**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 4-1 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

**（一）问题描述**

**假设要在足够多的会场里安排一批活动，并希望使用尽可能少的会场。设计一个有效的贪心算法进行安排。对于给定的k个待安排的活动，计算使用最少会场的时间表。**

**（二）算法思路**（用文字简单说明）

**方法一：**

**按照开始时间排序，每次选择最早的，若满足该最早的元素的开始时间晚于上一个任务的结束时间，则把该元素加入当前集合当中，当前集合已经无法再融入其他元素时，开启一个新的集合重复上述步骤，直到所有元素都被选中。**

**方法二：**

**遍历一次有序数组，若该元素能放入当前任意一个会场，则放入，否则开启新的集合存放该元素。**

**贪心选择：**

**先对开始时间早的活动进行安排：假设跳过最先开始的时间进行安排，完成后续安排后再考虑最早开始的活动。由于最早开始的活动比任意活动都要早，不可能在末尾添加，只可能在最前面添加，且需要保证会场最小化，需要比较最早开始活动的结束时间与所有会场开始时间，若有会场能够放入则放入，没有则开一个新的集合。完全等效于一开始对最早的活动进行安排。所以证得贪心选择得正确性。**

**最优子结构：**

**假设现在不考虑第一个元素，子问题有最优解，加第一个元素，由于是按开始时间排序的，第一个元素一定是在所有集合的最前面插入，如果能插入，则插入，不能插入则构建一个新的集合。这样一来的结果也是原问题的最优解。因此具有最优子结构。**

**（三）算法实施步骤和流程**（伪代码/流程图等方式描述）

写法一：

**int result = 0;**

**int flag = info.listnum;**

**while (flag > 0) {**

**result++;**

**int nowtime = 0;**

**for (int i = 0; i < info.listnum; i++) {**

**if (info.list[i].choose != 1&&info.list[i].startt>=nowtime) {**

**info.list[i].choose = 1;**

**nowtime = info.list[i].endtime;**

**flag--;**

**}**

**}**

**}**

**写法二：**

int minplace(imfomation info) {

sort(info.list,0,info.listnum-1);//进行快速排序

int result = 0;

int place[20] = {0};

for (int i = 0; i < info.listnum; i++) {

int flag = 0;

for (int j = 0; j < result; j++) {

if (info.list[i].startt > place[j]) {

flag = 1;

place[j] = info.list[i].endtime;

break;

}

}

if (flag == 0) {

place[result] = info.list[i].endtime;

result++;

}

}

return result;

}

**（四）源代码**（通过了编译运行的正确程序）

#include<iostream>

using namespace std;

struct activity {

int startt;

int endtime;

int choose;

};

struct imfomation {

activity\* list;

int listnum;

};

int partition(activity\* list, int start, int end) {

if (start == end) return start;

activity buf = list[start];//要是后期需要改动的话就把所有的这部分进行替换

while (start < end){

while (start<end && list[end].startt>=buf.startt) end--;

list[start] = list[end];

while (start < end && list[start].startt <= buf.startt) start++;

list[end] = list[start];

}

list[start] = buf;

return start;

}

void sort(activity\* list,int start,int end) {//一结构体中的start作为key进行排序

int middle = partition(list, start, end);

if (start < middle-1) sort(list, start, middle - 1);

if (end > middle + 1) sort(list, middle + 1, end);

}

int minplace(imfomation info) {

sort(info.list,0,info.listnum-1);//进行快速排序

int result = 0;

int flag = info.listnum;

while (flag > 0) {

result++;

cout << "第" <<result<<"会场"<< endl;

int nowtime = 0;

for (int i = 0; i < info.listnum; i++) {

if (info.list[i].choose != 1&&info.list[i].startt>=nowtime) {

info.list[i].choose = 1;

nowtime = info.list[i].endtime;

flag--;

cout << "活动：";

printf("%-5d%-5d\n", info.list[i].startt, info.list[i].endtime);

//out << "开始时间:" << info.list[i].startt << "结束时间:" << info.list[i].endtime << endl;

}

}

}

return result;

return 0;

}

int main(void) {

int n = 7;

imfomation info;

info.list = new activity[n];

info.listnum = n;

info.list[0] = { 1,23,0 };

info.list[1] = { 12,28,0 };

info.list[2] = { 25,35,0 };

info.list[3] = { 27,80,0 };

info.list[4] = { 36,50,0 };

info.list[5] = { 2,60,0 };

info.list[6] = { 51,90,0 };

cout << "测试样例" << endl;

cout << n << endl;

for (int i = 0; i < info.listnum; i++) {

printf("%-5d%-5d\n", info.list[i].startt, info.list[i].endtime);

}

cout << "\n\n\n";

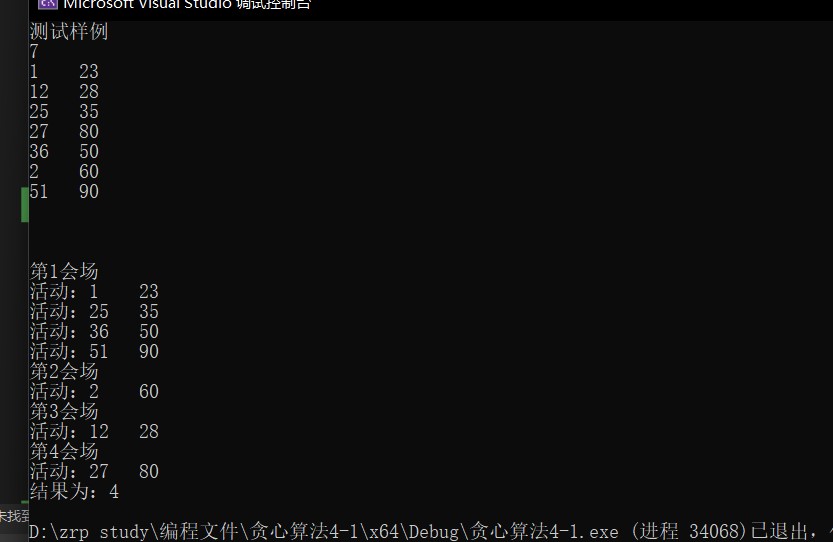
int result = minplace(info);

cout << "结果为：" << result << 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.贪心算法的时间复杂度大多是来源于排序算法，贪心算法本身而言比较简单。

2.采用了快速排序的算法，这样的话可以更好的提高排序的速度。使得代码效率提高。

3.使用结构体，使得排序更加的清晰

4.贪心算法感觉重在证明，其实代码的书写比较简单，难在如何证明算法的正确性

**暨南大学本科实验报告专用纸(附页)**