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

Report: HW0\_8

Author: F74071140 謝育萱 <[shirleysensible@gmail.com](mailto:shirleysensible@gmail.com)>

Class: 甲班

Description:

這次是linked list的用法練習，基本上應該要能直接寫出順勢的list，但不知怎麼搞得我就是會寫出segmentation fault，因此我後來又加了一個反轉函式把該list反過來；接著就是quicksort，c語言中本來就有qsort這個內建函式，但它只對陣列有效……，quicksort簡單來說就是取一個pivot值並做比較，比pivot小的放它左邊，比pivot大的放右邊，一直不斷地取值比較，最後便可以得到依序排列的list。

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

#include<stdio.h>

#include<stdlib.h>

struct node{

int value;

struct node \*next;

};

struct node \*reverse(struct node\*); //把原list倒過來

struct node \*print(struct node\*); //列印list

struct node\* q\_sort(struct node\*, struct node\*);

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

int n=atoi(argv[1]),m=atoi(argv[2]),s=atoi(argv[3]),num,i=0;

/\*n為有幾個元素 , m為數字範圍 , s為隨機變數依據\*/

srand(s);

struct node \*head=NULL, \*pivot=NULL;

//先建立link list開頭

for(i=0;i<n;i++){

num=rand()%m;

struct node \*new\_node=malloc(sizeof(struct node));

//開新空間 其size為已宣告的struct大小

new\_node->value=num;

//存隨機變數}

new\_node->next=head;

head=new\_node;

}

printf("Oringinal SLL:");

head = reverse(head);

print(head);

head=q\_sort(head,NULL);

printf("Sorted SLL:");

while(1){

if(head==NULL) break;

else{

printf("%4d",head->value);

head=head->next;

}

}

printf("\n");

return 0;

}

struct node\* reverse(struct node\* head){

if(head==NULL||head->next==NULL)

return head;

struct node \*previous=NULL, \*now=head, \*preceding=head->next;

while(preceding!=NULL){

now->next=previous; //把now->next轉向

previous=now; //previous往後挪

now=preceding; //now往後挪

preceding=preceding->next; //preceding往後挪

}

now->next=previous; //此時current位於最後一個node

head=now; //更新head為now

return head;

}

struct node \* print(struct node\* head){

struct node \*current=head;

while(1){

if(current==NULL) break;

printf("%4d",current->value);

current=current->next; }

printf("\n");

}

struct node\* q\_sort(struct node \*head,struct node \*tail)

{

if(head == tail || head->next == tail)

return head;

struct node \*pivot = head, \*current=NULL, \*previous, \*pre\_pivot = NULL;

for(previous = NULL, current = head; current != tail; previous = current, current = current->next)

{

if(current->value < pivot->value){

if(pivot == head) {

previous->next = current->next;

current->next = pivot;

head = pre\_pivot = current;

current = previous;

}

else

{

previous->next = current->next;

current->next = pivot;

pre\_pivot->next = current;

pre\_pivot = pre\_pivot->next;

current = previous;

}

}

}

head = q\_sort(head, pivot);

pivot->next = q\_sort(pivot->next, tail);

return head;

}

另外

struct node\* find\_p(struct node\* head){

struct node \*pivot=head, \*count=head;

int i, num, length=0;

for(;count!=NULL;count=count->next)

length++; //計算輸入的list長度

num=rand()%length; //隨機取要第num個做pivot

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

pivot=pivot->next; //開始將pivot移到要的位置

}

return pivot; //回傳pivot指摽

}

struct node\* q\_sort(struct node \*head){

struct node \*compare=NULL, \*left=NULL, \*right=NULL;

struct node \*pivot=find\_p(head);

//printf("pivot:%d\n",pivot->value);

if(head==NULL||head->next==NULL)

return head;

for(compare=head;compare!=NULL;){

compare=compare->next;

if(compare!=pivot){

if(compare->value<pivot->value){

compare->next=left;

left=compare;

}else{

compare->next=right;

right=compare;

}

}

compare=compare->next;

}

compare=concat(q\_sort(left),pivot,q\_sort(right));

return compare;

}