**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 4-7 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

**（一）问题描述**

**设有 n 个顾客同时等待一项服务。顾客 i 需要的服务时间为 ti ,1≤i≤n1≤i≤n。共有 s 处可以提供此项服务。应如何安排 n 个顾客的服务次序才能使平均等待时间达到最小?平均等待时 间是 n 个顾客等待服务时间的总和除以 n。**

**对于给定的 n 个顾客需要的服务时间和 s 的值，编程计算最优服务次序。**

**数据输入：**

**第一行有 2 个正整数 n 和 s，表示有 n 个顾客且有 s 处可以提供顾客需要的服务。接下来的 1 行中，有 n 个正整数，表示 n 个顾客需要的服务时间。**

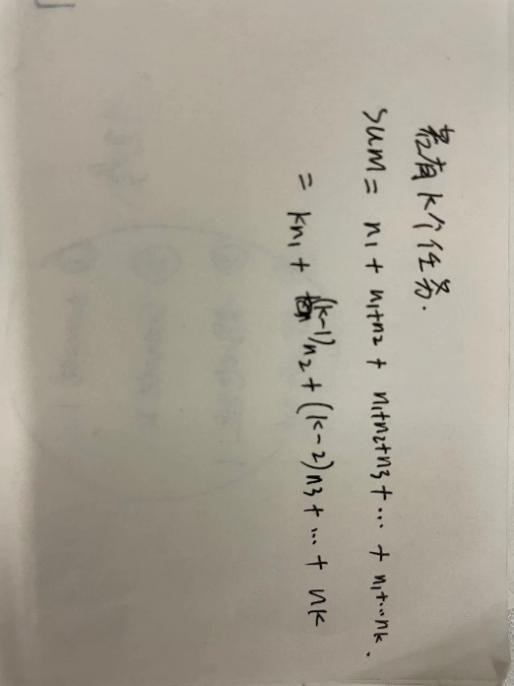
**（二）算法思路**（用文字简单说明）

**在思路来源：由于平均等待时间就是等待时间/n，相当于考虑总的等待时间最小。由于等待时间包括自身ti部分，以及等别人的时间bi，最终结果为sum（bi+ti）/n，sum(ti)为定值，为了让sum（bi）尽可能的小，则先让工作时间小的先执行。**

**于是需要先对执行时间进行排序，使用快速排序。**

**贪心选择：**

1. **选择等待时间最短的服务点加入新任务，假设有顾客n1，假设当前最少等待的任务点等待时间为b1，该顾客的最少等待时间为n1+b1，使得sum1 = sum（ni+bi）+(n1+b1)最小，假设没有选择b1而是选择另外的bi>b1,sum2 = sum(ni+bi)+(n1+bi)>sum1**

****

1. **每次选最短时间的任务加入，由上述公式得知，假设最小的元素不放在，最前端，则有sum’<sum**

**实际上根据上述公式即可以说明任意ni必须小于nj（i<j）,才能取得最小值。**

**最优子结构证明：**

**序列s是原问题的最优解，证明序列s-{n1}是子问题的最优解。若存在最优解s1-{n1}为子问题的最优解，则s1-{n1}并上{n1}优于s，与前提矛盾。**

**（三）算法实施步骤和流程**（伪代码/流程图等方式描述）

int bestplan(int s, int n, int\* list) {

int\* serve = new int[s];

int miniplace = 0;

int ttwaitt = 0;

for (int i = 0; i < s; i++) serve[i] = 0;

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

ttwaitt += (serve[miniplace]+list[i]);//一进去就默认好选哪个位置

cout << "时长为" << list[i] << "的顾客安排到第" << miniplace + 1 << "个服务点" << endl;

serve[miniplace] += list[i];

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

if (serve[j] < serve[miniplace]) miniplace = j;

}

}

return ttwaitt;

}

**（四）源代码**（通过了编译运行的正确程序）

#include<iostream>

using namespace std;

int partition(int\* list, int front, int end) {

int buffer = list[front];

while (front < end) {

while (front<end && list[end]>=buffer) end--;

list[front] = list[end];

while (front < end && list[front] <= buffer) front++;

list[end] = list[front];

}

list[front] = buffer;

return front;

}

void sort(int\* list, int front, int end) {

if (front >= end) return;

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

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

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

return;

}

int bestplan(int s, int n, int\* list) {

int\* serve = new int[s];

int miniplace = 0;

int ttwaitt = 0;

for (int i = 0; i < s; i++) serve[i] = 0;

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

ttwaitt += (serve[miniplace]+list[i]);//一进去就默认好选哪个位置

cout << "时长为" << list[i] << "的顾客安排到第" << miniplace + 1 << "个服务点" << endl;

serve[miniplace] += list[i];

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

if (serve[j] < serve[miniplace]) miniplace = j;

}

}

return ttwaitt;

}

int main(void) {

int s = 2;

int n = 10;

int list[10] = {56,12,1,99,1000,234,33,55,99,812};

sort(list,0,9);

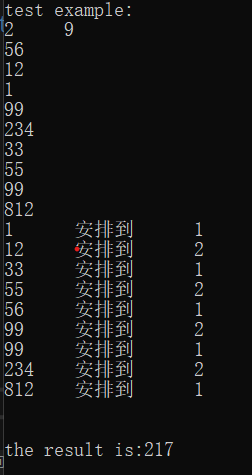
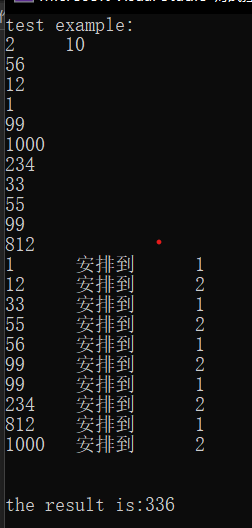
float result = bestplan(s, n, list)/n;

cout << "the result is:" << 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.简单写了一个快速排序，使得算法复杂度尽量降低。

2.时间复杂度为O(ns)n表示的是需要执行任务的人数，s表示的是任务点的个数，为了每次保证任务点最小，需要多次遍历。

3.关于贪心算法的证明，只需要证明首次选择，以及子问题的最优选择，能够推出原问题的最优选择。这类似于数学归纳法的意思，数学归纳法证明n=1，时间成立，假设n=n0时成立，能够推出n = n0+1时成立，这也相当于n取任何数时都能够成立。

**暨南大学本科实验报告专用纸(附页)**