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

Report: HW8

Author: E94074029 江羿賢 <e94074029@gs.ncku.edu.tw>

Class: 資訊111級乙班

Description:

在這次的練習中學習到了linked list的應用，並且也會用了其中的函式insert delete search，並且利用ip的轉換來做這些操作，也在我的node中選擇了我所方便使用的資料型態來做，並且在insert時候一併儲存好，delete也得注意free，要不然有可能會出問題，也有另外去計算clock cycles並且畫圖，放在output之後。

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

#include <stdio.h>

#include <stdlib.h>

int takem(char \*input);

typedef struct node //linked list node

{

long long data;

int ip[33];

int them;

struct node \*next;

} node;

struct node \*insert(struct node \*first, long long input, char in[]) //the insert function

{

struct node \*temp = (struct node \*)malloc(sizeof(struct node));

char tem[4][4];

unsigned int n[4];

int store[4];

int row = 0;

int col = 0;

for (int i = 0; i < 16; i++)

{

if (in[i] != 0 && in[i] != 46)

{

tem[row][col] = in[i];

col++;

}

else

{

row++;

col = 0;

}

if (row == 4)

break;

}

for (int i = 0; i < 4; i++)

{

store[i] = atoi(tem[i]);

n[i] = \*(unsigned int \*)&store[i];

}

int index = 0;

for (int i = 0; i < 4; i++)

{

for (int j = 7; j >= 0; j--)

{

if ((n[i] >> j) & 1)

temp->ip[index] = 1; //printf("1");

else

temp->ip[index] = 0; //printf("0");

}

}

//

temp->them = takem(in);

//

temp->data = input;

temp->next = 0;

if (first == 0)

{

first = temp;

return first;

}

struct node \*current = first;

while (current->next != 0)

current = current->next;

current->next = temp;

return first;

}

struct node \*deletenode(struct node \*first, long long input) //the delete function

{

struct node \*current = first;

struct node \*previous = 0;

while (current != 0 && current->data != input)

{

previous = current;

current = current->next;

}

if (current == 0)

return first;

else if (current == first)

{

first = current->next;

free(current);

current = 0;

return first;

}

else

{

previous->next = current->next;

free(current);

current = 0;

return first;

}

}

int search(struct node \*first, long long input) //to search the node

{

int theip[33];

for (int i = 31; i >= 0; i--)

{

theip[i] = input % 2;

input /= 2;

}

int getout = 0;

struct node \*current = first;

while (current != 0 && current->data != input)

{

for (int i = 0; i <= current->them; i++)

{

if (current->ip[i] != theip[i])

getout++;

}

if (getout > 0)

break;

current = current->next;

}

if (current == 0)

return 0;

else

return 1;

}

void print(struct node \*first) //print the linked list(for testing)

{

if (first == 0)

return;

struct node \*current = first;

while (current != 0)

{

printf("%lld ", current->data);

current = current->next;

}

printf("\n");

}

unsigned long long power(int x, int y) //power function

{

unsigned long long total = 1;

while (y--)

{

total \*= x;

}

return total;

}

inline unsigned long long int rdtsc() //to count the clock

{

unsigned hi, lo;

\_\_asm\_\_ \_\_volatile\_\_("rdtsc"

: "=a"(lo), "=d"(hi));

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

}

long long takevalue(char \*input) //to get value of the ip

{

char temp[5][5];

int store[5];

int row = 0;

int col = 0;

long long sum = 0;

for (int i = 0; i < 20; i++)

{

if (input[i] != 0 && input[i] != 46)

{

temp[row][col] = input[i];

col++;

}

else

{

row++;

col = 0;

}

if (row == 5)

break;

}

for (int i = 0; i < 5; i++)

{

store[i] = atoi(temp[i]);

}

sum += store[0] \* 256 \* 256 \* 256;

sum += store[1] \* 256 \* 256;

sum += store[2] \* 256;

sum += store[3];

return sum;

}

int takeindex(char \*input) //to get the index in 8~15

{

char temp[5][5];

int store[5];

int row = 0;

int col = 0;

for (int i = 0; i < 20; i++)

{

if (input[i] != 0 && input[i] != 46 && input[i] != 47)

{

temp[row][col] = input[i];

col++;

}

else

{

row++;

col = 0;

}

if (row == 5)

break;

}

for (int i = 0; i < 5; i++)

{

store[i] = atoi(temp[i]);

}

return store[0];

}

int takeindex2(char \*input) //to get the index for the others

{

char temp[4][4];

unsigned int n[4];

int store[4];

int row = 0;

int col = 0;

int theindex = 0;

int exp = 11;

for (int i = 0; i < 16; i++)

{

if (input[i] != 0 && input[i] != 46)

{

temp[row][col] = input[i];

col++;

}

else

{

row++;

col = 0;

}

if (row == 4)

break;

}

for (int i = 0; i < 4; i++)

{

store[i] = atoi(temp[i]);

n[i] = \*(unsigned int \*)&store[i];

}

for (int j = 7; j >= 0; j--, exp--)

{

if ((n[0] >> j) & 1)

theindex += power(2, exp);

}

for (int j = 7; j >= 4; j--, exp--)

{

if ((n[1] >> j) & 1)

theindex += power(2, exp);

}

return theindex;

}

int takem(char \*input) //to get the m value

{

char temp[5][5];

int store[5];

int row = 0;

int col = 0;

for (int i = 0; i < 20; i++)

{

if (input[i] != 0 && input[i] != 46 && input[i] != 47)

{

temp[row][col] = input[i];

col++;

}

else

{

row++;

col = 0;

}

if (row == 5)

break;

}

for (int i = 0; i < 5; i++)

{

store[i] = atoi(temp[i]);

}

return store[4];

}

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

{

struct node \*t815[256]; //m 8~15

struct node \*t1623[4096]; //m 16~23

struct node \*t2432[4096]; //m 24~32

for (int i = 0; i < 256; i++) //initialize

t815[i] = 0;

for (int i = 0; i < 4096; i++)

{

t1623[i] = 0;

t2432[i] = 0;

}

//table

FILE \*fp;

fp = fopen(argv[1], "r");

char input[20];

int index = 0;

if (fp != NULL)

{

while (!feof(fp))

{ //to check the m to put in to linked list

fscanf(fp, "%s", input);

if (takem(input) >= 8 && takem(input) <= 15)

{

index = takeindex(input);

t815[index] = insert(t815[index], takevalue(input), input);

}

if (takem(input) >= 16 && takem(input) <= 23)

{

index = takeindex2(input);

t1623[index] = insert(t1623[index], takevalue(input), input);

}

if (takem(input) >= 24 && takem(input) <= 32)

{

index = takeindex2(input);

t2432[index] = insert(t2432[index], takevalue(input), input);

}

}

}

else

{

printf("Error");

fclose(fp);

return 0;

}

fclose(fp);

//table end

//search table

fp = fopen(argv[2], "r");

int count = 0;

int su = 0;

int fa = 0;

int temp1 = 0;

int temp = 0;

long long number = 0;

if (fp != NULL)

{

int begin, end;

begin = rdtsc();

while (!feof(fp))

{ //turn the number to ip to check the site and do search

fscanf(fp, "%lld", &number);

int origin = number;

long long i, k;

int exp = 0;

int exp1 = 0;

for (i = 1; i <= 32; i++)

{

k = number % 2;

number /= 2;

if (i >= 21)

{

if (k == 1)

temp += power(2, exp);

exp++;

}

if (i >= 25)

{

if (k == 1)

temp1 += power(2, exp1);

exp1++;

}

}

//found the site and do search

count += search(t2432[temp], origin);

count += search(t1623[temp], origin);

count += search(t815[temp1], origin);

//count the success and fail

if (count > 0)

su++;

else

fa++;

count = 0;

temp = 0;

temp1 = 0;

exp = 0;

exp1 = 0;

}

end = rdtsc();

printf("After seg. table create\n");

printf("success search times= %d\nfail search times= %d\n\n", su, fa);

su = 0;

fa = 0;

}

else

{

printf("Error");

fclose(fp);

return 0;

}

fclose(fp);

//search table end

//insertion

int begin = 0;

int end = 0;

int thet = 0;

fp = fopen(argv[3], "r");

begin = rdtsc();

if (fp != NULL)

{ //insert to linked list

while (!feof(fp))

{

fscanf(fp, "%s", input);

if (takem(input) >= 8 && takem(input) <= 15)

{

index = takeindex(input);

t815[index] = insert(t815[index], takevalue(input), input);

}

if (takem(input) >= 16 && takem(input) <= 23)

{

index = takeindex2(input);

t1623[index] = insert(t1623[index], takevalue(input), input);

}

if (takem(input) >= 24 && takem(input) <= 32)

{

index = takeindex2(input);

t2432[index] = insert(t2432[index], takevalue(input), input);

}

thet++;

}

}

else

{

printf("Error");

fclose(fp);

return 0;

}

fclose(fp);

end = rdtsc();

printf("After insertion\n");

printf("avg. insertion time= %d cycles\n", (end - begin) / (thet)); //print the cycles

//insertion end

//insertion search

fp = fopen(argv[2], "r");

if (fp != NULL)

{

int begin, end;

begin = rdtsc();

while (!feof(fp))

{ //do search

fscanf(fp, "%lld", &number);

int origin = number;

long long i, k;

int exp = 0;

int exp1 = 0;

for (i = 1; i <= 32; i++)

{

k = number % 2;

number /= 2;

if (i >= 21)

{

if (k == 1)

temp += power(2, exp);

exp++;

}

if (i >= 25)

{

if (k == 1)

temp1 += power(2, exp1);

exp1++;

}

}

//find the site and search

count += search(t2432[temp], origin);

count += search(t1623[temp], origin);

count += search(t815[temp1], origin);

if (count > 0)

su++;

else

fa++;

count = 0;

temp = 0;

temp1 = 0;

exp = 0;

exp1 = 0;

}

end = rdtsc();

printf("success search times= %d\nfail search times= %d\n\n", su, fa); //find the site and search

su = 0;

fa = 0;

}

else

{

printf("Error");

fclose(fp);

return 0;

}

//insertion search end

//deletion

fp = fopen(argv[4], "r");

index = 0;

if (fp != NULL)

{

int begin = 0, end = 0;

int t = 0;

begin = rdtsc();

while (!feof(fp))

{ //do delete

fscanf(fp, "%s", input);

if (takem(input) >= 8 && takem(input) <= 15)

{

index = takeindex(input);

t815[index] = deletenode(t815[index], takevalue(input));

}

if (takem(input) >= 16 && takem(input) <= 23)

{

index = takeindex2(input);

t1623[index] = deletenode(t1623[index], takevalue(input));

}

if (takem(input) >= 24 && takem(input) <= 32)

{

index = takeindex2(input);

t2432[index] = deletenode(t2432[index], takevalue(input));

}

t++;

}

end = rdtsc();

printf("After deletion\n");

printf("avg. deletion time= %d cycles\n", (end - begin) / t); //count cycle

}

else

{

printf("Error");

fclose(fp);

return 0;

}

fclose(fp);

//deletion end

//search deletion

fp = fopen(argv[2], "r");

if (fp != NULL)

{

int begin = 0, end = 0;

begin = rdtsc();

while (!feof(fp))

{ //do search after deletion

fscanf(fp, "%lld", &number);

int origin = number;

long long i, k;

int exp = 0;

int exp1 = 0;

for (i = 1; i <= 32; i++)

{

k = number % 2;

number /= 2;

if (i >= 21)

{

if (k == 1)

temp += power(2, exp);

exp++;

}

if (i >= 25)

{

if (k == 1)

temp1 += power(2, exp1);

exp1++;

}

}

//find the site and do search

count += search(t2432[temp], origin);

count += search(t1623[temp], origin);

count += search(t815[temp1], origin);

if (count > 0)

su++;

else

fa++;

count = 0;

temp = 0;

temp1 = 0;

exp = 0;

exp1 = 0;

}

end = rdtsc();

printf("avg. search times= %d cycles\n", (end - begin) / (su + fa)); //count cycle

printf("success search times= %d\nfail search times= %d\n\n", su, fa);

su = 0;

fa = 0;

}

else

{

printf("Error");

fclose(fp);

return 0;

}

//search deletion end

}

Compilation:

g++ hw8.c -o hw8

(用gcc，clock的function有小問題)

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= 1

After insertion

avg. insertion time= 7545 cycles

success search times= 5080

fail search times= 1

After deletion

avg. deletion time= 4355 cycles

avg. search times= 2646 cycles

success search times= 5020

fail search times= 61





