

**实验（实习）报告**

实验(实习)名称：实验1

日期：2022.5.3

学院：应用技术学院

专业：计算机科学与技术

班级：1班

姓名：成凯

学号：201833050025

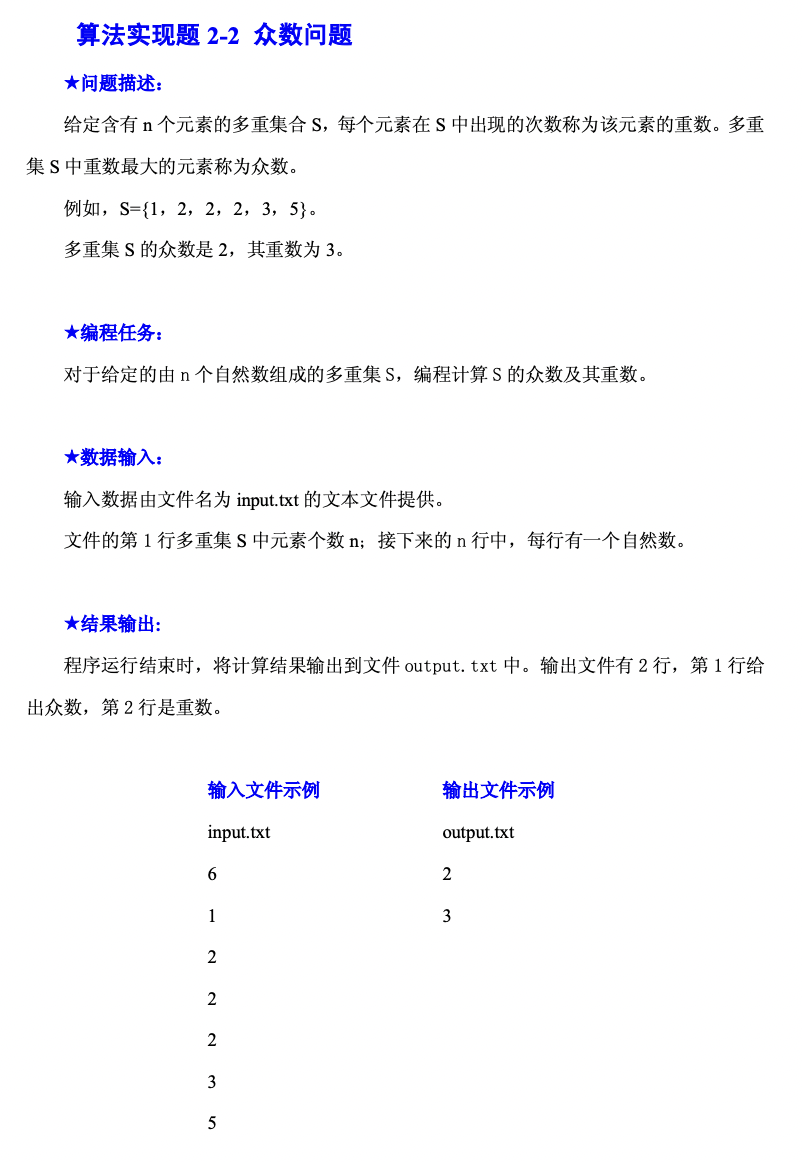
指导教师：庞亚伟

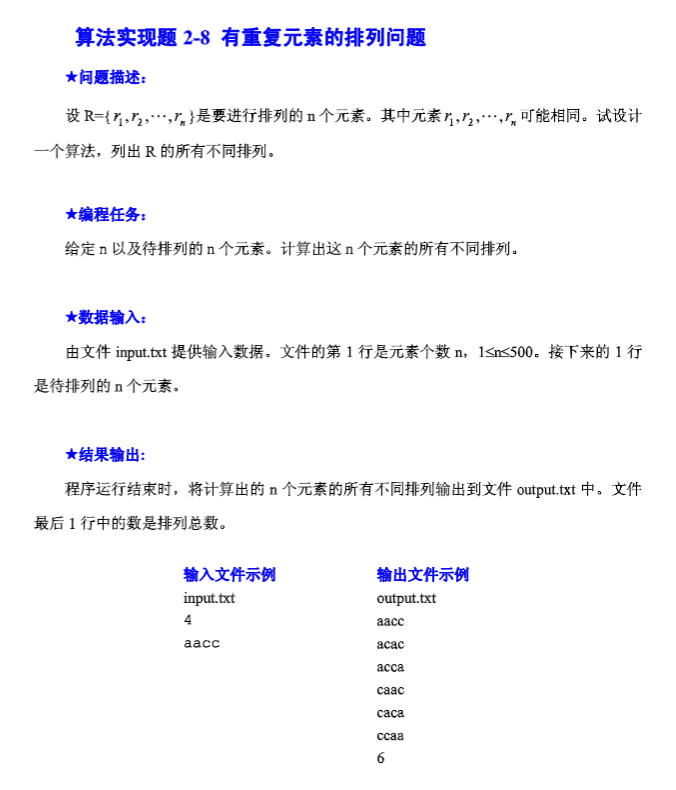
1. 实验目的

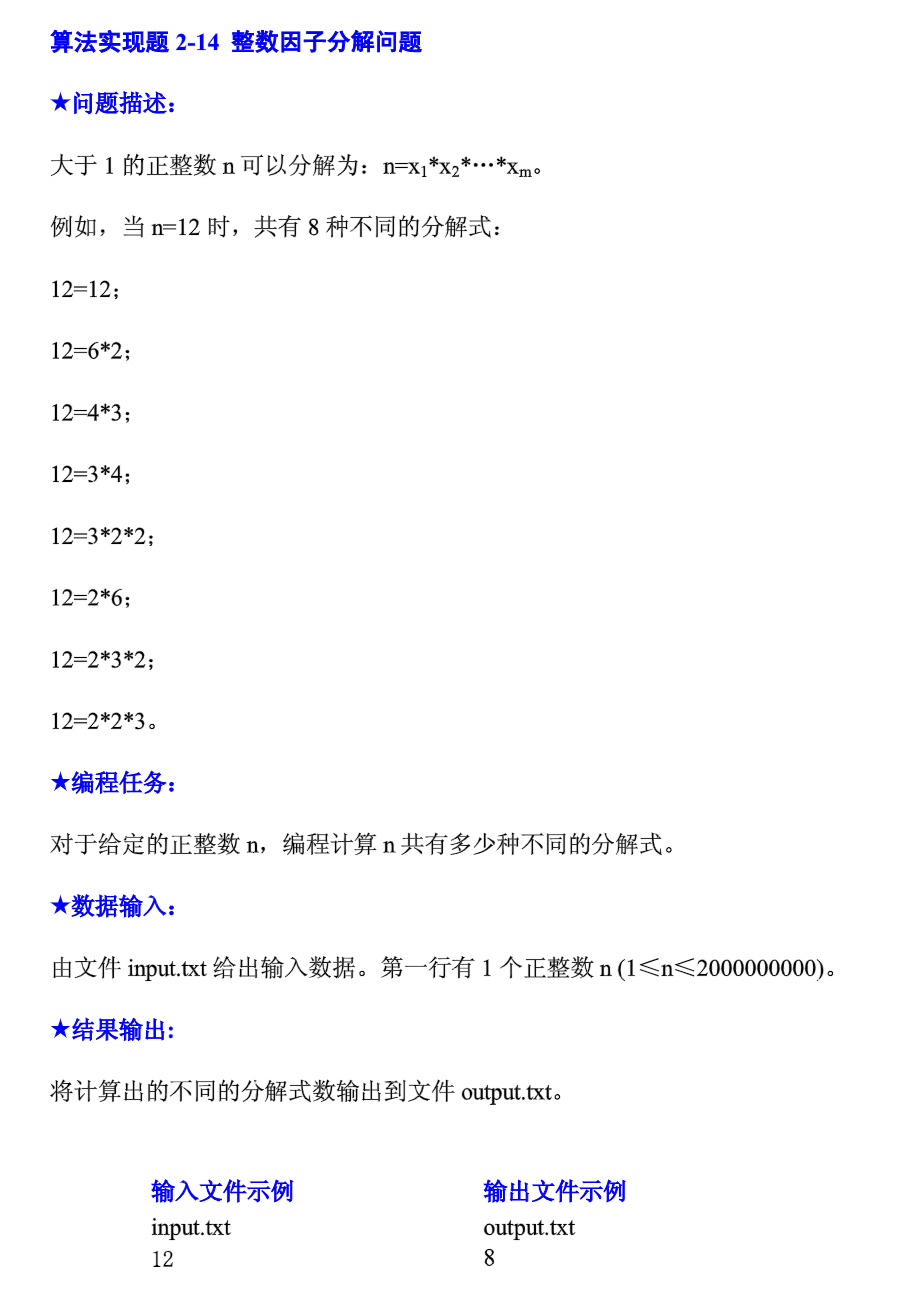
（1）理解掌握所学的第二章的总体算法思想；

（2）学会使用递归与分置解决部分算法问题；

1. 实验内容







1. 实验步骤和实验结果

算法一实现代码：

#include<stdio.h>

#include<string.h>

void sort(int buf[], int len) {

int i, j, k;

for (i = 0; i < len - 1; i++) {

k = i;

for (j = i + 1; j < len; j++) {

if (buf[k] > buf[j]) {

k = j;

}

}

if (k != i) {

buf[k] = buf[k] ^ buf[i];

buf[i] = buf[k] ^ buf[i];

buf[k] = buf[k] ^ buf[i];

}

}

}

int main() {

int i, j, k, max, m, buf[100], buf1[100], buf2[100];

bzero(buf, sizeof(buf));

bzero(buf1, sizeof(buf1));

bzero(buf2, sizeof(buf2));

scanf("%d", &m);

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

scanf("%d", &buf[i]);

}

sort(buf, m);

j = 0;

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

k = i;

buf2[j] = buf[i];

while (buf[k] == buf[i]) {

buf1[j]++;

i++;

}

i--;

j++;

}

max = 0;

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

if (buf1[max] < buf1[i]) {

max = i;

}

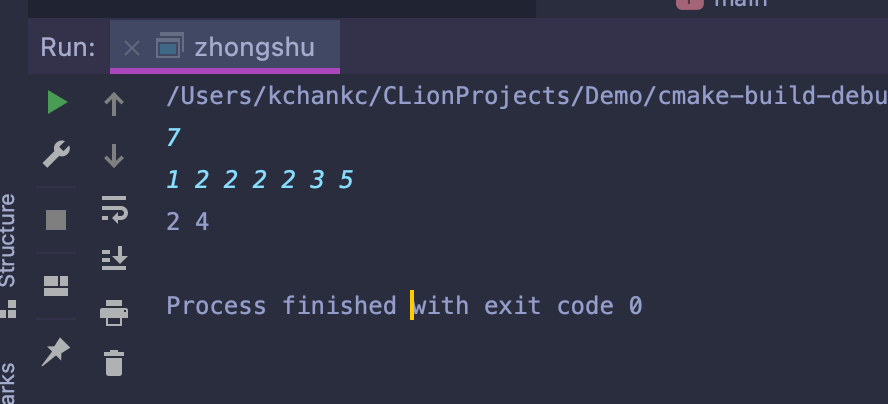
}

printf("%d %d\n", buf2[max], buf1[max]);

return 0;

}

算法一运行结果：



算法二实现代码：

#include<iostream>

using namespace std;

char xx[6666];

int cnt = 0;

int findsame(int k, int m) {

for (int i = k; i < m; i++) {

if (xx[i] == xx[m])

return 1;

}

return 0;

}

void perm(int k, int m) {

if (k == m) {

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

cout << xx[i];

}

cout << endl;

cnt++;

return;

}

for (int i = k; i <= m; i++) {

if (findsame(k, i))

continue;

swap(xx[k], xx[i]);

perm(k + 1, m);

swap(xx[k], xx[i]);

}

}

int main() {

int n;

cout << "请输入元素个数：";

cin >> n;

cout << "请输入" << n << "个不同的元素：";

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

cin >> xx[i];

perm(0, n - 1);

cout << "count=" << cnt;

return 0;

}

算法二运行结果：



算法三实现代码：

#include<stdio.h>

#include<stdlib.h>

#define maxsize 100

int count = 1;

void f(int\* a, int m) {

int i, j, k;

int\* p;

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

k = -1;

p = (int\*) malloc(sizeof(int) \* maxsize);

for (j = 2; j < a[i]; j++) {

if (a[i] % j == 0) {

k++;

p[k] = j;

}

}

if (k > -1) {

f(p, k);

count = count + k + 1;

}

free(p);

}

return;

}

int main() {

int\* a;

a = (int\*) malloc(sizeof(int) \* maxsize);

scanf("%d", &a[0]);

f(a, 0);

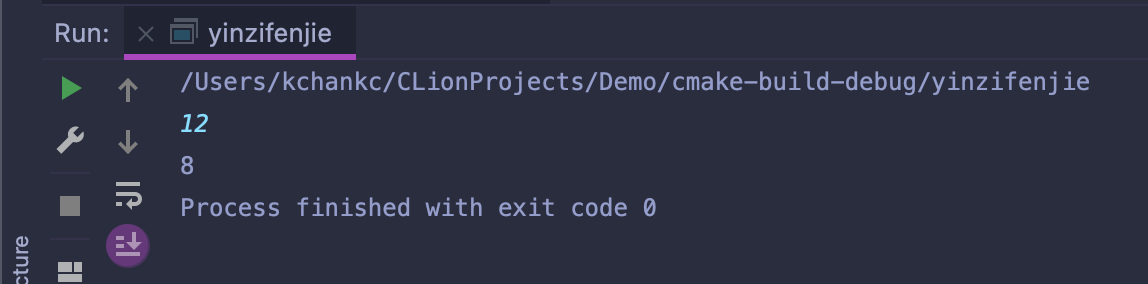
free(a);

printf("%d", count);

return 0;

}

算法三运行结果：



1. 分析与讨论

通过本次实验掌握了如何使用递归算法解决一些小问题。了解了分治算法的应用，总的来说通过本次实验受益匪浅。



**实验（实习）报告**

实验(实习)名称：实验二

日期：2022.5.10

学院：应用技术学院

专业：计算机科学与技术

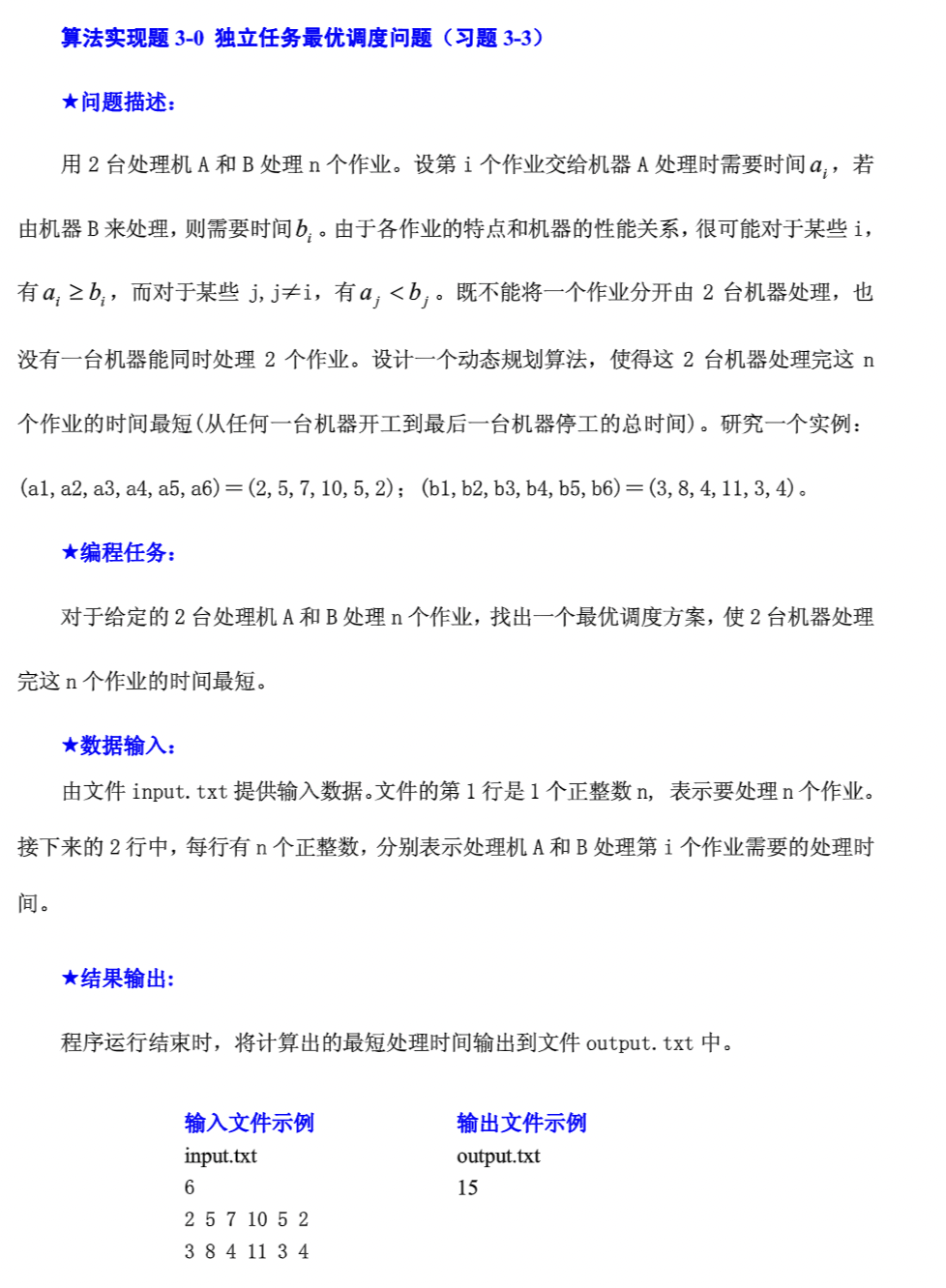
班级：1班

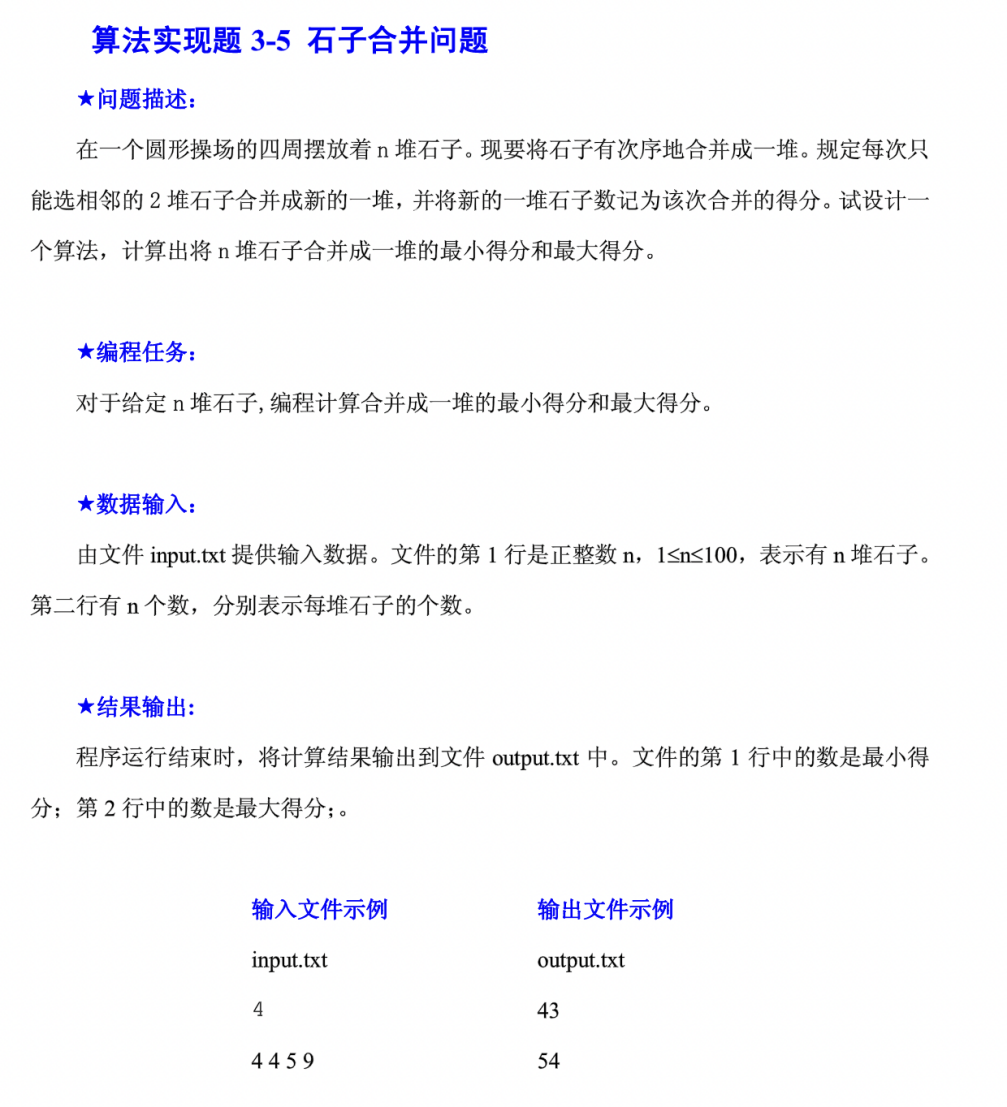
姓名：成凯

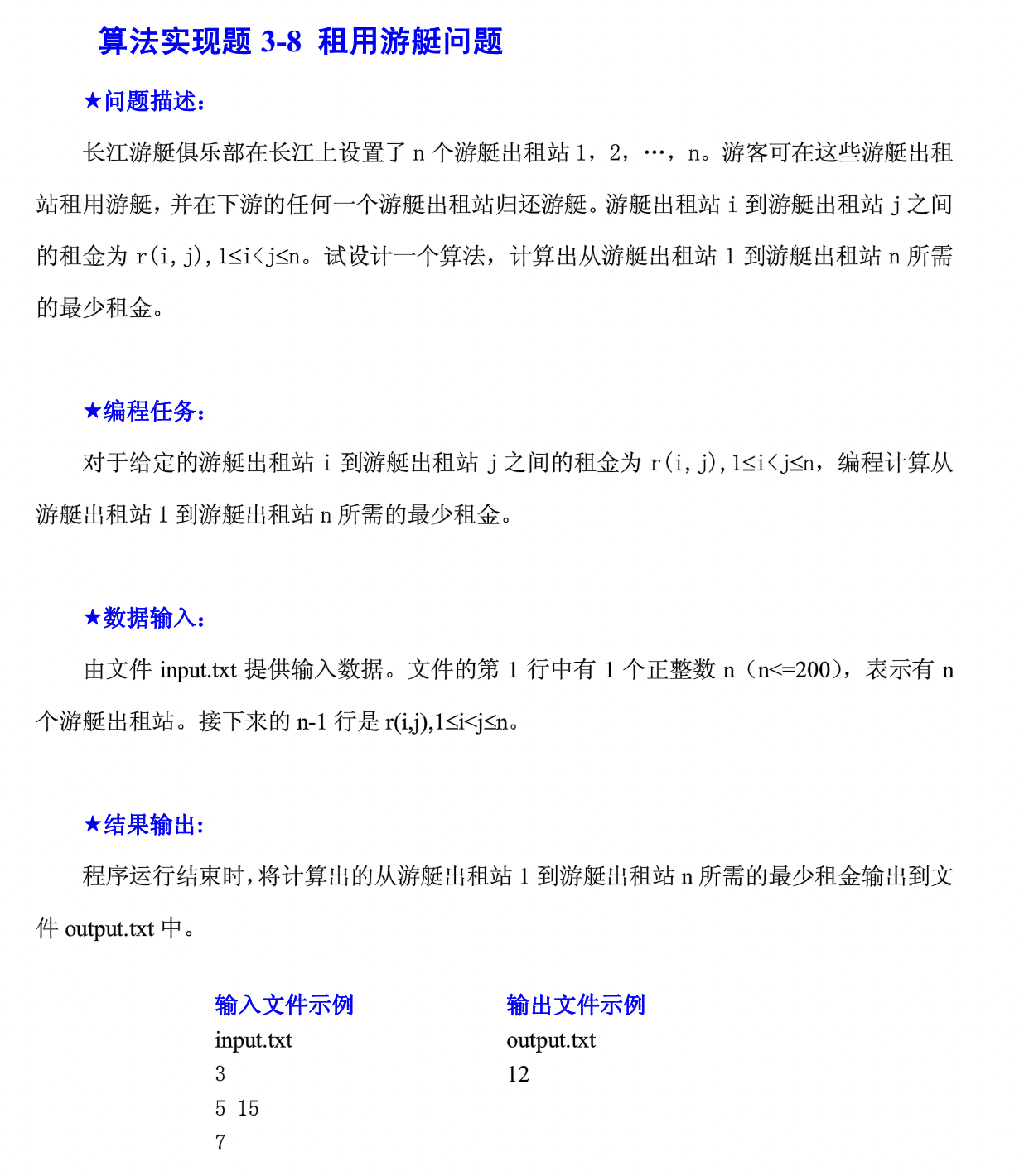
学号：201833050025

指导教师：庞亚伟

1. 实验目的
2. 实验内容







1. 实验步骤和实验结果

算法一实现代码：

#include <stdio.h>

int a[100];

int b[100];

int dp[100][100];

int n, sum;

int main() {

scanf("%d", &n);

int i, j;

sum = 0;

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

scanf("%d", &a[i]);

sum += a[i];

}

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

scanf("%d", &b[i]);

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

for (j = 0; j <= sum; j++)

if (j < a[i - 1])

dp[i][j] = dp[i - 1][j] + b[i - 1];

else if (dp[i - 1][j - a[i - 1]] > dp[i - 1][j] + b[i - 1])

dp[i][j] = dp[i - 1][j] + b[i - 1];

else

dp[i][j] = dp[i - 1][j - a[i - 1]];

}

int temp, ans = 99999999;

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

temp = dp[n][i] > i ? dp[n][i] : i; //机器a与b所花时间取个大值

if (temp < ans)

ans = temp; //求出最小

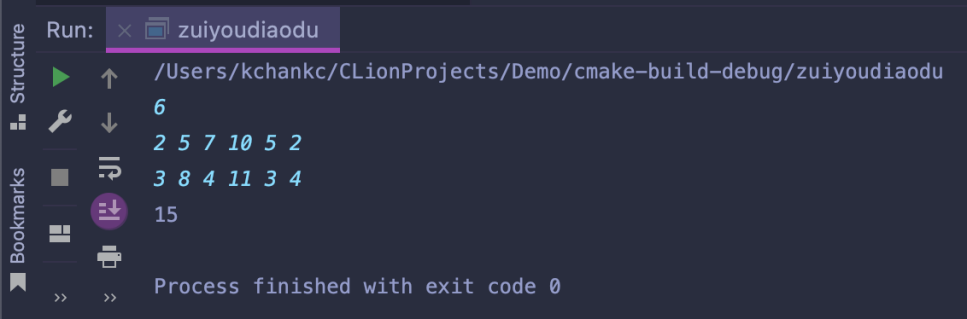
}

printf("%d\n", ans);

return 0;

}

算法一运行结果：



算法二实现代码：

#include <iostream>

#include <algorithm>

#define INF 0x3f3f3f3f

using namespace std;

int Arr[300], Sum[300];

int Min[300][300], Max[300][300];

int main() {

int n;

cin >> n;

// 初始化数组

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

cin >> Arr[i];

Arr[i + n] = Arr[i];

}

// 计算最大和

for (int i = 1; i <= 2 \* n; i++) {

Sum[i] = Sum[i - 1] + Arr[i];

}

// 开始递归循环

for (int i = 2 \* n - 1; i >= 1; i--) {

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

Min[i][j] = INF;

for (int k = i; k < j; k++) {

Min[i][j] = min(Min[i][j], Min[i][k] + Min[k + 1][j] + Sum[j] - Sum[i - 1]);

Max[i][j] = max(Max[i][j], Max[i][k] + Max[k + 1][j] + Sum[j] - Sum[i - 1]);

}

}

}

// 遍历找到最大与最小值

int MaxValue = 0, MinValue = INF;

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

MaxValue = max(MaxValue, Max[i][i + n - 1]);

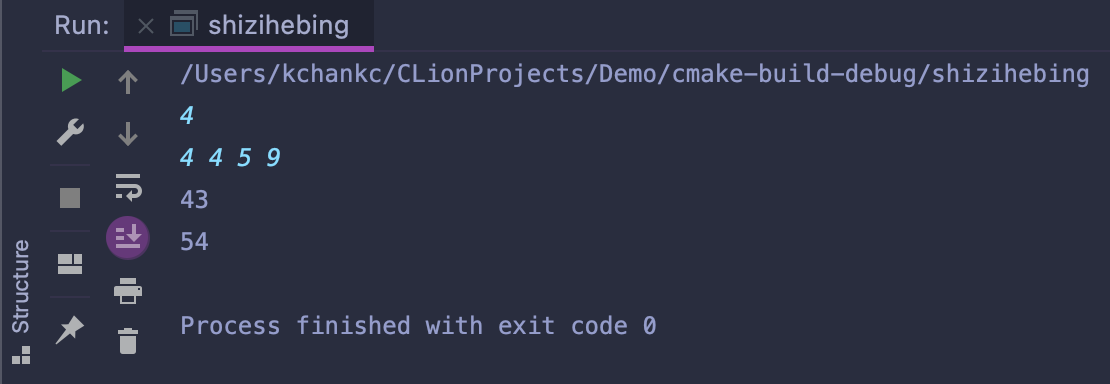
MinValue = min(MinValue, Min[i][i + n - 1]);

}

cout << MinValue << endl << MaxValue << endl;

}

算法二运行结果：



算法三实现代码：

#include<iostream>

#include<cstdio>

using namespace std;

void read(int &n) {

char c = '+';

int x = 0;

int flag = 0;

while (c < '0' || c > '9') {

c = getchar();

if (c == '-')

flag = 1;

}

while (c >= '0' && c <= '9')

x = x \* 10 + (c - 48), c = getchar();

flag == 1 ? n = -x : n = x;

}

const int MAXN = 1001;

int maxt, n;

int dp[MAXN][MAXN];

int main() {

read(n);

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

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

dp[i][j] = 438438;

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

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

read(dp[i][j]);

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

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

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

if (dp[i][k] != 438438 && dp[k][j] != 438438) {

dp[i][j] = min(dp[i][j], dp[i][k] + dp[k][j]);

}

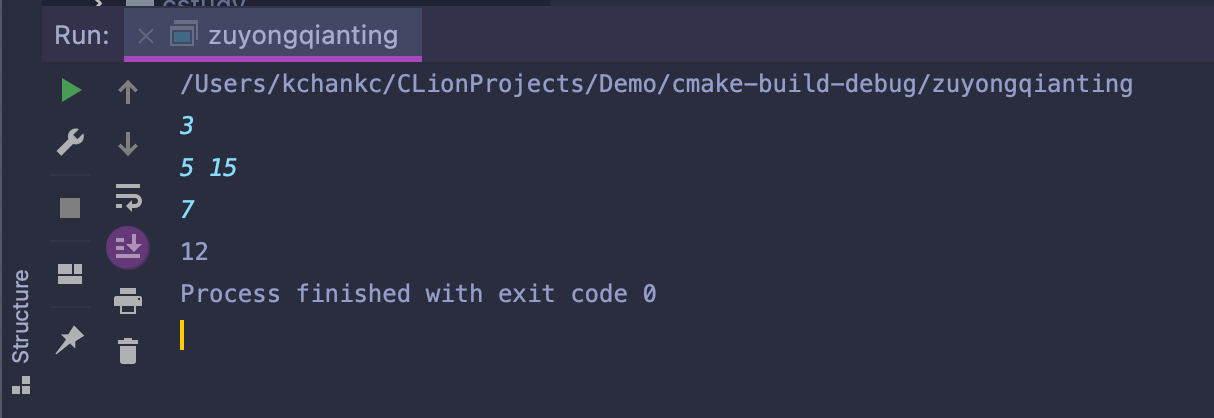
}

cout << dp[1][n];

return 0;

}

算法三运行结果：



1. 分析与讨论

通过本次实验学习了动态规划以及贪心算法，例如第二题环形石子合并问题就可以使用动态规划来解决。在动态规划问题中，经常可以解得如下的转移方程：dp[i][j]=min{dp[i][k]+dp[k+1][j]+cost[i][j]}。总而言之透过本次实验让我受益匪浅。



**实验（实习）报告**

实验(实习)名称：实验三

日期：2022.5.17

学院：应用技术学院

专业：计算机科学与技术

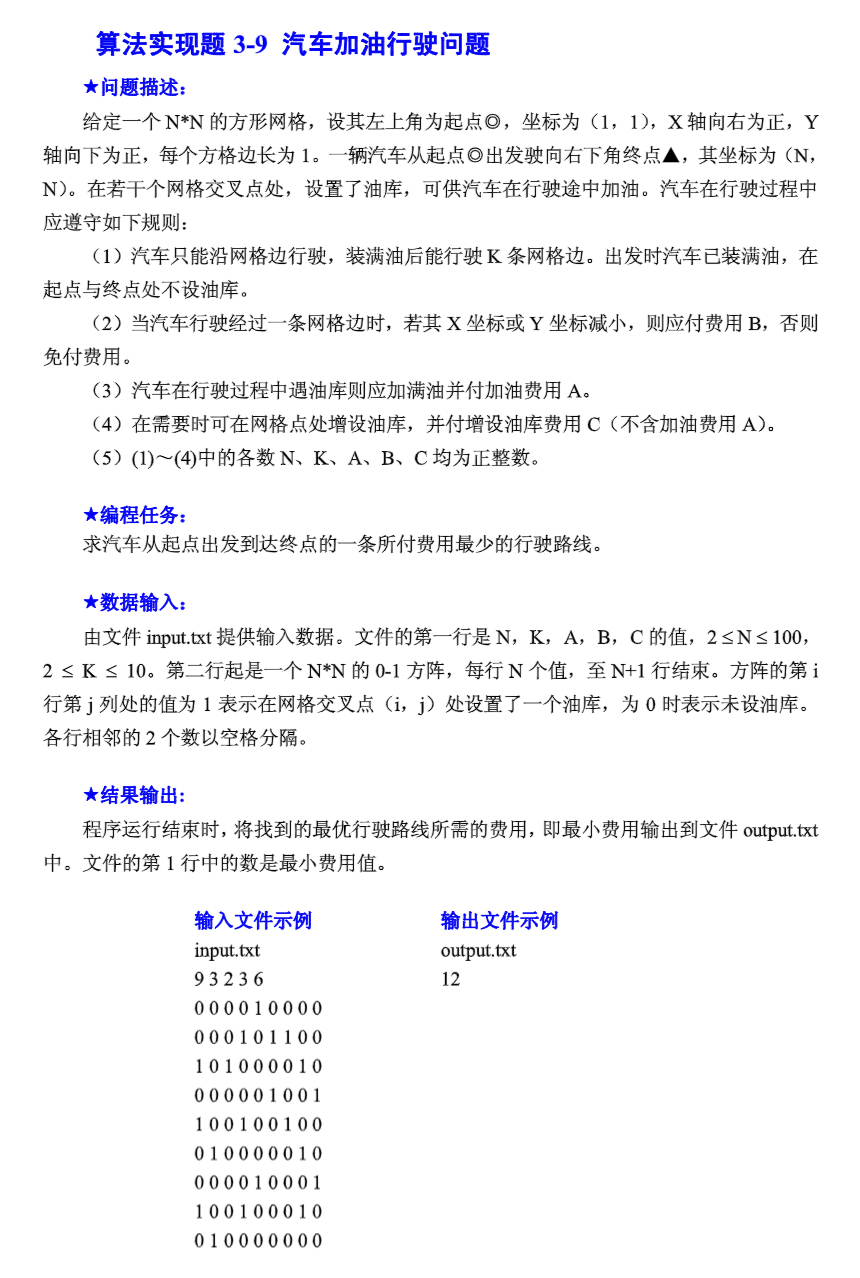
班级：1班

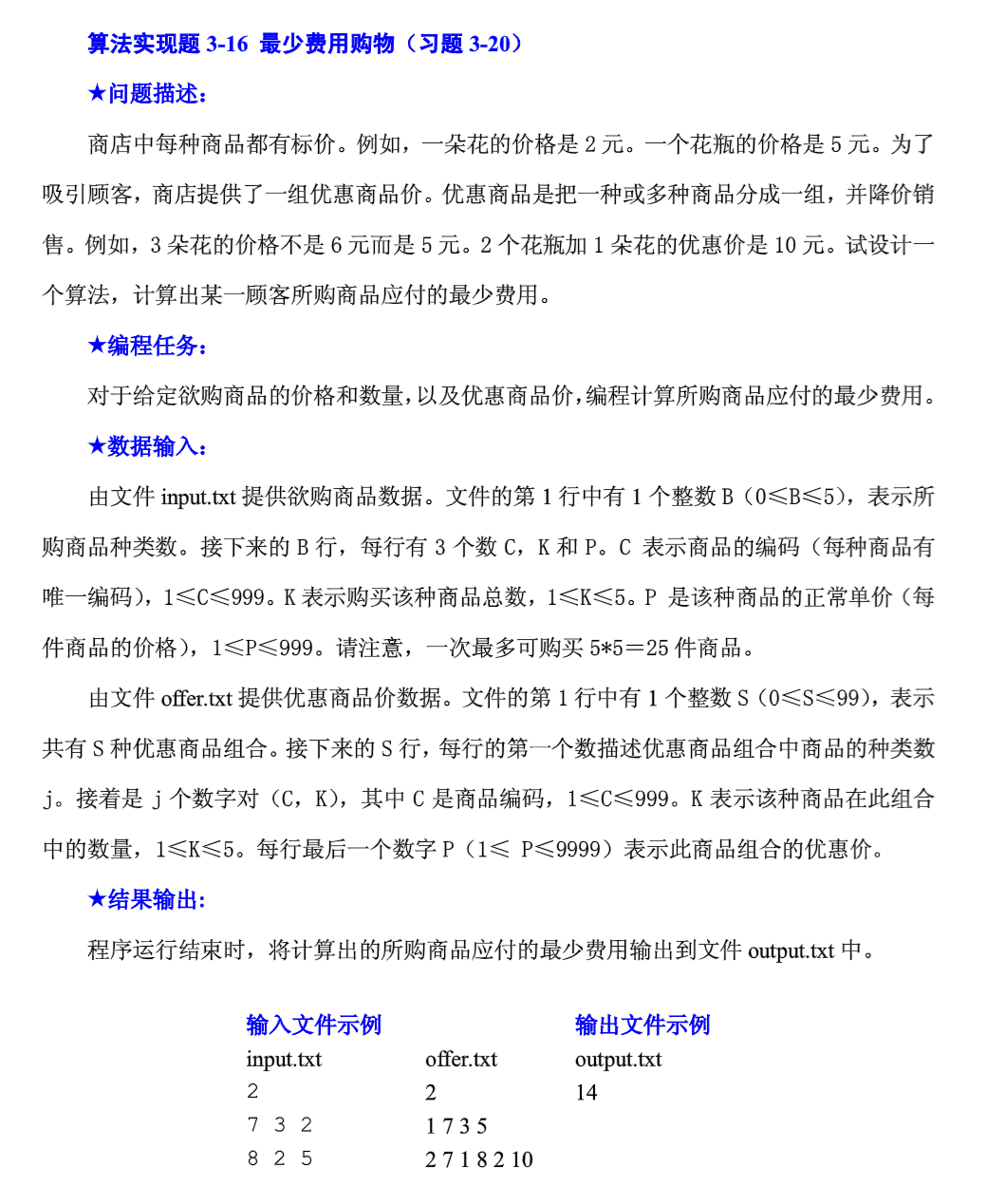
姓名：成凯

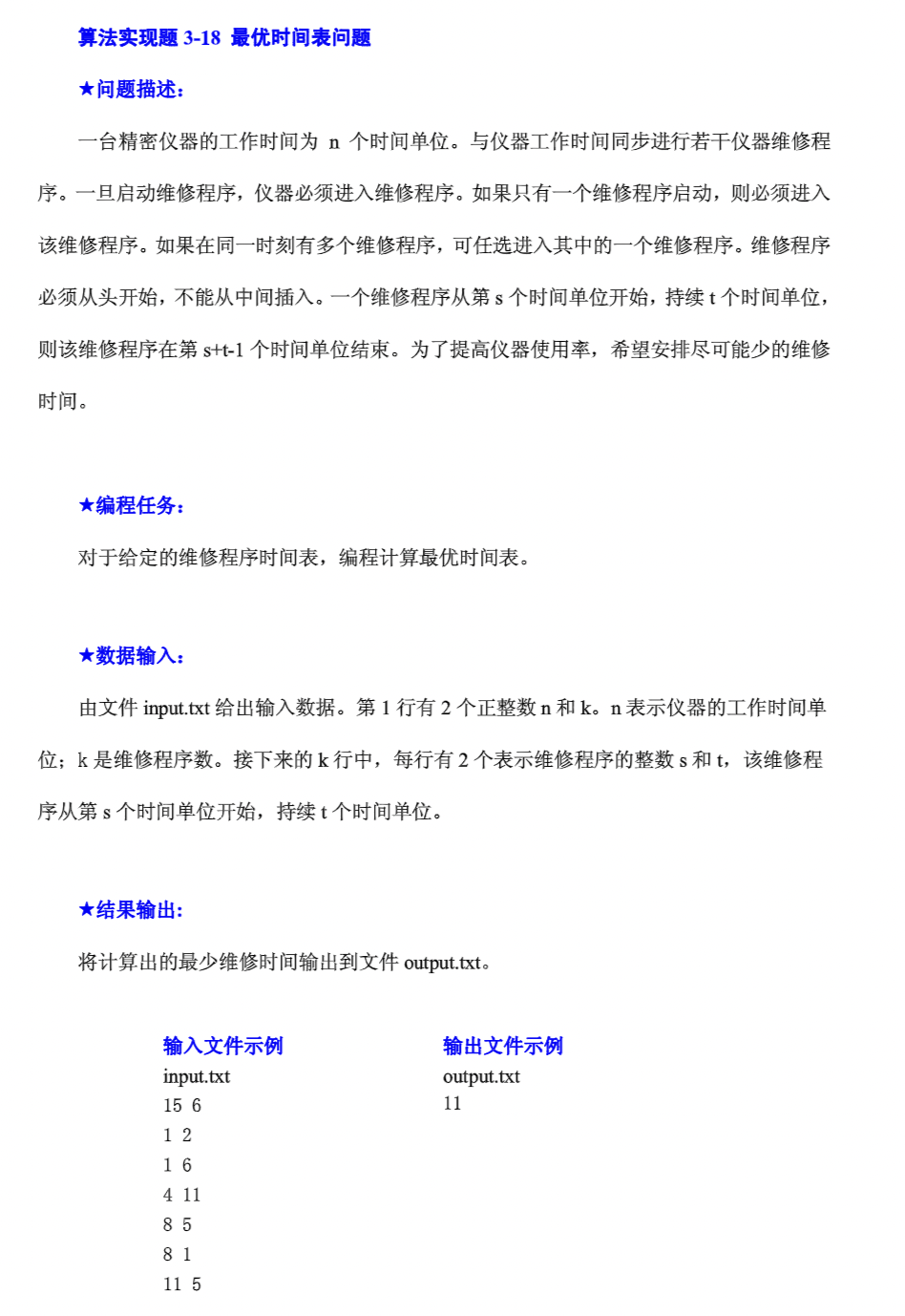
学号：201833050025

指导教师：庞亚伟

1. 实验目的
2. 实验内容







1. 实验步骤和实验结果

算法一实现代码：

#include<cstdio>

#include<iostream>

#include<algorithm>

#include<cstring>

#include<queue>

#define maxn 1000010

#define maxm 3000010

#define inf 1000000000

using namespace std;

int head[maxn], dis[maxn], n, k, a, b, c, cnt;

bool inq[maxn];

int map[110][110];

struct edge {

int next;

int to;

int val;

} e[maxm];

void insert(int u, int v, int w) {

e[++cnt].next = head[u];

head[u] = cnt;

e[cnt].to = v;

e[cnt].val = w;

}

queue<int> q;

void spfa(int s) {

memset(dis, 0x3f, sizeof(dis));

memset(inq, false, sizeof(inq));

dis[s] = 0;

inq[s] = true;

q.push(s);

while (!q.empty()) {

int now = q.front();

q.pop();

inq[now] = false;

for (int i = head[now]; i; i = e[i].next) {

int v = e[i].to;

if (dis[v] > dis[now] + e[i].val) {

dis[v] = dis[now] + e[i].val;

if (!inq[v]) {

q.push(v);

inq[v] = true;

}

}

}

}

}

int getid(int x, int y, int level) {

return ((x - 1) \* n + y) + n \* n \* (k - level);

}

int main() {

cin >> n >> k >> a >> b >> c;

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

for (int j = 1; j <= n; j++) scanf("%d", &map[i][j]);

}

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

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

if (map[i][j] == 1) {

for (int l = 0; l <= k - 1; l++) insert(getid(i, j, l), getid(i, j, k), a);

if (i < n) insert(getid(i, j, k), getid(i + 1, j, k - 1), 0);

if (j < n) insert(getid(i, j, k), getid(i, j + 1, k - 1), 0);

if (i > 1) insert(getid(i, j, k), getid(i - 1, j, k - 1), b);

if (j > 1) insert(getid(i, j, k), getid(i, j - 1, k - 1), b);

} else {

for (int l = 0; l <= k - 1; l++) insert(getid(i, j, l), getid(i, j, k), a + c);

for (int l = 1; l <= k; l++) {

if (i < n) insert(getid(i, j, l), getid(i + 1, j, l - 1), 0);

if (j < n) insert(getid(i, j, l), getid(i, j + 1, l - 1), 0);

if (i > 1) insert(getid(i, j, l), getid(i - 1, j, l - 1), b);

if (j > 1) insert(getid(i, j, l), getid(i, j - 1, l - 1), b);

}

}

}

}

spfa(getid(1, 1, k));

int ans = inf;

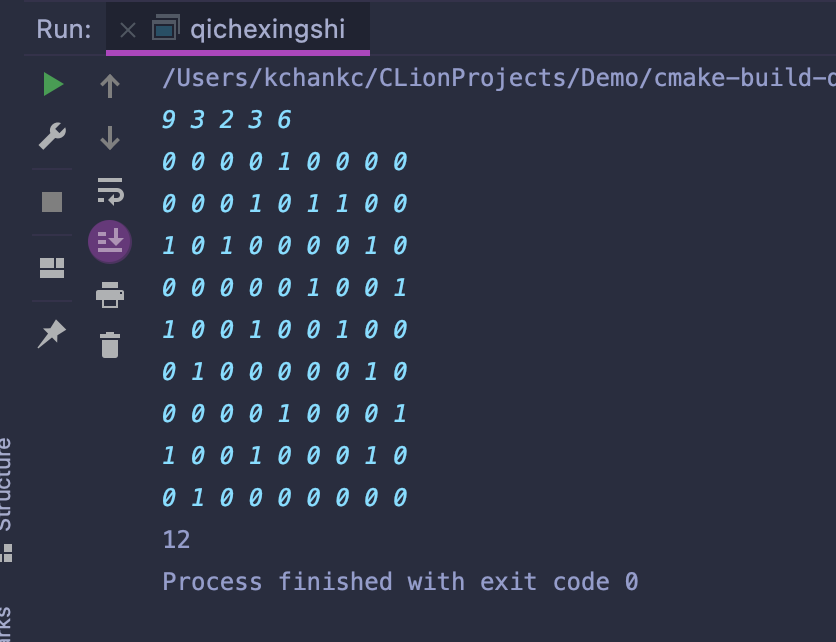
for (int i = 0; i <= k; i++) ans = min(ans, dis[getid(n, n, i)]);

cout << ans;

return 0;

}

算法一运行结果：



算法二实现代码：

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

typedef pair<int, int> P;

const int MAX = 6;

const int INF = 1e9;

int map[1000];

int n, m;

int ids[MAX];

int price[MAX];

int nums[MAX];

vector<P> pairs[100];

int pP[100];

int pcnt = 0;

int dp[MAX][MAX][MAX][MAX][MAX];

int times = 0;

int dfs(int\* x) {

times++;

int r = dp[x[0]][x[1]][x[2]][x[3]][x[4]];

if (r > 0) {

return (r);

}

if (x[0] == 0 && x[1] == 0 && x[2] == 0 && x[3] == 0 && x[4] == 0) {

return (0);

}

int minf = INF;

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

vector<P> &vec = pairs[i];

int f = 1;

int\* y = new int[5];

for (int t = 0; t < 5; t++)

y[t] = 0;

for (auto p: vec) {

int id = map[p.first];

int num = p.second;

if (x[id] < num) {

f = 0;

break;

}

y[id] = -num;

}

if (!f)

continue;

for (int k = 0; k < 5; k++)

y[k] += x[k];

minf = min(minf, pP[i] + dfs(y));

}

int s = 0;

for (int i = 0; i < 5; i++) {

s += x[i] \* price[i];

}

minf = min(minf, s);

return (dp[x[0]][x[1]][x[2]][x[3]][x[4]] = minf);

}

int main() {

cin >> n;

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

int C, K, PP;

cin >> C >> K >> PP;

ids[i] = C;

nums[i] = K;

price[i] = PP;

if (!map[C]) {

map[C] = i;

}

}

cin >> m;

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

int k;

cin >> k;

vector<P> v;

int f = 1;

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

int a, b;

cin >> a >> b;

v.push\_back(make\_pair(a, b));

}

int PP;

cin >> PP;

if (f) {

pairs[pcnt] = v;

pP[pcnt++] = PP;

}

}

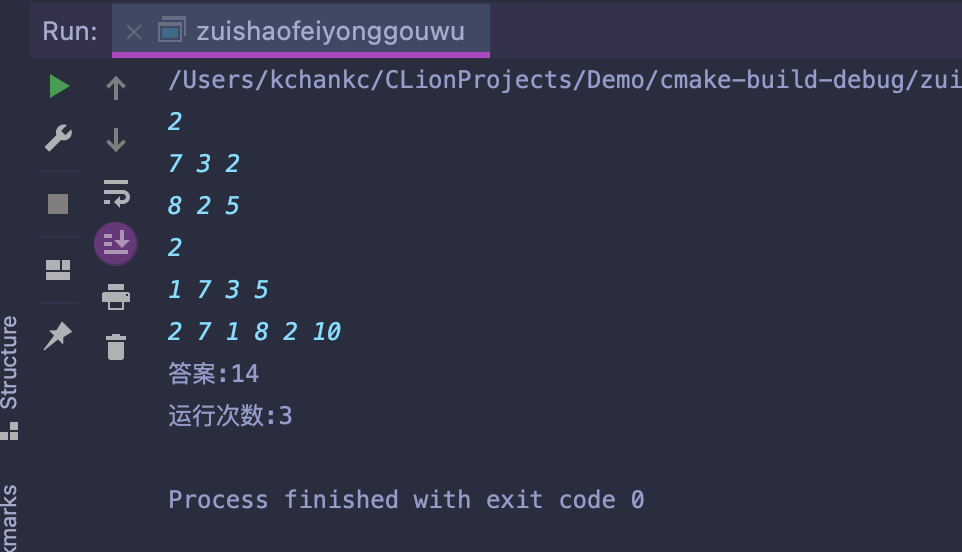
cout << "答案:" << dfs(nums) << endl;

cout << "运行次数:" << times << endl;

return (0);

}

算法二运行结果：



算法三实现代码：

#include <iostream>

#include <vector>

#include <cstring>

using namespace std;

const int MAXN = 2e5 + 100;

vector<int> vs[MAXN];

int dp[MAXN];

int main() {

int n, k, x, y;

cin >> n >> k;

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

cin >> x >> y;

vs[x].emplace\_back(y);

}

memset(dp, 0x3f, sizeof dp);

dp[n] = 0;

for (int i = n - 1; i > 0; i--) {

int len = vs[i].size();

if (len == 0) dp[i] = dp[i + 1];

else {

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

int t = dp[i + vs[i][j]] + vs[i][j];

if (t <= n) {

dp[i] = min(dp[i], dp[i + vs[i][j]] + vs[i][j]);

} else {

dp[i] = min(dp[i], n);//不能超过n

}

}

}

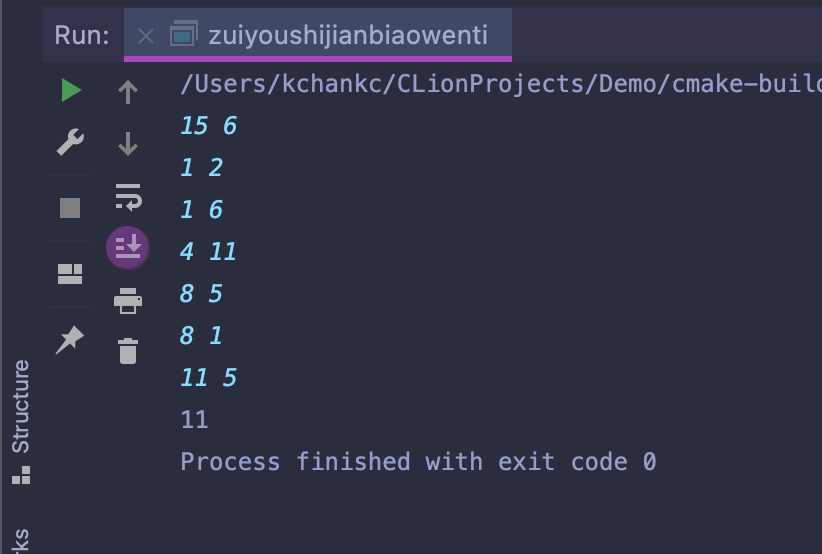
}

cout << dp[1];

return 0;

}

算法三运行结果：



1. 分析与讨论

通过本次实验加深了对动态规划算法的理解，透过本次实验让我受益匪浅。



**实验（实习）报告**

实验(实习)名称：实验四

日期：2022.5.26

学院：应用技术学院

专业：计算机科学与技术

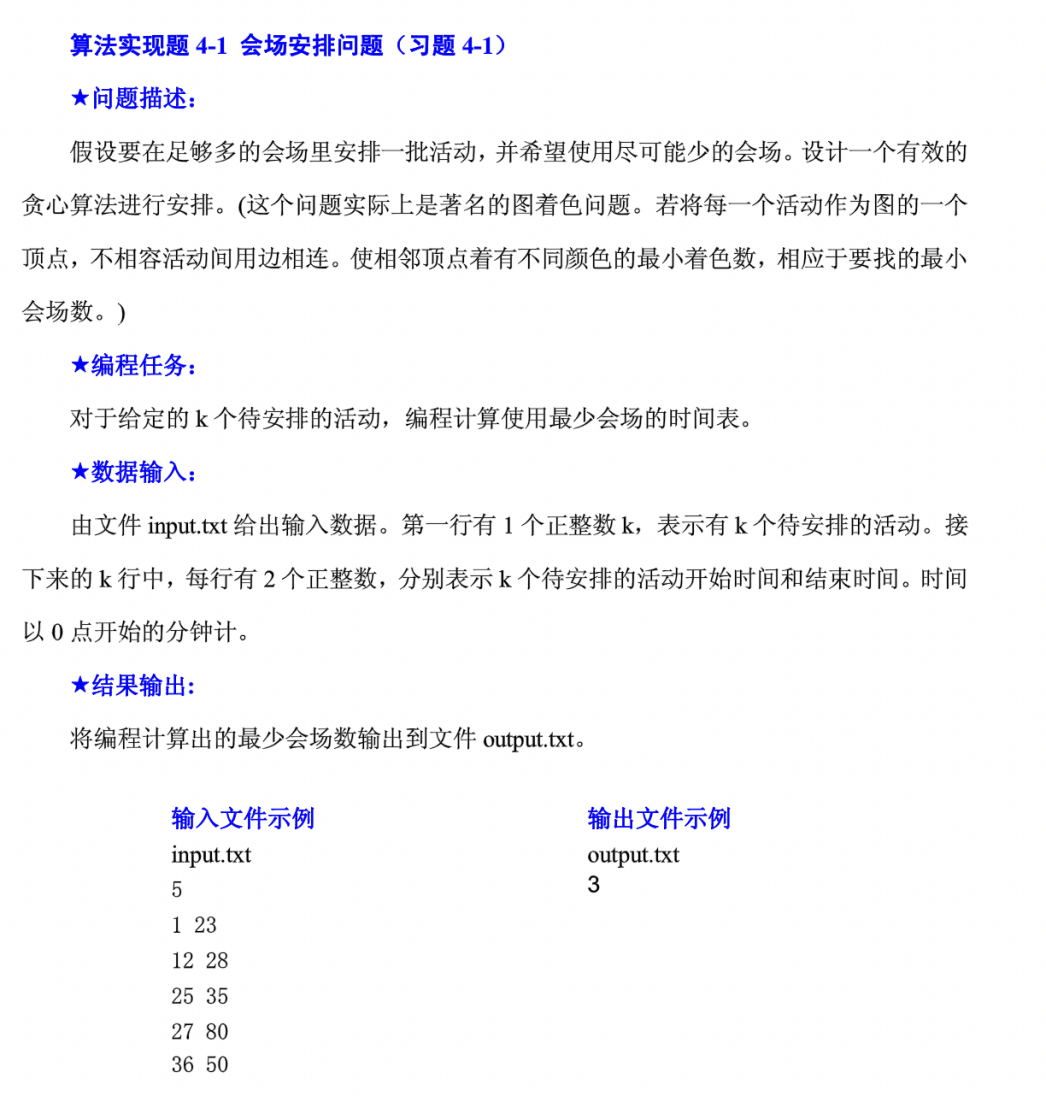
班级：1班

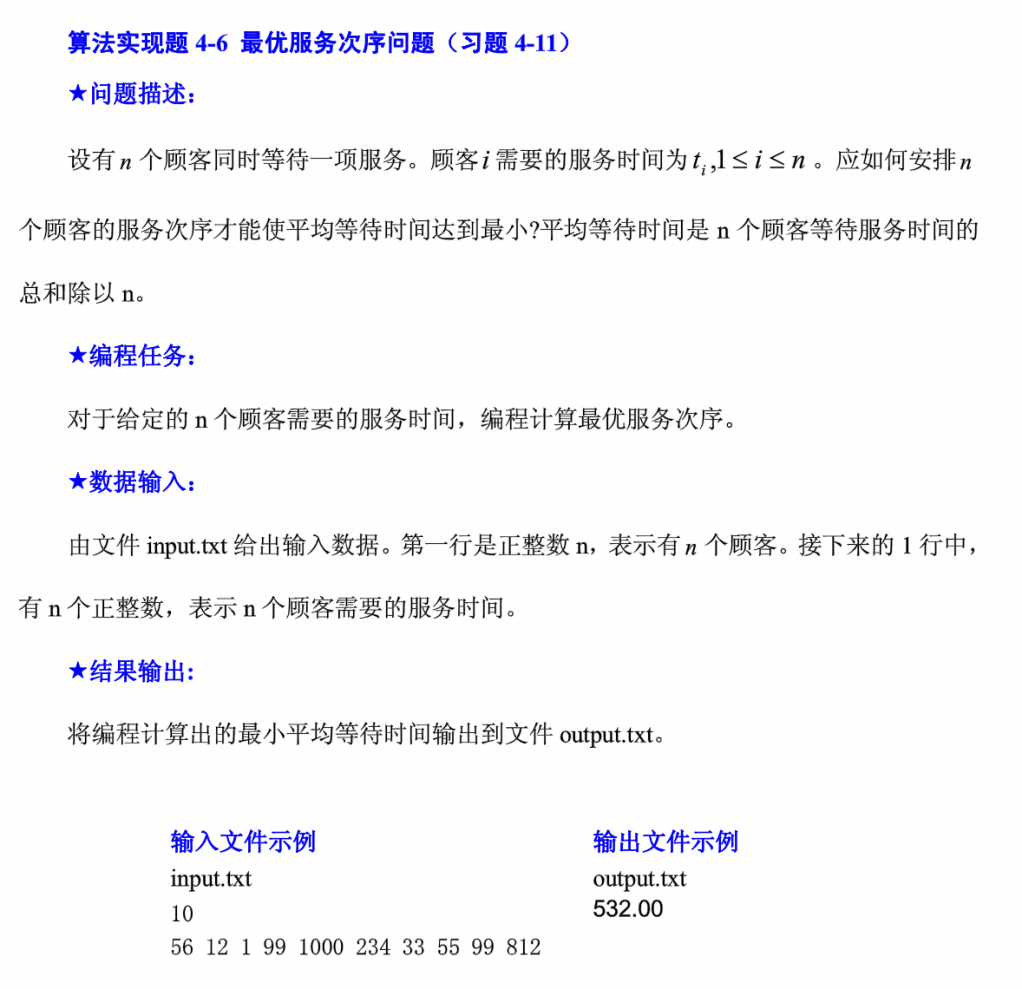
姓名：成凯

学号：201833050025

指导教师：庞亚伟

1. 实验目的
2. 实验内容







1. 实验步骤和实验结果

算法一实现代码：

#include <iostream>

#include <algorithm>

using namespace std;

int main() {

ios::sync\_with\_stdio(false);

cin.tie(NULL);

cout.tie(NULL);

int n, i, Begins[10002], Ends[10002]; // Begins开始时间, Ends结束时间

cin >> n;

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

cin >> Begins[i] >> Ends[i];

sort(Begins, Begins + n); // 升序排序

sort(Ends, Ends + n); // 升序排序

int j = 0, ans = 0;

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

if (Begins[i] < Ends[j]) // 如果开始时间小于结束时间, 会场数加1

ans++;

else // 开始时间大于等于结束时间, 会场不用增加, 换下一家活动

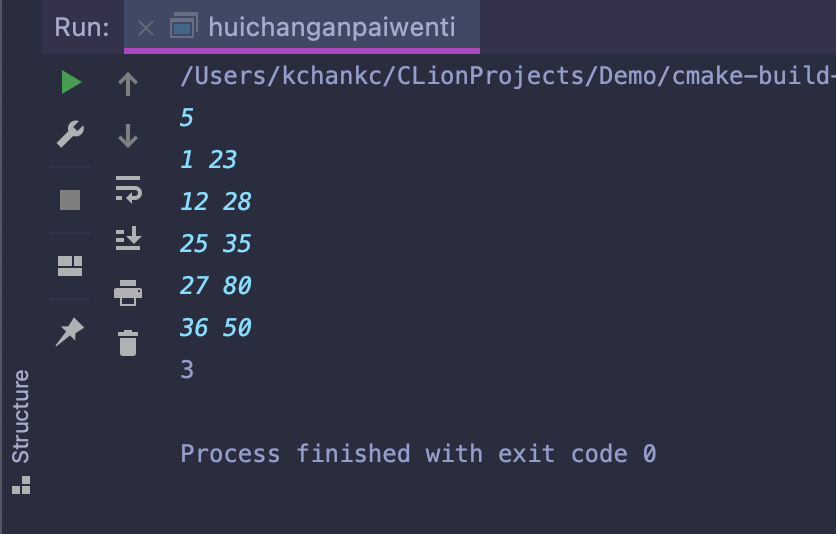
j++;

cout << ans << endl;

return 0;

}

算法一运行结果：



算法二实现代码：

#include<stdio.h>

#include<algorithm>

#define n 10

#define s 2

using namespace std;

double fun1(int\* list) { //最优服务次序问题

int i, sum = 0;

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

sum += list[i] \* (n - i);

return sum / (n \* 1.0);

}

double fun2(int\* list) { //多次最优服务次序问题

int i, sum = 0;

int\* temp = new int[s];

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

temp[i] = 0;

}

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

temp[i % s] += list[i];

sum += temp[i % s];

}

return sum / (n \* 1.0);

}

int main() {

int i;

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

sort(list, list + n);

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

printf("%d\t", list[i]);

}

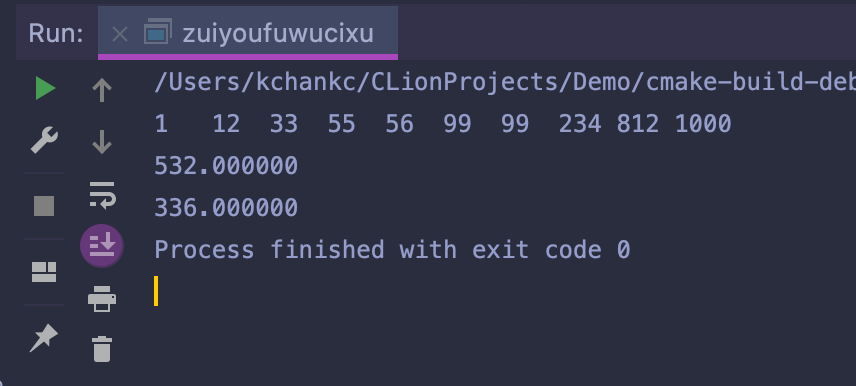
printf("\n%1f", fun1(list));

printf("\n%1f", fun2(list));

return 0;

}

算法二运行结果：



算法三实现代码：

#include<cstdio>

#include<iostream>

#include<algorithm>

#include<cstring>

#include<queue>

#define maxn 1000010

#define maxm 3000010

#define inf 1000000000

using namespace std;

int head[maxn], dis[maxn], n, k, a, b, c, cnt;

bool inq[maxn];

int map[110][110];

struct edge {

int next;

int to;

int val;

} e[maxm];

void insert(int u, int v, int w) {

e[++cnt].next = head[u];

head[u] = cnt;

e[cnt].to = v;

e[cnt].val = w;

}

queue<int> q;

void spfa(int s) {

memset(dis, 0x3f, sizeof(dis));

memset(inq, false, sizeof(inq));

dis[s] = 0;

inq[s] = true;

q.push(s);

while (!q.empty()) {

int now = q.front();

q.pop();

inq[now] = false;

for (int i = head[now]; i; i = e[i].next) {

int v = e[i].to;

if (dis[v] > dis[now] + e[i].val) {

dis[v] = dis[now] + e[i].val;

if (!inq[v]) {

q.push(v);

inq[v] = true;

}

}

}

}

}

int getid(int x, int y, int level) {

return ((x - 1) \* n + y) + n \* n \* (k - level);

}

int main() {

cin >> n >> k >> a >> b >> c;

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

for (int j = 1; j <= n; j++) scanf("%d", &map[i][j]);

}

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

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

if (map[i][j] == 1) {

for (int l = 0; l <= k - 1; l++) insert(getid(i, j, l), getid(i, j, k), a);

if (i < n) insert(getid(i, j, k), getid(i + 1, j, k - 1), 0);

if (j < n) insert(getid(i, j, k), getid(i, j + 1, k - 1), 0);

if (i > 1) insert(getid(i, j, k), getid(i - 1, j, k - 1), b);

if (j > 1) insert(getid(i, j, k), getid(i, j - 1, k - 1), b);

} else {

for (int l = 0; l <= k - 1; l++) insert(getid(i, j, l), getid(i, j, k), a + c);

for (int l = 1; l <= k; l++) {

if (i < n) insert(getid(i, j, l), getid(i + 1, j, l - 1), 0);

if (j < n) insert(getid(i, j, l), getid(i, j + 1, l - 1), 0);

if (i > 1) insert(getid(i, j, l), getid(i - 1, j, l - 1), b);

if (j > 1) insert(getid(i, j, l), getid(i, j - 1, l - 1), b);

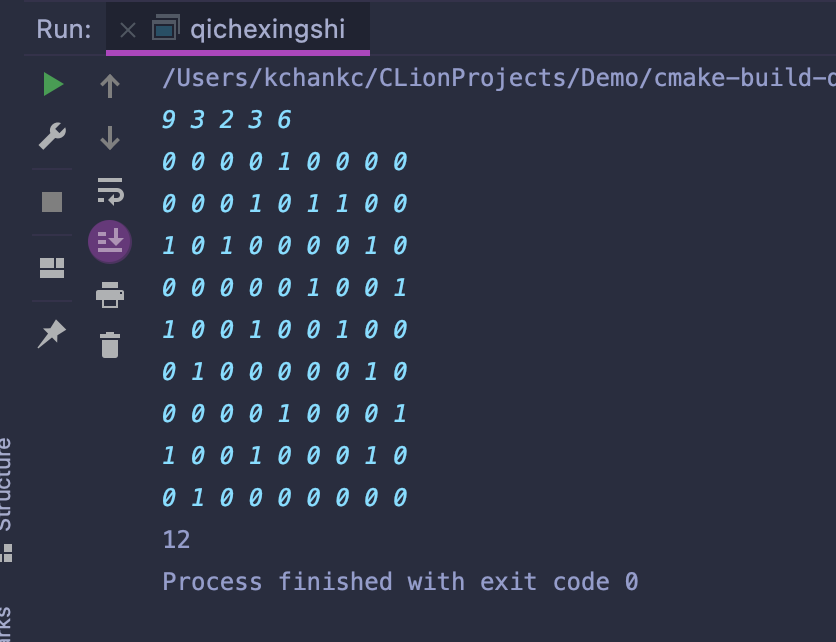
}

}

}

}

算法三运行结果：



1. 分析与讨论

通过本次实验再次加深了对动态规划算法的理解，总而言之透过本次实验让我受益匪浅。