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

Report: HW8

Author: E64061151 林友鈞 <[E64061151@mail.ncku.edu.tw](mailto:wang@xiaoming.tw)>

Class: 資訊系二乙

Description:

我從這份程式作業中學到如何使用linked list和function pointer以及如何使用linked list做快速排序。以下將解釋程式中各物件以及函數的功能及實作方法。

**typedef struct Node \* NodePointer**:宣告NodePointer型態

**struct Node** 宣告Node物件型態，他的data member 有int val和 NodePointer next。val儲存物件的值，next則指向下一個node。

**int RandomInt(int m)**：函數回傳亂數，亂數會介在0 ~ m-1 之間。

**NodePointer nodeConstructor(int num):**

Node物件的Constructor，函數會製造一個node，並做初始化：val = num; next = NULL;並將node物件回傳。

Parameter:

int num: node->val = num;給予node初始值。

**NodePointer createRandomList(int amounts, int m):**

製造亂數linked list。並將head回傳

Parameter：

int amounts:linked list需要幾個nodes。

int m : node的值會介在0 ~ m-1之間。

**void Show(NodePointer head)**：

將linked list的head node傳入，則將node的val print出。

**void quickSort(void \* list,unsigned int size, int cmpr(void \* , void \*),void operation(void \* , void \*),void \* iterator(void \*))**

quickSort的主程式。

Parameter:

void \* list:傳入linked list的head。

unsigned int size:linked list 要被sorted的長度。

int cmpr(void \* ,void \*)：傳入的是一個function pointer。需要回傳兩nodes比較的結果。

void operation(void \*, void \* ):傳入的是一個function pointer，主要的功能是兩nodes比較後需要做什麼處理。

void \* iterator(void \*) : 傳入的是一個function pointer，功能是要如何換到下一個node。

程式邏輯：

Quicksort選擇的pivot是第一個node，iter一開始指向pivot。iter會迭代size - 1次意思是說當所有的node都被scan過時for迴圈就會結束。

在迭代的過程，cmpr(iter ,pivot)如果回傳1代表需要做交換或插入，因為程式撰寫的方便性，cmpr(iter, pivot)會比較的是iter->next->val與pivot->val的大小關係，而不是iter->val與pivot->val的大小關係。

operation(iter, list)同樣因為程式撰寫的方便性iter->next會被插在list的前面。

當for迴圈結束時，linked list會以pivot為界。pivot左邊必定小於(大於)pivot，pivot右邊必定大於(小於)pivot。

程式中變數int new\_size記錄的是pivot前有幾個node。

因此最後遞迴呼叫的時候是quickSort(list,new\_size,cmpr,operation,iterator);左半邊

quickSort(pivot,size-1-new\_size,cmpr,operation,iterator);右半邊

**int NodeCmpr(void \* v\_node1, void \* v\_node2):**

比較的是v\_node1->next->val與v\_node2->val的大小關係。

**void nodeInsertion(void \* v\_node1,void\* v\_node2)**：

v\_node1->next會被inserted在v\_node2之後。

**void \* NodeIterator(void \* node)**：

回傳node->next。

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

Code:

#include <stdlib.h>

#include <stdio.h>

#include <time.h>

typedef struct Node \* NodePointer;

struct Node{

int val;

NodePointer next;

};

typedef struct Node Node;

int RandomInt(int m){

return (rand()%m);

}

NodePointer nodeConstructor(int num){

NodePointer newNode = (NodePointer)malloc(sizeof(Node));

newNode->val = num;

newNode->next = NULL;

return newNode;

}

NodePointer createRandomList(int amounts,int m){

if(amounts){

NodePointer np;

NodePointer iterator;

iterator = np = nodeConstructor(-1);

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

iterator->next = nodeConstructor(RandomInt(m));

iterator = iterator->next;

}

return np;

}

return NULL;

}

void Show(NodePointer head){

NodePointer iterator = head->next;

while(iterator){

printf("%2d",iterator->val);

iterator = iterator->next;

}

printf("\n");

}

int NodeCmpr(void \* v\_node1, void \* v\_node2){

return ((NodePointer)v\_node1)->next->val < ((NodePointer)v\_node2)->val;

}

void \* NodeIterator(void \* node){

NodePointer Next = ((NodePointer)node)->next;

return (void \*)Next;

}

void nodeInsertion(void \* v\_node1,void\* v\_node2){

// node1->next will be insert after node2

NodePointer node2 = (NodePointer)v\_node2;

NodePointer node1 = (NodePointer)v\_node1;

NodePointer temp ;

temp = node2->next;

node2->next = node1->next;

node1->next = node2->next->next;

node2->next->next = temp;

}

void quickSort(void \* list,unsigned int size, int cmpr(void \* , void \*),void operation(void \* , void \*),void \* iterator(void \*)){

void \* pivot = iterator(list);

void \* iter = pivot;

int i;

int new\_size = 0;

if(size > 1){

for(i = 0; i < size - 1; i++){

if(cmpr(iter,pivot)){

operation(iter,list);

++new\_size;

}else iter = iterator(iter);

}

quickSort(list,new\_size,cmpr,operation,iterator);

quickSort(pivot,size-1-new\_size,cmpr,operation,iterator);

}else return;

}

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

int num = atoi(argv[1]);

srand(atoi(argv[3]));

NodePointer head = createRandomList(num,atoi(argv[2]));

printf("Original SLL:");

Show(head);

float time\_1 = clock();

quickSort(head,num,NodeCmpr,nodeInsertion,NodeIterator);

float time\_2 = clock();

// printf("time consume : %f\n",(time\_2 - time\_1)/CLOCKS\_PER\_SEC);

printf("Sorted SLL:");

Show(head);

}

Compilation:

gcc -std=c99 -o hw8 hw8.c

Execution:

./hw8 n m s

Output:

>./hw8 10 10 10

Original SLL: 5 8 8 5 8 7 5 1 7 7

Sorted SLL: 1 5 5 5 7 7 7 8 8 8