Wenlan Tian

COP 3530

Section 1087, MAEB 211

11/18/2014

Homework 8

"On my honor, I have neither given nor received unauthorized aid in doing this assignment.”

Summary of learning experience:

The easiest part of the task is to understand the theory of search function in hash table with open addressing and implement.

The most difficult parts of the task: In part 2, I don’t know when h’s value could be negative because I think all the modulus should be non-negative.

The assignment’s educational objective: part 1:understand the remove function I hash table with open addressing, especially that after we remove the item in the slot, we have to move the items in the same key forward. Part2: understand universal hashing.

How well I think I achieved them: For part 1, I think I meet all the specifications for the assignment. My test showed my implementation is correct. For part 2, I don’t know the answer.

Wenlan Tian

COP 3530

Section 1087, MAEB 211

11/18/2014

Homework 8

1. Does the program compile without errors?

No error.

2. Does the program compile without warnings?

No warnings.

3. Does the program run without crashing?

No crashing.

4. Describe how you tested the program.

I also write the insert() and print() function. With the print function, we can easily see the result after each step.

5. Describe the ways in which the program does not meet assignment’s specifications.

I think the program meets the assignment’s specifications.

6. Describe all known and suspected bugs.

No bugs detected.

7. Does the program run correctly?

It runs correctly.

Wenlan Tian

COP 3530

Section 1087, MAEB 211

11/18/2014

Homework 8

Homework 8:

Part I. Write in C++, the remove function [should handle wrap-around].

const int mapSize = 10;

template <typename K, typename V>

class HMOA{

private:

V a[mapSize];

int mySize;

public:

HMOA();

int hash(K);

bool insert (K, V);

bool remove (K, V&);

bool search (K, V&);

void clear();

bool is\_empty();

std::size\_t capacity();

std::size\_t size();

double load();

std::ostream& print (std::ostream&) const;

};

template <typename K, typename V>

HMOA<K,V>::HMOA():mySize(0){

for (int i =0; i < mapSize; ++i){

a[i]=V();

}

}

template <typename K, typename V>

int HMOA<K,V>::hash(K key){

return key % mapSize;

}

template <typename K, typename V>

bool HMOA<K,V>::insert(K key, V value){

int i = hash(key);

while(a[i]){

++i;

}

if(!a[i]){

a[i]= value;

mySize ++;

return true;

}else{

return false;

}

}

template <typename K, typename V>

bool HMOA<K,V>::remove(K key, V& value){

if (!search(key, value)) return false;

int i = hash(key);

while(a[i] && a[i]!=value){

++i;

}

while(a[i+1]){

a[i]=a[i+1];

++i;

}

a[i]=V();

mySize --;

return true;

}

template <typename K, typename V>

bool HMOA<K,V>::search(K key, V& value){

int i = hash(key);

while(a[i] && a[i]!=value){

++i;

}

if(!a[i]){

return false;

}else{

return true;

}

}

int main(){

///////////////TEST <int, char> ///////////

HMOA<int, char> hmoa;

//test insert

hmoa.insert (12,'x');

hmoa.insert (12,'y');

hmoa.insert (15,'z');

hmoa.print(std::cout);

//test size

std::cout << "Size: " << hmoa.size() << std::endl;

//test search

char m = 'm';

if(hmoa.search(12,m)){

std::cout << "found" << std::endl;

}else{

std::cout << "not found" << std::endl;

}

}

Part II. Program 14.2

- Is this an example of universal hashing?

Yes, it is.

- Under what circumstances, if any, would h’s value be negative? Justify your answer.

I don’t know.