\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Report: HW6

Author: F74084729 林冠圻 <lltzpp@gmail.com>

Class: 資訊系112級 乙班

Description:

rdtsc: 計算 cpu cycle

cmp, checker, checker2: 比較資料

calc: 將 IP 四個數字計算成同一個

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Code:

// by. MiohitoKiri5474

#include<stdio.h>

#include<stdlib.h>

#define bool \_Bool

#define nullptr NULL

unsigned long long int begin, end, total = 0;

static unsigned long long rdtsc ( void ){

unsigned hi, lo;

\_\_asm\_\_ \_\_volatile\_\_ ( "rdtsc" : "=a" ( lo ), "=d" ( hi ) );

return ( ( unsigned long long ) lo ) | ( ( ( unsigned long long ) hi ) << 32 );

}

typedef struct node{

struct node \*nxt;

unsigned len, num;

} node;

node \*a[256], \*b[4096], \*c[4096];

bool cmp ( node \*x, node \*y ){

if ( x -> num == y -> num )

return x -> len > y -> len;

return x -> num > y -> num;

}

unsigned calc ( unsigned A, unsigned B, unsigned C, unsigned D ){

return A << 24 | B << 16 | C << 8 | D;

}

bool checker ( node o, unsigned u ){

return o.num >> 32 - o.len != u >> 32 - o.len;

}

bool checker2 ( node \*o, unsigned u, unsigned len ){

return o -> num != u || o -> len != len;

}

#define maxN 10000005

unsigned Tms = 0, Begin, End;

double Avg = 0.0;

bool search ( unsigned u ){

node \*o;

int A = 0, B = 0, C = 0;

for ( o = a[u >> 24] ; o && checker ( \*o, u ) ; o = o -> nxt );

if ( o )

A = 1;

for ( o = b[u >> 20] ; o && checker ( \*o, u ) ; o = o -> nxt );

if ( o )

B = 1;

for ( o = c[u >> 20] ; o && checker ( \*o, u ) ; o = o -> nxt );

if ( o )

C = 1;

return A || B || C;

}

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

unsigned A, B, C, D, len, u, id, begin, tms, success = 0, fail = 0, end, i;

double avg;

node \*o, \*l, \*swp = ( node\* ) malloc ( sizeof ( node ) );

// build

freopen ( argv[1], "r", stdin );

while ( ~scanf ( "%d.%d.%d.%d/%d", &A, &B, &C, &D, &len ) ){

swp = ( node\* ) malloc ( sizeof ( node ) );

id = u = calc ( A, B, C, D );

if ( len < 16 ){

id >>= 24, swp -> num = u, swp -> len = len;

if ( !a[id] ){

a[id] = swp, swp -> nxt = nullptr;

continue;

}

o = l = a[id];

while ( o && cmp ( o, swp ) )

l = o, o = o -> nxt;

if ( o == a[id] )

swp -> nxt = a[id], a[id] = swp;

else

l -> nxt = swp, swp -> nxt = o;

}

else if ( len < 24 ){

id >>= 20, swp -> num = u, swp -> len = len;

if ( !b[id] ){

b[id] = swp, swp -> nxt = nullptr;

continue;

}

o = l = b[id];

while ( o && cmp ( o, swp ) )

l = o, o = o -> nxt;

if ( o == b[id] )

swp -> nxt = b[id], b[id] = swp;

else

l -> nxt = swp, swp -> nxt = o;

}

else if ( len <= 32 ){

id >>= 20;

swp -> num = u;

swp -> len = len;

if ( !c[id] ){

c[id] = swp;

swp -> nxt = nullptr;

continue;

}

o = l = c[id];

while ( o && cmp ( o, swp ) )

l = o, o = o -> nxt;

if ( o == c[id] )

swp -> nxt = c[id], c[id] = swp;

else

l -> nxt = swp, swp -> nxt = o;

}

}

// search

freopen ( argv[2], "r", stdin );

while ( ~scanf ( "%d", &u ) ){

Begin = rdtsc(), Tms++;

search ( u ) ? success++ : fail++;

End = rdtsc();

Avg += End - Begin;

}

printf ( "After seg. table create\nsuccess search times = %d\nfail search times = %d\n", success, fail );

// insert

freopen ( argv[3], "r", stdin );

avg = 0.0, tms = 0;

while ( ~scanf ( "%d.%d.%d.%d/%d", &A, &B, &C, &D, &len ) ){

begin = rdtsc(), id = u = calc ( A, B, C, D ), tms++;

swp = ( node\* ) malloc ( sizeof ( node ) );

if ( len <= 15 ){

id >>= 24;

swp -> num = u, swp -> len = len;

if ( !a[id] ){

a[id] = swp;

swp -> nxt = nullptr;

continue;

}

o = l = a[id];

while ( o && cmp ( o, swp ) )

l = o, o = o -> nxt;

if ( o == a[id] )

swp -> nxt = a[id], a[id] = swp;

else

l -> nxt = swp, swp -> nxt = o;

}

else if ( len <= 23 ){

id >>= 20;

swp -> num = u, swp -> len = len;

if ( !b[id] ){

b[id] = swp;

swp -> nxt = nullptr;

continue;

}

o = l = b[id];

while ( o && cmp ( o, swp ) )

l = o, o = o -> nxt;

if ( o == b[id] )

swp -> nxt = b[id], b[id] = swp;

else

l -> nxt = swp, swp -> nxt = o;

}

else if ( len <= 32 ){

id >>= 20;

swp -> num = u, swp -> len = len;

if ( !c[id] ){

c[id] = swp;

swp -> nxt = nullptr;

continue;

}

o = l = c[id];

while ( o && cmp ( o, swp ) )

l = o, o = o -> nxt;

if ( o == c[id] )

swp -> nxt = c[id], c[id] = swp;

else

l -> nxt = swp, swp -> nxt = o;

}

end = rdtsc();

avg += end - begin;

}

avg /= tms \* 1.0;

// search

freopen ( argv[2], "r", stdin );

success = fail = 0;

while ( ~scanf ( "%d", &u ) ){

Begin = rdtsc(), Tms++;

search ( u ) ? success++ : fail++;

End = rdtsc();

Avg += End - Begin;

}

printf ( "\nAfter insertion\navg. insertion time = %lf cycles\nsuccess search times = %d\nfail search times = %d\n", avg, success, fail );

// delete

freopen ( argv[4], "r", stdin );

avg = 0.0, tms = 0;

while ( ~scanf ( "%d.%d.%d.%d/%d", &A, &B, &C, &D, &len ) ){

begin = rdtsc(), id = u = calc ( A, B, C, D ), tms++;

swp = ( node\* ) malloc ( sizeof ( node ) );

if ( len < 16 ){

id >>= 24;

o = l = a[id];

while ( o && checker2 ( o, u, len ) )

l = o, o = o -> nxt;

if ( o ){

if ( o == a[id] ){

a[id] = a[id] -> nxt;

free ( o );

continue;

}

l -> nxt = o -> nxt;

free ( o );

o = nullptr;

}

}

else if ( len < 24 ){

id >>= 20;

o = l = b[id];

while ( o && checker2 ( o, u, len ) )

l = o, o = o -> nxt;

if ( o ){

if ( o == b[id] ){

b[id] = b[id] -> nxt;

free ( o );

continue;

}

l -> nxt = o -> nxt;

free ( o );

o = nullptr;

}

}

else if ( len <= 32 ){

id >>= 20;

o = l = c[id];

while ( o && checker2 ( o, u, len ) )

l = o, o = o -> nxt;

if ( o ){

if ( o == c[id] ){

c[id] = c[id] -> nxt;

free ( o );

continue;

}

l -> nxt = o -> nxt;

free ( o );

o = nullptr;

}

}

end = rdtsc();

avg += end - begin;

}

avg /= tms \* 1.0;

// search - last time

freopen ( argv[2], "r", stdin );

success = fail = 0;

while ( ~scanf ( "%d", &u ) ){

Begin = rdtsc(), Tms++;

search ( u ) ? success++ : fail++;

End = rdtsc();

Avg += End - Begin;

}

Avg /= Tms \* 1.0;

printf ( "\nAfter deletion\navg. deletion time = %lf cycles\navg. search times = %lf cycles\nsuccess search times = %d\nfail search times = %d\n", avg, Avg, success, fail );

}

Compilation:

gcc -o hw8 hw8.c -lm -Ofast

Execution:

./hw8 prefix\_10K.txt trace\_IPaddress\_100K.txt insert\_1K.txt delete\_1K.txt

Output:

After seg. table create

success search times = 5080

fail search times = 0

After insertion

avg. insertion time = 209.071283 cycles

success search times = 5080

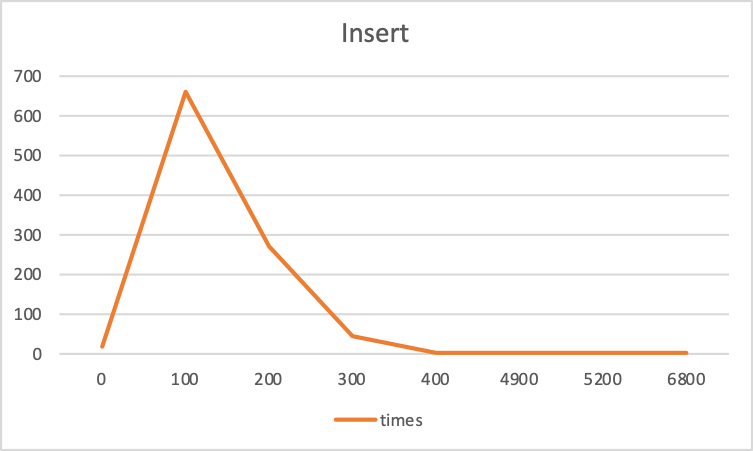
fail search times = 0

After deletion

avg. deletion time = 674.135751 cycles

avg. search times = 256.528150 cycles

success search times = 3930

 fail search times = 1150

