

**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

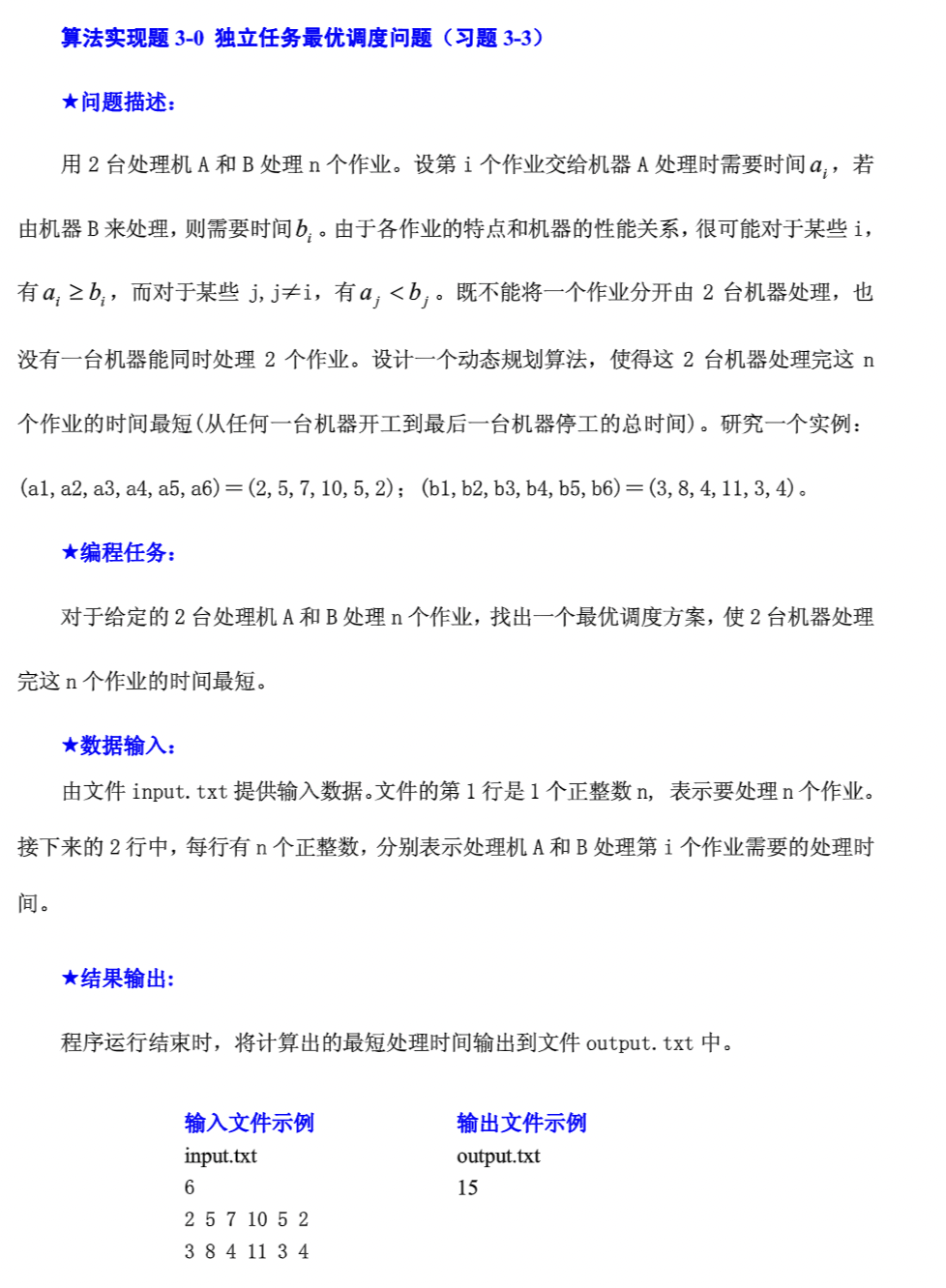
班级：1班

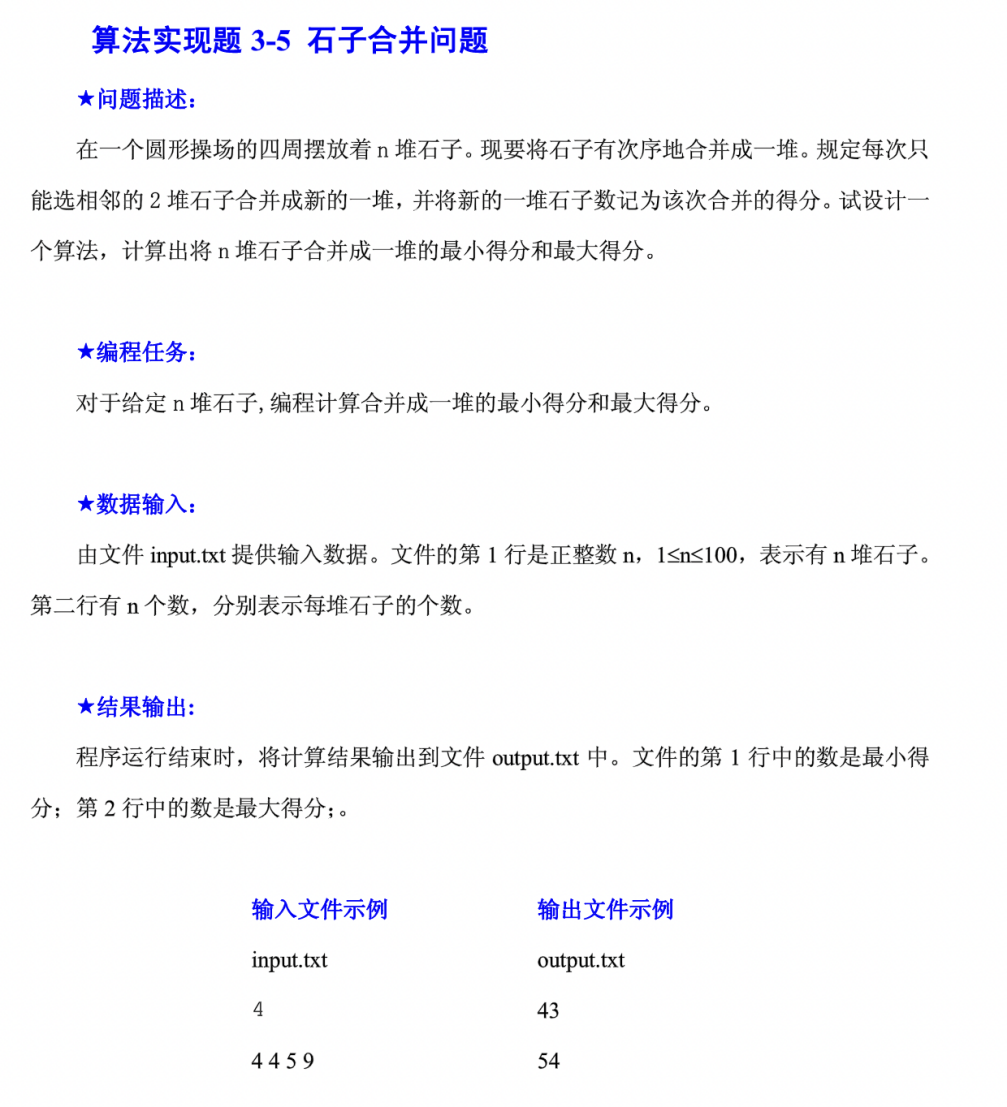
姓名：成凯

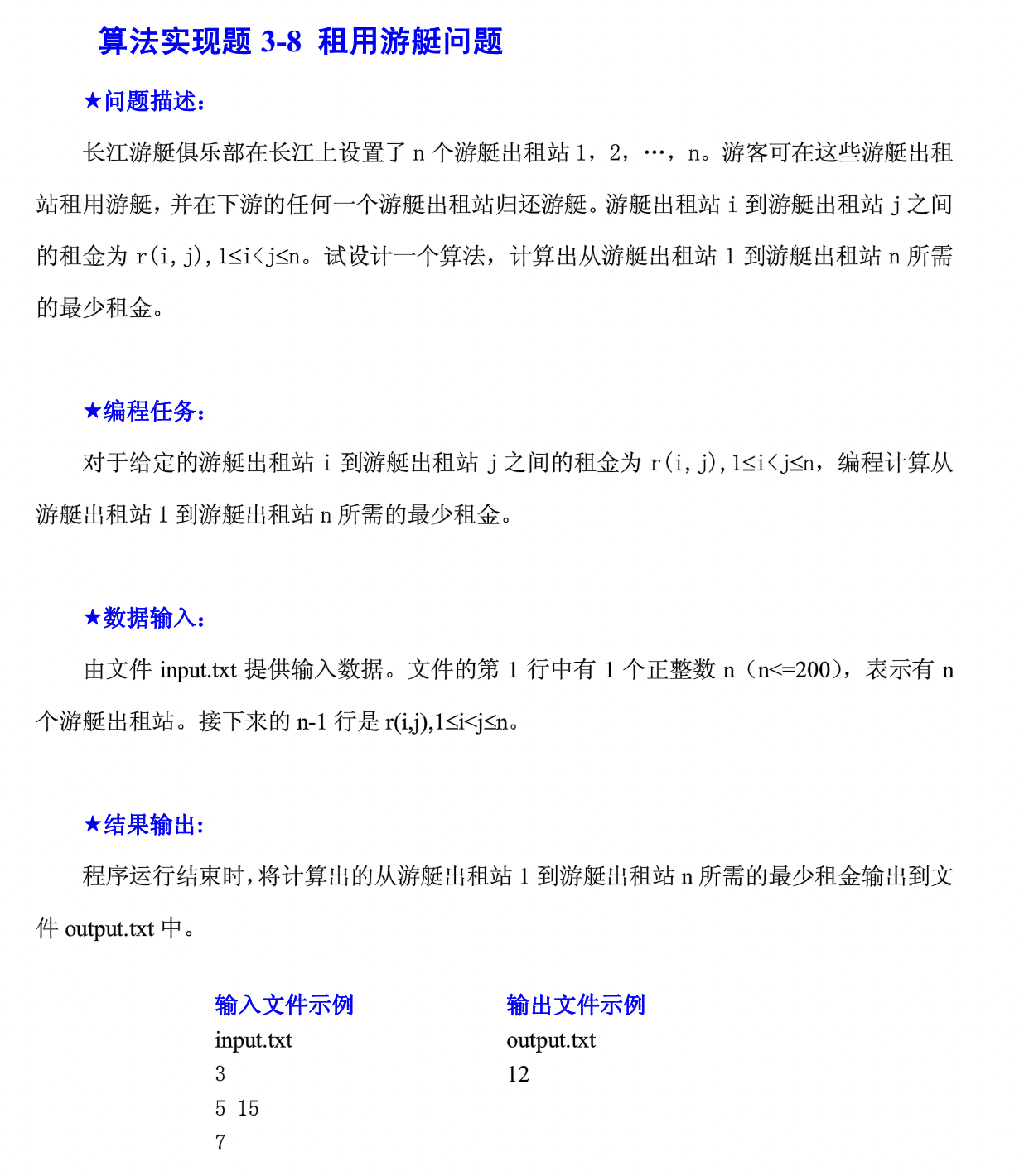
学号：201833050025

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]);

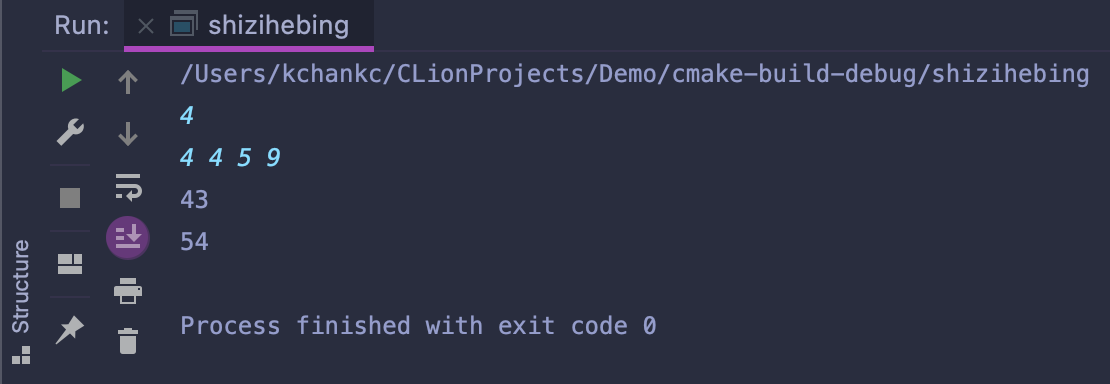
MinValue = min(MinValue, Min[i][i + n - 1]);

}

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]);

}

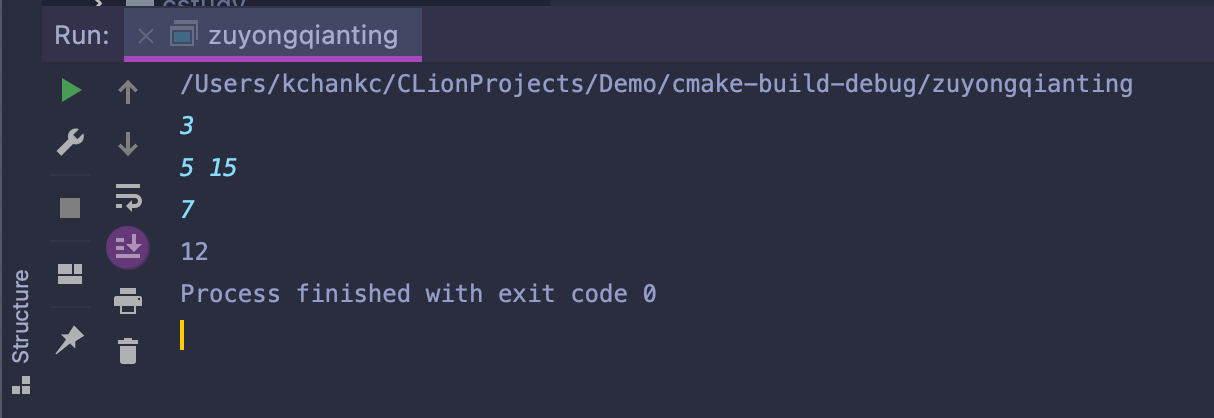
}

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。