Alex McNurlin

Programming assignment 5

CS121

4/30/16

# Programming log

Time spent:

* Saturday, Apr 30
  + 11:30-12:30
  + 1:30-3:00
* **Total Time: 2.5 hours**

Things encountered

* Since the majority of the code was already written and was given on the course website, the programming portion was much easier and quicker than I expected
* Hash tables are much faster and more efficient than I expected!

Difficulties

* The code given uses a lot of C syntax, so some bits are a bit more difficult to understand
* When an empty line is encountered by the program, it treats it as a string and puts it into the table with a hash value of 0. Not a huge issue, but would be necessary to fix in a scenario when the input files aren't as well formatted

## Notes on runtime

The following is the time taken to execute the program with various hash table sizes. The input file was “dict4.txt”, which contained 58,734 words to store in the table. The time was recorded by executing the program with the `time` command (i.e. `time ./hw4` )

* Table Size 7159:
  + Time: 0.061
  + Empty buckets: 0
* Table Size 11037:
  + Time: 0.066
  + Empty buckets: 50
* Table Size 49937:
  + Time: 0.210
  + Empty buckets: 15448
* Size 100:
  + Time: 0.307
  + Empty buckets: 0

# Code – Output (typescript.txt)

Script started on Sat 30 Apr 2016 02:54:26 PM PDT

alexmcnurlin@Laquisha: ~/Google Drive/schoolStuff/cs121/hw5alexmcnurlin@Laquisha:~/Google Drive/schoolStuff/cs121/hw5$ ./hw5

Enter filename: dict4.txt

Hash table Bucket Size:

--------------------

0: 595 items

1: 568 items

2: 585 items

3: 513 items

4: 594 items

5: 591 items

6: 616 items

7: 553 items

8: 540 items

9: 592 items

10: 565 items

11: 544 items

12: 606 items

13: 560 items

14: 625 items

15: 546 items

16: 569 items

17: 587 items

18: 574 items

19: 610 items

20: 598 items

21: 570 items

22: 608 items

23: 559 items

24: 573 items

25: 606 items

26: 587 items

27: 576 items

28: 599 items

29: 543 items

30: 608 items

31: 611 items

32: 603 items

33: 603 items

34: 545 items

35: 577 items

36: 564 items

37: 581 items

38: 574 items

39: 611 items

40: 593 items

41: 607 items

42: 615 items

43: 618 items

44: 597 items

45: 638 items

46: 585 items

47: 588 items

48: 578 items

49: 615 items

50: 616 items

51: 565 items

52: 585 items

53: 579 items

54: 564 items

55: 531 items

56: 555 items

57: 582 items

58: 592 items

59: 573 items

60: 571 items

61: 603 items

62: 614 items

63: 564 items

64: 603 items

65: 550 items

66: 598 items

67: 614 items

68: 600 items

69: 549 items

70: 631 items

71: 684 items

72: 585 items

73: 574 items

74: 578 items

75: 602 items

76: 569 items

77: 614 items

78: 526 items

79: 632 items

80: 565 items

81: 595 items

82: 562 items

83: 580 items

84: 613 items

85: 626 items

86: 581 items

87: 609 items

88: 567 items

89: 629 items

90: 591 items

91: 579 items

92: 578 items

93: 630 items

94: 600 items

95: 608 items

96: 586 items

97: 589 items

98: 572 items

99: 609 items

0 buckets had no contents

Max: Bucket 71 has 684 nodes

Min: Bucket 3 has 513 nodes

Found "unity" in bucket 89. 54 probes requred

Found "za" in bucket 79. 1 probes requred

Found "abler" in bucket 88. 567 probes requred

Found "wager" in bucket 54. 14 probes requred

Found "wormy" in bucket 54. 5 probes requred

Found "memorable" in bucket 78. 266 probes requred

Found "mango" in bucket 30. 322 probes requred

Found "physics" in bucket 63. 208 probes requred

Found "nameless" in bucket 16. 286 probes requred

Found "cubit" in bucket 75. 476 probes requred

alexmcnurlin@Laquisha: ~/Google Drive/schoolStuff/cs121/hw5alexmcnurlin@Laquisha:~/Google Drive/schoolStuff/cs121/hw5$ exit

Script doe on Sat 30 Apr 2016 02:55:01 PM PDT

# Code – hw5.cpp

#include <iostream>

#include <cstdlib>

#include <fstream>

#include "hash.h"

using namespace std;

int main() {

char filename[32];

cout << "Enter filename: ";

cin >> filename;

ifstream in\_file(filename);

if ( in\_file.fail() ) {

cout << "File " << filename << " is invalid " << endl << "Exiting..." << endl;

exit(1);

}

// Read in lines until end of file

while ( !in\_file.eof() ) {

char in\_char\_string[32];

in\_file.getline( in\_char\_string, 32 );

Insert( in\_char\_string ); // Insert line into hash table

}

// Count the number of values in each bucket

CountHashTable();

cout << endl;

// find buckets with minimum and maximum values

HashMinMax();

cout << endl;

// Lookup 10 words in the table

char words\_to\_lookup[10][32] = {

"unity",

"za",

"abler",

"wager",

"wormy",

"memorable",

"mango",

"physics",

"nameless",

"cubit"

};

for (int i = 0; i < 10; i++ ) {

Lookup\_with\_output( words\_to\_lookup[i] );

}

cout << endl;

}

# Code – hash.h

/\* hash.h

\*/

#ifndef HASH\_H

#define HASH\_H

struct nList /\* table entry: \*/

{

char \*name; /\* defined name \*/

//char \*defn; /\* replacement text \*/

struct nList \*next; /\* next entry in chain \*/

};

typedef struct nList \*NListPtr;

unsigned Hash( char \*s );

NListPtr Lookup( char \*s );

NListPtr Lookup\_with\_output( char \*s );

NListPtr Insert( char \*name );

void CountHashTable();

void PrintHashTable();

void HashMinMax();

#endif /\* HASH\_H \*/

# Code – hash.cpp

/\* Hash.cpp

\*

\* Hash table implementation from:

\* Kernighan & Ritchie, The C Programming Language,

\* Second Edition, Prentice-Hall, 1988.

\*/

#include <iostream>

#include <iomanip>

#include <cstdlib>

#include <cstring>

using namespace std;

#include "hash.h"

const int HASH\_TABLE\_SIZE = 100;

static NListPtr hashTable[HASH\_TABLE\_SIZE];

// Prototypes

static char \*Strdup( const char \* ); // in cstring, but....

/\* Hash

\* Generate hash value for string s

\*/

unsigned Hash( char \*s )

{

unsigned hashVal;

for( hashVal = 0 ; \*s != '\0' ; s++ )

hashVal = \*s + 31 \* hashVal;

return hashVal % HASH\_TABLE\_SIZE;

}

/\* Lookup

\* Look for s in hashTable

\*/

NListPtr Lookup( char \*s )

{

NListPtr np;

for( np = hashTable[Hash(s)] ; np != NULL ; np = np->next )

{

if( strcmp(s, np->name) == 0 )

return np; // found

}

return NULL; // not found

}

// Similar to lookup, but prints results to the screen

NListPtr Lookup\_with\_output( char \*s )

{

NListPtr np;

int i = 0;

for( np = hashTable[Hash(s)] ; np != NULL ; np = np->next )

{

i++;

if( strcmp(s, np->name) == 0 ) {

cout << "Found \"" << s << "\" in bucket " << Hash(s) << ". " << i << " probes requred" << endl;

return np; // found

}

}

cout << "\"" << s << "\" not found." << endl;

return NULL; // not found

}

/\* Insert

\* Put name in hash table

\*/

NListPtr Insert( char \*name )

{

unsigned hashVal;

NListPtr np;

if( (np = Lookup(name)) == NULL ) // not found

{

np = (NListPtr) malloc(sizeof(\*np));

if( np == NULL || (np->name = Strdup(name)) == NULL )

return NULL;

hashVal = Hash(name);

np->next = hashTable[hashVal];

hashTable[hashVal] = np;

}

return np;

}

/\* PrintHashTable

\* Print the hash table contents

\*/

void PrintHashTable()

{

NListPtr np;

cout << "Hash table contents:" << endl;

cout << "--------------------\n" << endl;

for( int i = 0 ; i < HASH\_TABLE\_SIZE ; i++ )

{

np = hashTable[i];

while( np != NULL )

{

cout << setw(3) << i << ": ";

cout << np->name;

cout << endl;

np = np->next;

}

}

}

/\* Strdup

\* Make a duplicate copy of s

\*/

static char \*Strdup( const char \*s )

{

char \*p;

p = (char \*) malloc(strlen(s)+1); /\* +1 for '\0' \*/

if( p != NULL )

strcpy(p,s);

return p;

}

// Count the number of items in each bucket and print to the screen

void CountHashTable() {

NListPtr np;

int k = 0;

cout << "Hash table Bucket Size:" << endl;

cout << "--------------------\n" << endl;

// Count the number of items in each bucket

for( int i = 0 ; i < HASH\_TABLE\_SIZE ; i++ )

{

np = hashTable[i];

int j = 0;

while( np != NULL )

{

j++;

np = np->next;

}

if ( j == 0 ) {

k++;

}

cout << i << ": " << j << " items" << endl;

}

cout << k << " buckets had no contents" << endl;

}

// Finds the bucket with the most entries and the least

void HashMinMax() {

NListPtr np;

// minimum and maximum keys/values

int min\_i = -1;

int min\_j = 0;

int max\_i = -1;

int max\_j = 0;

for( int i = 0 ; i < HASH\_TABLE\_SIZE ; i++ )

{

np = hashTable[i];

int j = 0;

while( np != NULL )

{

j++;

np = np->next;

}

// Set the min and max to the first if it hasn't been set yet

if ( min\_i == -1 ) {

min\_i = i;

min\_j = j;

max\_i = i;

max\_j = j;

} else if ( j < min\_j ) { // Set min value

min\_j = j;

min\_i = i;

} else if ( j > max\_j ) { // Set max value

max\_j = j;

max\_i = i;

}

}

// Print results

cout << "Max: Bucket " << max\_i << " has " << max\_j << " nodes" << endl;

cout << "Min: Bucket " << min\_i << " has " << min\_j << " nodes" << endl;

}