Midterm Topics

Matt Warner

- Complexity analysis
- Type declarations
- Containers
  - vector
  - deque
  - list
  - forward list
  - set
  - map
- iterators
- functors

1 Containers

1.1 vector

1.1.1 Container type

Sequence container.

Implemented as a dynamic array.

1.1.2 Iterator type

Random access iterator

1.1.3 Advantages/disadvantages

Adding/removing elements at the end is fast. having the ability to grow/shrink the container and have random access to elements is beneficial in some cases.

Inserting/removing in the middle/beginning is slow.

1.1.4 Time complexities

- Linear time
  - clear
  - insert
  - erase
  - assign
  - resize
  - dtor
  - operator=
  - copy ctor
  - ctor with iterator range.
- Constant time
  - push_back
  - emplace_back
  - pop_back
  - swap
  - operator[] and at()

2 deque

2.1 container type

sequence container. Implemented as multiple dynamic arrays.

Almost the same interface as vector, but does not provide capacity, reserve.

Provides a couple extra modifiers that are not in vector:

- push_front
- emplace_front
- pop_front

Internal memory is not contiguous. Element access and iterator movement involves more calculation comparing to vector. Iterators may need to jump between difference memory blocks

2.2 iterator type

Random access iterator

2.3 advantages/disadvantages

Fast for adding/removing at both ends. Slow in adding/removing in the middle

Possibly larger max size

Massive reallocations are avoided

2.4 Time complexities

3 list

3.1 container type

sequence container. Implemented as a doubly linked list

3.2 iterator type

bidirectional iterator

3.3 advantages/disadvantages

Adding/removing anywhere is fast so long as you have a reference to the location. Otherwise you need to traverse the list.

No reallocation needed.

3.4 Time complexities

4 forward list

4.1 container type

Sequence container. Internally implemented as a singly linked list.

4.2 iterator type

Forward iterator

4.3 advantages/disadvantages

One linking field compared to list with two linking fields. More memory efficient.

Adding/removing at begin of a list is fast. Adding/removing at the next position of a given element is fast.

4.4 Time complexities

5 set

5.1 container type

Associative container. Implemented as a self-balancing binary search tree. (red-black tree).

5.2 iterator type

birectional iterator

5.3 advantages/disadvantages

The advantage is not for sorting, but for searching. The order of insertion does not matter.

5.4 Time complexities

6 map

6.1 container type

6.2 iterator type

6.3 advantages/disadvantages

6.4 Time complexities