**暨南大学本科实验报告专用纸**

课程名称 算法分析与设计实验 成绩评定

实验项目名称 指导教师 李展 实验地点 N116

实验项目编号 7-7 实验项目类型 综合型

学生姓名 张瑞鹏 学号 2020101124

学院 信息科学技术学院 系 计算机 专业 计算机科学与技术

实验时间 2022 年 月 日 午～ 月 日 下 午

1. **问题描述**

**有序链表的几个操作，利用舍伍德算法降低时间复杂度。**

**（二）算法思路**（用文字简单说明）

**正常而言，start是从首个元素，end是链表的末尾，通过产生随机数，sqrt（n）次，缩小的搜索的范围，从而降低时间按复杂度。**

1. **对于找后继元素，算法最核心的还是缩小搜索范围，随机产生一个随机数，查看该位置，是属于查找元素之后的元素，还是查找元素之前的元素，再进一步判断，该元素能否起到缩小范围的作用，若果可以，就将他替换成新的起点（终点）。**
2. **剩下的就是在新的范围执行我们耳熟能详的查找操作。**

**（三）算法实施步骤和流程**（伪代码/流程图等方式描述）

void processor(int \*value,int \*link,int n) {

cout << endl << "please tell which number you want to search for processor:";

int destination;

cin >> destination;

int time = sqrt(n);

int start = link[0], end = 0;

int min = 0, max = 100;

for (int i = 1; i < time; i++) {

int j = rand();

j = j % n;

if (value[j]<n && value[j]>min) {

min = value[j];//底数改成这个数字

start = j;//起始指针改成这个

}

if (value[j] > n && value[j] < max) {

max = value[j];

end = j;

}

}

int flag = 0;

for (int pointer = start; pointer != end; ) {

int next = link[pointer];

if (value[next] == destination) {

cout << "the processor is:" << value[pointer]<<endl;

flag = 1;

break;

}

if (value[next] > destination) {

cout << "no element to match what you input!" << endl;

break;

}

pointer = next;

}

if(flag==0) cout<<"no element to match what you input!" << endl;

return;

}

void successor(int\* value, int\* link, int n) {

cout << endl << "please tell which number you want to search for successor:";

int destination;

cin >> destination;

int time = sqrt(n);

int start = link[0], end = 0;

int min = 0, max = 100;

for (int i = 1; i < time; i++) {

int j = rand();

j = j % n;

if (value[j]<n && value[j]>min) {

min = value[j];//底数改成这个数字

start = j;//起始指针改成这个

}

if (value[j] > n && value[j] < max) {

max = value[j];

end = j;

}

}

int flag = 0;

for (int pointer = start; pointer != end; ) {

int next = link[pointer];

if (value[pointer] == destination) {

cout << "the processor is:" << value[next] << endl;

flag = 1;

break;

}

if (value[next] > destination) {

cout << "no element to match what you input!" << endl;

break;

}

pointer = next;

}

if (flag == 0) cout << "no element to match what you input!" << endl;

return;

}

void max(int\* value, int\* link, int n) {

int time = sqrt(n);

int start = link[0], end = 0;

int min = 0;

for (int i = 1; i < time; i++) {

int j = rand();

j = j % n;

if (link[j]==0) {

cout << "the max is :" << value[j]<<endl;

return;

}

else if (value[j] > min) {

min = value[j];

start = j;

}

}

int pointer = start;

for (; link[pointer] != end; ) pointer = link[pointer];

cout << "the max is :" << value[pointer] << endl;

return;

}

**（四）源代码**（通过了编译运行的正确程序）

#include<iostream>

#define INFINITY 100

using namespace std;

void processor(int \*value,int \*link,int n) {

cout << endl << "please tell which number you want to search for processor:";

int destination;

cin >> destination;

int time = sqrt(n);

int start = link[0], end = 0;

int min = 0, max = 100;

for (int i = 1; i < time; i++) {

int j = rand();

j = j % n;

if (value[j]<n && value[j]>min) {

min = value[j];//底数改成这个数字

start = j;//起始指针改成这个

}

if (value[j] > n && value[j] < max) {

max = value[j];

end = j;

}

}

int flag = 0;

for (int pointer = start; pointer != end; ) {

int next = link[pointer];

if (value[next] == destination) {

cout << "the processor is:" << value[pointer]<<endl;

flag = 1;

break;

}

if (value[next] > destination) {

cout << "no element to match what you input!" << endl;

break;

}

pointer = next;

}

if(flag==0) cout<<"no element to match what you input!" << endl;

return;

}

void successor(int\* value, int\* link, int n) {

cout << endl << "please tell which number you want to search for successor:";

int destination;

cin >> destination;

int time = sqrt(n);

int start = link[0], end = 0;

int min = 0, max = 100;

for (int i = 1; i < time; i++) {

int j = rand();

j = j % n;

if (value[j]<n && value[j]>min) {

min = value[j];//底数改成这个数字

start = j;//起始指针改成这个

}

if (value[j] > n && value[j] < max) {

max = value[j];

end = j;

}

}

int flag = 0;

for (int pointer = start; pointer != end; ) {

int next = link[pointer];

if (value[pointer] == destination) {

cout << "the processor is:" << value[next] << endl;

flag = 1;

break;

}

if (value[next] > destination) {

cout << "no element to match what you input!" << endl;

break;

}

pointer = next;

}

if (flag == 0) cout << "no element to match what you input!" << endl;

return;

}

void max(int\* value, int\* link, int n) {

int time = sqrt(n);

int start = link[0], end = 0;

int min = 0;

for (int i = 1; i < time; i++) {

int j = rand();

j = j % n;

if (link[j]==0) {

cout << "the max is :" << value[j]<<endl;

return;

}

else if (value[j] > min) {

min = value[j];

start = j;

}

}

int pointer = start;

for (; link[pointer] != end; ) pointer = link[pointer];

cout << "the max is :" << value[pointer] << endl;

return;

}

int main(void) {

int value[8] = { INFINITY,2,3,13,1,5,21,8 };

int link[8] = { 4,2,5,6,1,7,0,3 };//两个数组初始化

cout << "test example:" << endl;

char c = '~';

for (int i = 0; i < 8; i++) {

if (value[i] == INFINITY) printf("%-3c",c );

else printf("%-3d", value[i]);

}

cout << endl;

for (int i = 0; i < 8; i++) printf("%-3d", link[i]);

processor(value, link, 7);

successor(value, link, 7);

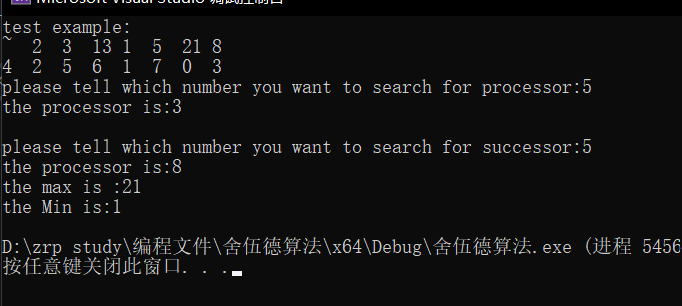
max(value, link, 7);

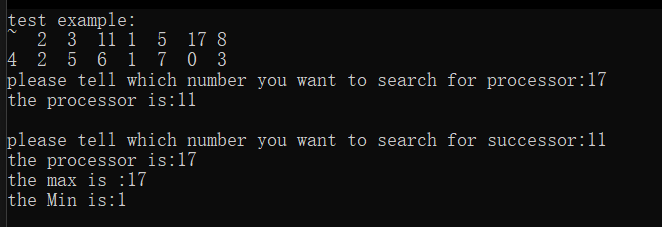
cout << "the Min is:" << value[link[0]] << endl;

return 0;

}

**（五）测试结果**（至少有两个以上算例及程序运行结果，截图贴进实验报告）





**（六）实验总结**（至少三句话，可以写复杂度分析、遇到问题、可能的改进措施、心得体会等）

1.对于有序链表而言，最小的元素就在链表的最最前面，直接而用value[link[0]]读取就可以了，时间复杂度就是O(1);

2.对于找先驱元素，和找后继元素的时间复杂度其实都是和查找是一样的，相当于查找该元素的时间复杂度。由于做了sqrt(n)的优化，平均会缩搜索空间（n/(sqrt(n)+1)）,化简之后便是O（sqrt(n)）,这便是一个很好的时间复杂度的提升。

3.我认为舍伍德算法的思路和常规人性化的思路有点相似，人在做一些事情的时候，也总是会随机抽样一个进行尝试，然后，在进行下一步的继续尝试。

**暨南大学本科实验报告专用纸(附页)**