数据结构课程设计



班级： 1618403

学号： 161840227

姓名： 韦 鑫

指导教师：孙 涵

目录

1.采用的数据结构 ………………………………………… 3

2.算法设计思想 …………………………………………… 3

3.关键代码 ………………………………………………… 3

4.测试数据和结果 ………………………………………… 7

5.算法的时间复杂度及其改进方法 ……………………… 8

6.结束语 …………………………………………………… 9

一、采用的数据结构

链式队列

struct node

{

int key,time;

node \*next;

};

//链式队列存储取、放钥匙的事件

struct Queue

{

node \*front;

};

二、算法设计思想

定义两个链式队列用于存放钥匙的“取”、“放”时间，事件插入链表的规则为：当时间相同时，按照钥匙编号从小到大的顺序插入，否则按照时间的顺序插入。这样做可以使得每次出队的“取”、“放”时间符合题目要求。

三、关键代码

//运行环境：Visual Studio 2017

#include<cstdio>

#include<iostream>

#include<cstdlib>

#pragma warning(disable:4996)

using namespace std;

const int maxn = 100005;

struct node

{

int key,time;

node \*next;

};

//链式队列存储取、放钥匙的事件

struct Queue

{

node \*front;

};

//初始化取、放钥匙的列表

void init(Queue &get, Queue &put)

{

get.front = (node \*)malloc(sizeof(node));

get.front->next = NULL;

put.front = (node \*)malloc(sizeof(node));

put.front->next = NULL;

}

//判断是否为空

bool empty(Queue Q)

{

if (Q.front->next == NULL)return true;

else return false;

}

//获取队列首元素

node \*gethead(Queue Q)

{

if (!empty(Q))return Q.front->next;

}

//入队，当时间相同时按照编号大小排序，否则按照时间排序

void enque(Queue &Q,int key,int time)

{

//p为插入位置的前驱

node \*p = Q.front;

//按时间顺序

while (p->next != NULL && p->next->time < time)p = p->next;

//时间相同时按照钥匙编号大小

while (p->next != NULL && p->next->time == time && p->next->key < key)p = p->next;

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

s->key = key;

s->time = time;

s->next = p->next;

p->next = s;

}

//出队

void deque(Queue &Q)

{

if (!empty(Q))

{

node \*p = Q.front->next;

Q.front->next = p->next;

free(p);

}

}

//打印队列状态

void stat\_queue(Queue get,Queue put)

{

node \*p = get.front->next;

printf("获取钥匙事件队列状态：");

if(p != NULL)printf("\n钥匙编号 获取时间\n");

else printf("无事件\n");

while (p != NULL)

{

printf("%-5d %-5d\n", p->key, p->time);

p = p->next;

}

p = put.front->next;

printf("归还钥匙事件队列状态：");

if(p != NULL)printf("\n钥匙编号 归还时间\n");

else printf("无事件\n");

while (p != NULL)

{

printf("%-5d %-5d\n", p->key, p->time);

p = p->next;

}

printf("-------------------------------------\n");

}

//打印钥匙盒状态

void stat\_key(int a[],int n)

{

printf("钥匙盒状态：\n");

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

{

if (a[i] == -1)printf("x ");

else printf("%d ", a[i]);

}

printf("\n");

}

int main()

{

FILE \*fp;

if ((fp = fopen("time.txt", "r")) == NULL)

{

printf("打开文件失败！\n");

exit(0);

}

Queue get, put;

init(get, put);

int n, k;

int a[maxn];//存放钥匙编号

int start = maxn, end = -1;//开始时间和结束时间

fscanf(fp,"%d%d", &n, &k);

for (int i = 1; i <= n; i++)a[i] = i;//初始化钥匙编号

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

{

int key, s, con;

fscanf(fp,"%d%d%d", &key, &s, &con);

if (s < start)start = s;

if (s + con > end)end = s + con;

enque(get, key, s); //取钥匙事件加入队列

enque(put, key, s + con); //放钥匙事件加入队列

}

fclose(fp);

//打印队列和钥匙盒初始状态

stat\_key(a, n);

stat\_queue(get, put);

for (int i = start; i <= end; i++)

{

//执行放钥匙事件

while (!empty(put))

{

node \*head = gethead(put);

if (head->time == i)

{

for (int j = 1; j <= n; j++)

{

if (a[j] == -1)

{

a[j] = head->key;

deque(put);

stat\_key(a, n);

stat\_queue(get, put);

break;

}

}

}

else break;

}

//执行取钥匙事件

while (!empty(get))

{

node \*head = gethead(get);

if (head->time == i)

{

for (int j = 1; j <= n; j++)

{

if (a[j] == head->key)

{

a[j] = -1;

deque(get);

stat\_key(a, n);

stat\_queue(get, put);

break;

}

}

}

else break;

}

}

printf("钥匙盒最终状态：");

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

{

printf("%d ", a[i]);

}

system("pause");

return 0;

}

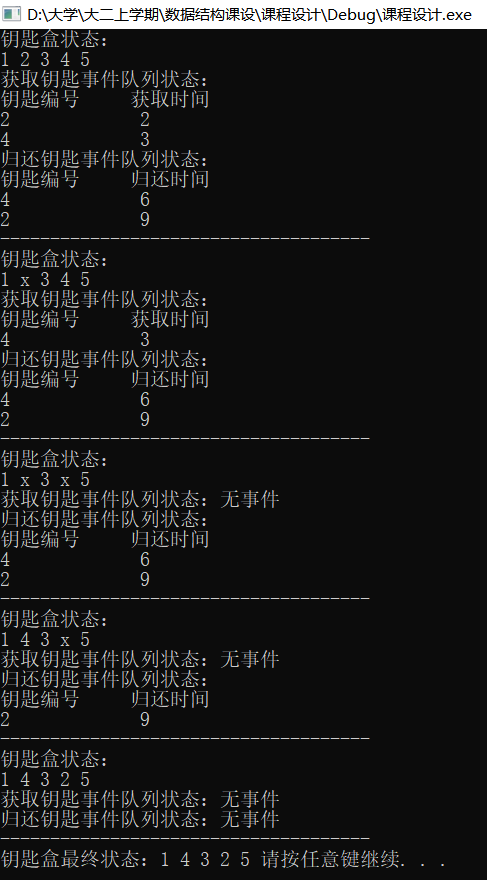
四、测试数据和结果

time.txt：

5 2

4 3 3

2 2 7



五、算法的时间复杂度即改进方法

链式队列入队的时间复杂度：O(n)

六、结束语

代码共180行

我认为该实验主要考察的是对链式队列有顺序的插入操作。我在给队列初始化时就踩了坑，没让队列的front指针置为NULL，导致了在插入第一个元素时出错。以后在使用链式队列时一定要谨记此次出现的错误，少在简单细节上犯错！