
          - Random access → std::queue
          - Sequential access → std::priority_queue
            - Can traverse in sorted order?
              - Yes → Sorted
              - No → std::unordered_set
            - Cannot traverse → std::unordered_multiset
        - Keyed access → std::unordered_map
      - Both ends → Homogeneous
      - One end only → std::deque
    - No → Accessing elements?
      - Restricted → std::array
      - Bidirectional
        - Random access → std::forward_list
        - Sequential access → std::tuple
  - No → Heterogeneous

- **Size changes at run time?**
  - Yes → Fixed size
    - Elements all the same type?
      - Yes → Homogeneous
      - No → std::pair
    - Access first only → std::vector
    - Access last only → std::stack
    - Accessing elements?
      - Restricted → std::list
      - Bidirectional
        - Random access → std::queue
        - Sequential access → std::priority_queue
          - Can traverse in sorted order?
            - Yes → Sorted
            - No → std::unordered_set
          - Cannot traverse → std::unordered_multiset
      - Both ends → std::deque
    - One end only → std::deque
    - Keyed access → std::unordered_map
  - No → std::pair

- **Elements all the same type?**
  - Yes → Homogeneous
  - No → std::pair
Multi-map and Multi-set

• They are very similar to map and set but they keys do not have to be unique
  • For multi-map the key-value pair has to be unique
What are missing?

- What are not provided in Java Collections, Python, or C++ STL?
What are missing?

• What are not provided in Java Collections, Python, or C++ STL?

  Google Guava (Java)
  https://github.com/google/guava

  Apache Commons Collections (Java)
  http://commons.apache.org/proper/commons-collections/

  The Boost Graph Library (C++)
  http://www.boost.org/doc/libs/1_65_1/libs/graph/doc/index.html
What are missing?

- CGAL (Computational Geometry Algorithms Library) for spatial data structures
  - Trees: binary space partitioning tree, K-D tree, Range-tree, Quad-tree, ...
  - Graph: subdivisions, doubly-linked edge list, ...
- Out of Core Data structures
  - B+ tree,
  - Log Structured Merge (LSM) trees
    - Multiple levels of B-tree for fast write
    - Used by Google, Apache, Yahoo!, and many modern databases
    - But much more complex than a B-Tree...

http://www.benstopford.com/2015/02/14/log-structured-merge-trees/
What are missing?

- Hashing
- Data structures for specific architectures, e.g. GPU
- Data structures related to time, temporal sequence
- Data structures for high-dimensional data (>100 dimensions)
- Data structures for parallelization and concurrent computation
What is Next?

• CS 483
  • More graphs
  • More complexity analysis
  • More problem solving techniques

• Advanced data structures
  • https://courses.csail.mit.edu/6.851/fall17/lectures/
  • http://jeffe.cs.illinois.edu/teaching/datastructures/
    • There is a “similar courses elsewhere” that would take you to more courses
  • I-O aware http://www.daimi.au.dk/~large/ioS13/
  • http://cglab.ca/~morin/teaching/5408/