Jacob Wahbeh

CS 32

Project 2 Report

1. I implemented my Set as a doubly linked list with a size variable, no dummy node, and not circular. Each item is inserted in order, and each node contains a value, next pointer, and previous pointer.

Typical Set: Size = 4

T

H

“D”

“C”

“B”

“A”

Empty Set: Head and Tail are nullptr, size = 0

H

T

2.

**Copy Constructor**

If other set is empty

Head, tail are nullptr, and size is zero

Return

Create new Node

Assign value of the first node from the other set

Assign head to new node

Loop through other Set

Create a new node

Assign it the value for nth node in other set

Link next and prev pointers

Assign tail to last node

**Assignment operator**

If other set equals this set

Return this set

Create temporary set copy of other

Swap this set with temporary

**Insert**

If the value is already in the set

Return false

Create new node

Assign its value as specified in the parameter

If set is empty

Add to front

Increment size and return

Loop through Set

If value < first node

Add to front

Break

If value > last node

Add to rear

Break

If nth node < value < nth +1 node

Add between the two nodes

Break

Increment size and return

**Erase**

If value is not contained in set, or set is empty

Return false

If size = 1

Delete node that head points to

Set head and tail to nullptr

Decrement size and return true

If value is equal to the first node

Create temporary pointer that points to first node

Assign head to next node

Delete temporary pointer

Decrement size and return true

If value is equal to last node

Follow the same pattern as first node but assign tail to the second to last node

Loop through set

If value is equal to nth + 1 node

Create temporary node pointer that points to nth +1 node

Link nth and nth + 2 next and prev pointers

Delete temporary pointer

Decrement size and return true

**Get**

Return false if pos is out of bounds

Loop through set until the position in the parameter

Set value to the item of that node

Return true

**Unite**

Create new empty set

Loop through set1 and insert all items into empty set

Loop through set2 and insert all items into empty set

Swap result and empty sets

**Subtract**

Create empty set

Loop through set1

If the node in set1 is not contained in set2

Insert it into empty set

Swap result and empty sets

3. Test Cases

//default constructor

Set a;

Set b;

assert(a.empty()); //test empty

assert(a.size() == 0);

assert(!a.erase("t")); //empty sets cant erase

assert(a.insert("d")); //inserting to an empty set

a.dump();

assert(a.insert("b")); //more insertions

assert(a.insert("f"));

assert(a.insert("k"));

assert(!a.insert("f")); //cannot insert duplicates

a.dump();

assert(a.size() == 4); //make sure size is correct

assert(!a.empty());

assert(a.contains("f")); //testing contains

assert(!a.contains("q"));

ItemType x = "hey";

assert(!a.get(4, x) && x == "hey"); //out of bounds pos test

assert(!a.get(-1, x) && x == "hey");

assert(a.get(0, x));

assert(x == "b"); //first node is b

assert(a.get(3, x));

assert(x == "k"); //last node is k

assert(!b.get(0, x) && x == "k"); //cannot use get on empty sets

b.swap(a);

a.dump(); //swapping empty set with existing set

b.dump();

a.insert("u");

a.swap(b); //swapping a set with 1 item

b.insert("t");

b.insert("i");

a.swap(b); //swapping with multiple items

a.dump();

b.dump();

b.swap(a);

Set c(a); //copy constructor

ItemType u, v;

assert(c.size() == a.size()); //size is the same

for (int i = 0; i < a.size(); i++) { //each node is the same

assert(a.get(i, u) && c.get(i, v));

assert(u == v);

}

Set d;

d.insert("y");

d.insert("q");

d = b; //assignment operator

assert(d.size() == b.size()); //size and nodes are the same

for (int i = 0; i < b.size(); i++) {

assert(b.get(i, u) && d.get(i, v));

assert(u == v);

}

assert(!a.erase("r")); //cannot erase values not present in array

assert(a.erase("f"));

assert(a.erase("b"));

assert(a.erase("k")); //basic erase tests

assert(a.erase("d"));

assert(!a.erase("d")); //cant erase same value twice for it doesnt exist

assert(a.insert("t"));

assert(a.insert("u"));

assert(a.insert("y"));

assert(a.insert("e"));

a.dump();

d.dump();

Set e;

c = a;

b = d;

unite(a, d, e); //result is empty set

e.dump();

unite(a, d, a); //result is the first set

unite(a, a, a); //all sets are the same

unite(d, d, e); //first two sets are the same

a.dump();

Set f;

subtract(c, b, f); //result is empty

f.dump();

subtract(c, b, a); //result is an existing set

a.dump();

subtract(c, b, c); //first set is also result

c.dump();

subtract(c, c, c); //all same

c.dump();

subtract(b, b, f); //first two the same

f.dump();

}