Maria Martinez

Justin Swenson

CS 3240

06/03/2014

Program # 4

// updatecbt.cpp

#include <fstream>

#include <iomanip>

#include <iostream>

#include <new>

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include <time.h>

#include <math.h>

using namespace std;

struct mrec {

unsigned int acct;

char name[25]; // strlen( name ) <= 24

float baln;

};

struct node {

node \*left;

mrec info;

node \*rite;

};

struct trec {

unsigned int acct;

char name[25];

float tran;

};

node \*const nill = (node \*)0;

const unsigned int arraysize = 32; // fllg of max nodes in CBT

const unsigned int delta = 2;

const unsigned int ssize = 32;

unsigned int hi, lo, mhz = 2900;

clock\_t clk;

int main(int argc, const char \* argv[])

{

node \*lltoct( unsigned int, node \*); // linked list to cbt

void newerr();

unsigned int timeout( void );

void chop( node \* ), fitoll( ifstream &, unsigned int &, node \*& ), output( const node \*, ofstream & ), timein( void ), update( node \*&, unsigned int &, ifstream &, unsigned int &, unsigned int &, unsigned int & );

ifstream fp0, fp1; // master file, transaction file

ofstream fp2; // new master file

node \*tree; // this will always point to root node

unsigned int del, ins, maxh, n, t, tbegin, tend; // inst = insrtion ; maxh = max height ; n = # of nodes ; t = time

char ch;

if ( argc != 4 ) { // user input incorrect format

cout << "\*\* use is >updatecbt masfil tranfil newmasfil \*\*\n";

exit( 1 );

}

set\_new\_handler(newerr);

fp0.open( \*++argv, ios::in | ios::binary ); // open master file

if ( fp0.fail() ) {

cout << "\*\* can't open\"" << \*argv << "\" \*\*\n";

exit( 1 );

}

fp1.open( \*++argv, ios::in | ios::binary ); // open transaction file

if ( fp1.fail() ) {

cout << "\*\* can't open\"" << \*argv << "\" \*\*\n";

exit( 1 );

}

fp2.open( \*++argv, ios::in | ios::binary ); // open the new file to see if name already exists

if ( !fp2.fail() ) { // file was successfully opened so the name already exists

cout << "\*\* \"" << \*argv << "\" exists \*\*; overwrite? (y/n) ";

cin.get( ch );

if ( ch != '\n' ) {

while ( cin.get() != '\n' ) { // read entire input buffer so nothing is left behind

;

}

}

if ( (ch != 'y') && (ch != 'Y') ) { // user does not want to overwrite file; abort.

fp0.close();

fp1.close();

fp2.close();

cout << "-- ok: \"" << \*argv << "\" is unchanged --\n";

exit( 1 );

}

fp2.close();

}

fp2.clear();

fp2.open( \*argv, ios::out | ios::binary );

if ( fp2.fail() ) {

fp0.close();

fp1.close();

cout << "\*\* can't open \"" << \*argv << "\" \*\*\n";

exit( 1 );

}

tbegin = clock(); // start timer

fitoll( fp0, n, tree ); // file to linked list

tend = clock(); // end timer

t = tend - tbegin; // how long did this take?

fp0.close();

cout << "-- read " << n << " recs into linked list --; " << (float(t + 500) / 10) << " msec\n";

tbegin = clock(); // start timer

tree = lltoct(n, tree); // link list to complete tree ( n = # of nodes ; tree = head )

tend = clock(); // stop timer

t = tend - tbegin; // how long did this take?

cout << "-- made complete binary tree --; " << (float(t + 500) / 10) << " msec\n";

tbegin = clock(); // start timer

update( tree, n, fp1, del, ins, maxh ); // update to complete binary tree

tend = clock(); // stop timer

t = tend - tbegin; // how long did this take?

fp1.close();

cout << "-- updated --:\n " << del << " del; " << ins << " ins; " << del + ins << " changes; max height = " << maxh << "; " << (float(t + 500) / 10) << " msec\n";

tbegin = clock(); // start timer

output( tree, fp2 ); // create the new file

tend = clock(); // end timer

t = tend - tbegin; // how long did this take?

fp2.close();

cout << "-- wrote " << n << " recs to file \"" << \*argv << "\" --; " << (float(t + 500) / 10) << " msec\n";

tbegin = clock(); // start timer

chop( tree ); // LRN - post fix traversal of tree to destroy tree

tend = clock(); // end timer

t = tend - tbegin; // how long did this take?

cout << "-- chopped down tree --; " << (float(t + 500) / 10) << "msec\n";

return 0;

}

void

chop( node \* p) {

if ( p != nill ) {

chop(p->left);

chop(p->rite);

delete p;

}

}

void

fitoll( ifstream & f, unsigned int & n, node \* & head ) {

node \*p = nullptr, \*q;

mrec rec;

n = 0;

head = nill;

while ( f.read((char \*)&rec, sizeof(mrec)) ) {

q = new node;

q->info = rec;

if ( ++n == 1 )

head = q;

else

p->rite = q;

p = q;

}

}

void

fix( node \*& tree, unsigned int & n, unsigned int & h, unsigned int & maxh0, unsigned int & n0 ) {

node \*reshape( node \*, unsigned int & );

unsigned int fllg( unsigned int );

tree = reshape( tree, n );

maxh0 = ( h = fllg(n) ) + delta;

n0 = ( 1 << (h-1) );

}

unsigned int

fllg( unsigned int x ) { // floor log base 2 of x

unsigned int r = 0;

r = floor(log2(x));

return r;

}

void

freeleaf( node \*p, node \*q ) {

if ( q != nill ) {

if ( q->left == p )

q->left = nill;

else

q->rite = nill;

}

delete p;

}

unsigned int

isleaf( node \*p ) {

return ( (p->left == nill) && (p->rite == nill) );

}

node \*

lltoct( unsigned int n, node \* q ) { // n = # of nodes ; q = tree = head = root node

unsigned int fllg( unsigned int ), nu2( unsigned int );

node \*last[arraysize], \*p; // n < 2\*\*arraysize

unsigned int b, h, i, l;

char pat[arraysize]; // very short ints: 0 or 1

if ( n < 1 ) // this tree is empty

return (nill);

b = n + 1 - ( 1 << (h = fllg(n)) );

b += b; // double b

for ( l = 0 ; l <= h ; l++ ) {

last[l] = nill;

}

for ( i = 1 ; i <= n ; i++ ) {

l = h - nu2( i <= b ? i : i+i-b );

last[l] = p = q;

q = q->rite;

p->left = p->rite = nill;

pat[l] = (char)0;

if ( l < h ) {

if ( pat[l+1] == (char)0 ) {

p->left = last[l+1];

pat[l+1] = (char)1;

}

}

if ( l != 0 ) {

if ( last[l-1] ) {

if ( last[l-1]->rite == nill ) {

last[l-1]->rite = p;

pat[l] = (char)1;

}

}

}

}

return (\*last);

}

void

newerr() {

cout << "\*\* can't allocate space for node \*\*\n";

exit( 1 );

}

unsigned int

nu2( unsigned int x ) {

unsigned int result = 0;

unsigned int n = x;

while ( (n / 2) > 0 ){

if ( n == 1 )

break;

x = n % 2;

n = n / 2;

if ( x == 1 )

break;

if ( x == 0 )

result++;

}

return result;

}

void

output( const node \*p, ofstream & f ) {

if ( p != nill ) {

output( p->left, f );

if ( !f.write( (char\*)&(p->info),sizeof(mrec) ) ) {

cout << "\*\* can't f.write record \*\*\n";

exit( 1 );

}

else

output( p->rite, f );

}

}

void

remove( node \*p, node \*q ) { // p = pointer to node to delete ; q = pointer to parent

void freeleaf( node \*, node \* );

unsigned int isleaf( node \* );

node \*r, \*s;

if ( isleaf(p) ) {

freeleaf( p, q );

return;

}

q = p;

if ( p->rite != nill ) {

r = p->rite;

while ( r->left != nill ) {

q = r;

r = r->left;

}

}

else {

r = p->left;

if ( r->rite != nill ) {

q = r;

r = r->rite;

}

}

p->info = r->info; // put data in old node's place

remove( r, q ); // then remove the leaf

}

node \*

reshape( node \*root, unsigned int & n ) {

unsigned int fllg( unsigned int ), nu2( unsigned int );

node \*last[arraysize], \*p, \*q, \*stack[ssize], \*\*t;

unsigned int b, h, i, l;

char pat[arraysize];

if ( n < 1 )

return(nill);

b = n + 1 - ( 1 << (h = fllg(n)) );

b += b;

for ( l = 0 ; l <= h ; l++ ) {

last[l] = nill;

}

i = 0;

\*(t = stack) = root;

while ( stack <= t ) {

if ( (p = \*t--) != nill ) {

if ( p->left != nill ) {

if (stack + ssize - 2 <= t) {

cout << "\*\* reshape: stack overflow \*\*\n";

exit(1);

}

else {

q = p->left;

p->left = nill;

\*(++t) = p;

\*(++t) = q;

}

}

else {

i++;

l = h - nu2( i <= b ? i : i+i-b );

last[l] = p;

q = p->rite;

p->left = p->rite = nill;

pat[l] = (char)0;

if ( l < h ) {

if ( pat[l+1] == (char)0 ) {

p->left = last[l+1];

pat[l+1] = (char)1;

}

}

if ( l != 0 ) {

if ( last[l-1] ) {

if ( last[l-1]->rite == nill ) {

last[l-1]->rite = p;

pat[l] = (char)1;

}

}

}

if ( q != nill ) {

if ( stack + ssize - 1 <= t ) {

cout << "\*\* reshape: stack overflow \*\*\n";

exit( 1 );

}

\*(++t) = q;

}

}

}

}

return (\*last);

}

void

update( node \*& tree, unsigned int & n, ifstream & f, unsigned int & del, unsigned int & ins, unsigned int & maxh ) {

unsigned int fllg( unsigned int );

void remove( node \*, node \* ), fix( node \*&, unsigned int &, unsigned int &, unsigned int &, unsigned int & );

node \*p, \*q;

trec rec;

unsigned int chg, h, l, maxh0, n0;

chg = del = ins = 0;

maxh0 = (maxh = h = fllg(n)) + delta;

n0 = (1 << (h-1));

while ( f.read((char \*)&rec, sizeof(trec)) ) { // read records from transaction file to local record

q = nill;

p = tree;

l = 0;

while ( p != nill ) { // account exists

if ( rec.acct == p->info.acct ) {

if ( (p->info.baln += rec.tran) < 0.01 ) { // if account is empty or negative

del++;

chg++;

if ( (p != tree) || !isleaf(p) ) { // p is not whole tree or p is not leaf then remove

remove(p,q);

}

else {

cout << "\*\* tree is empty \*\*\n";

delete p;

tree = nill;

}

if ( --n < n0 ) {// balance less than 1 cent?

cout << "balance less than 1 cent\n";

fix(tree, n, h, maxh0, n0);

}

}

break;

}

else {

l++;

q = p;

if ( rec.acct < p->info.acct ) {

if ( p->left == nill ) {

}

p = p->left;

}

else {

p = p->rite;

}

}

}

if ( p == nill ) { // account does not exist

n++;

ins++;

chg++;

if ( h < ++l ) {

h = l;

if ( maxh < h )

maxh = h;

}

p = new node;

p->left = p->rite = nill; // copy info into new node

p->info.acct = rec.acct;

strcpy( p->info.name, rec.name );

p->info.baln = rec.tran;

if ( rec.acct < q->info.acct )

q->left = p;

else

q->rite = p;

}

if ( maxh0 < h )

fix(tree, n, h , maxh0, n0);

}

}

**Results**

>./updatecbt mf10000 tf1002

-- read 10000 recs into linked list --; 359.5 msec

-- made complete binary tree --; 82.9 msec

-- updated --:

2507 del; 2516 ins; 5023 changes; max height = 15; 423.1 msec

-- wrote 10009 recs to file "mf10001" --; 271.7 msec

-- chopped down tree --; 169.3msec

>./updatecbt mf20000 tf20002 nm20000

-- read 20000 recs into linked list --; 560.7 msec

-- made complete binary tree --; 91.5 msec

-- updated --:

5046 del; 4953 ins; 9999 changes; max height = 16; 941 msec

-- wrote 19907 recs to file "nm20000" --; 336.1 msec

-- chopped down tree --; 262.3msec

>./updatecbt mf40000 tf40002 nm40000

-- read 40000 recs into linked list --; 1127.1 msec

-- made complete binary tree --; 137.1 msec

-- updated --:

10049 del; 9882 ins; 19931 changes; max height = 17; 2498.8 msec

-- wrote 39833 recs to file "nm40000" --; 654.2 msec

-- chopped down tree --; 677.3msec

>./updatecbt mf80000 tf80002 nm80000

-- read 80000 recs into linked list --; 1800.9 msec

-- made complete binary tree --; 201.4 msec

-- updated --:

19965 del; 19997 ins; 39962 changes; max height = 18; 4716.9 msec

-- wrote 80032 recs to file "nm80000" --; 1218.1 msec

-- chopped down tree --; 1306.6msec