

**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

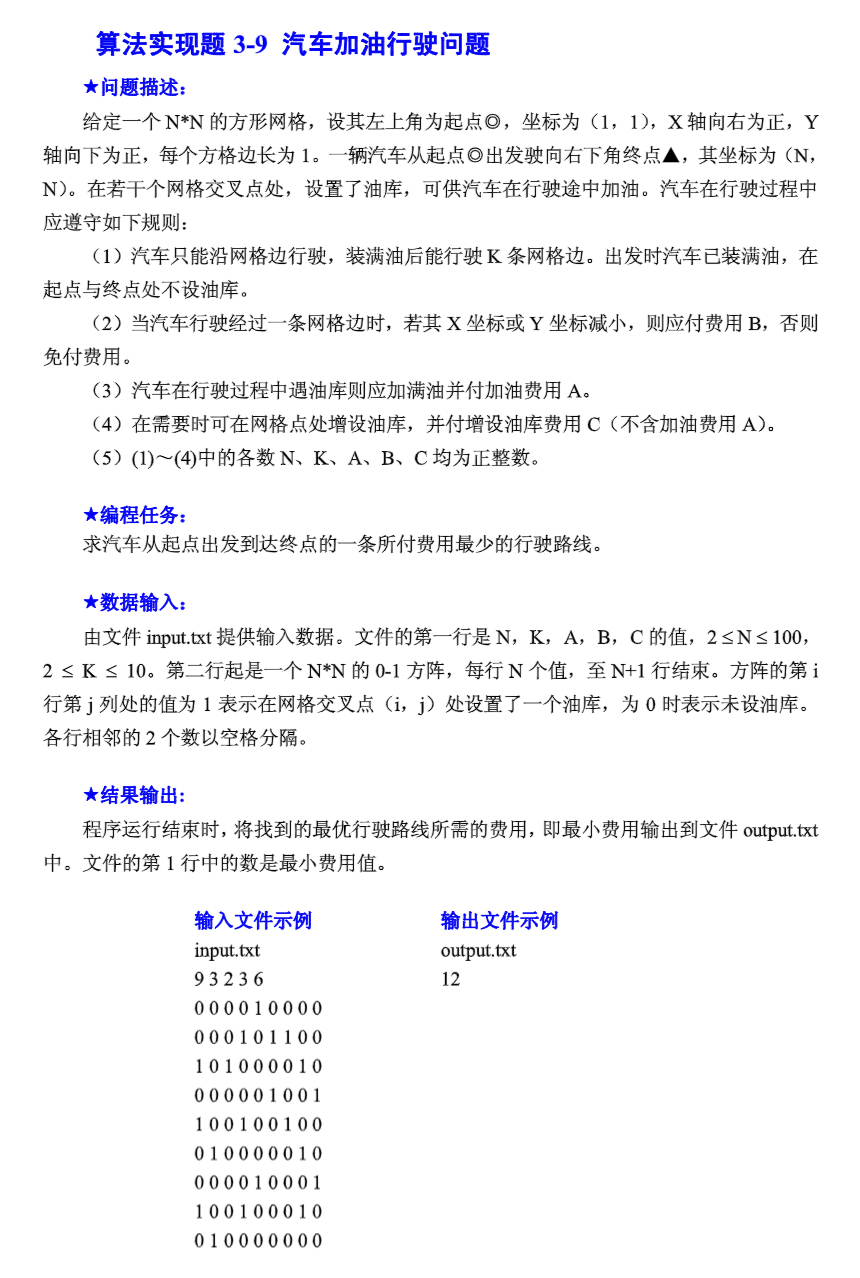
班级：1班

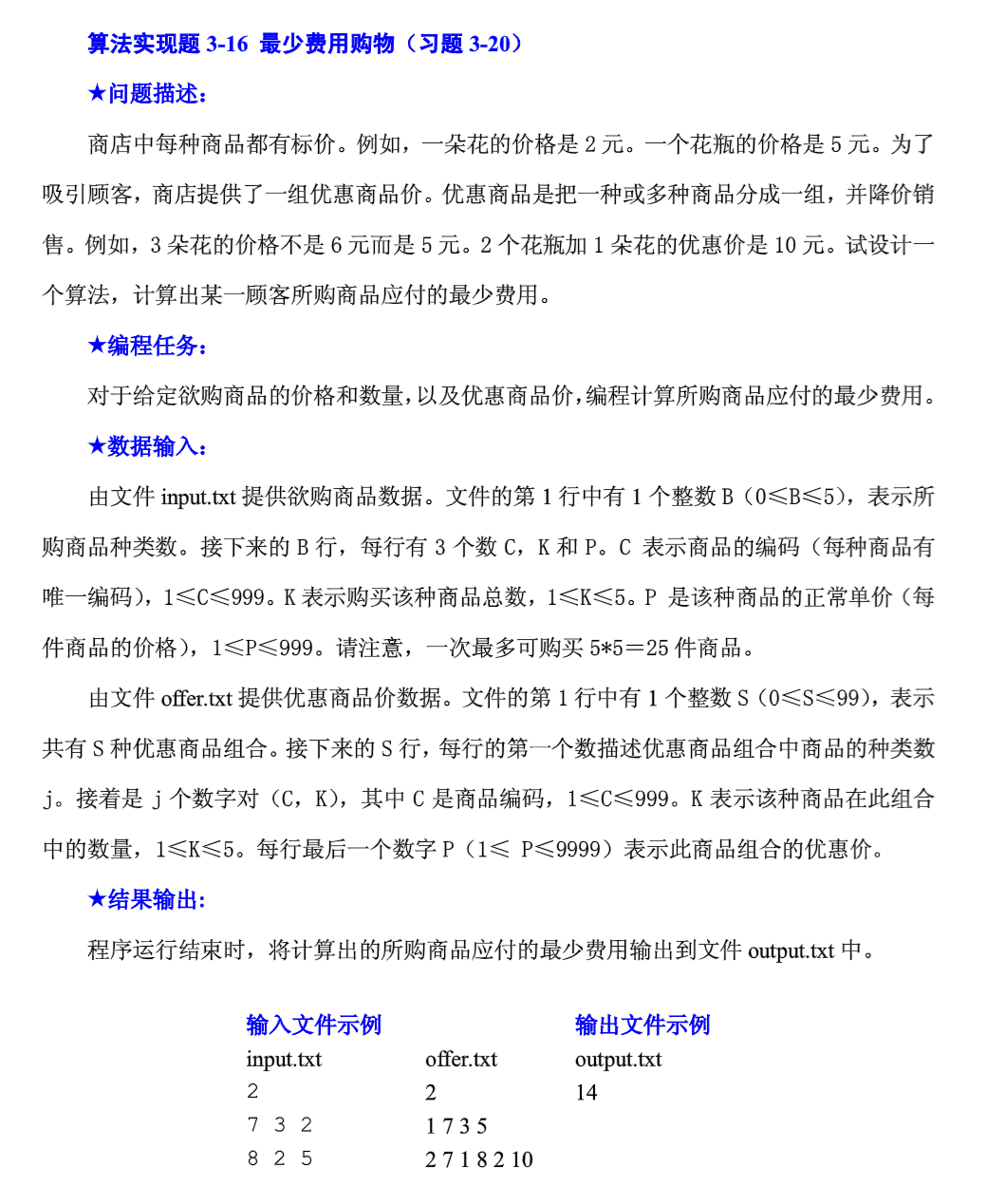
姓名：成凯

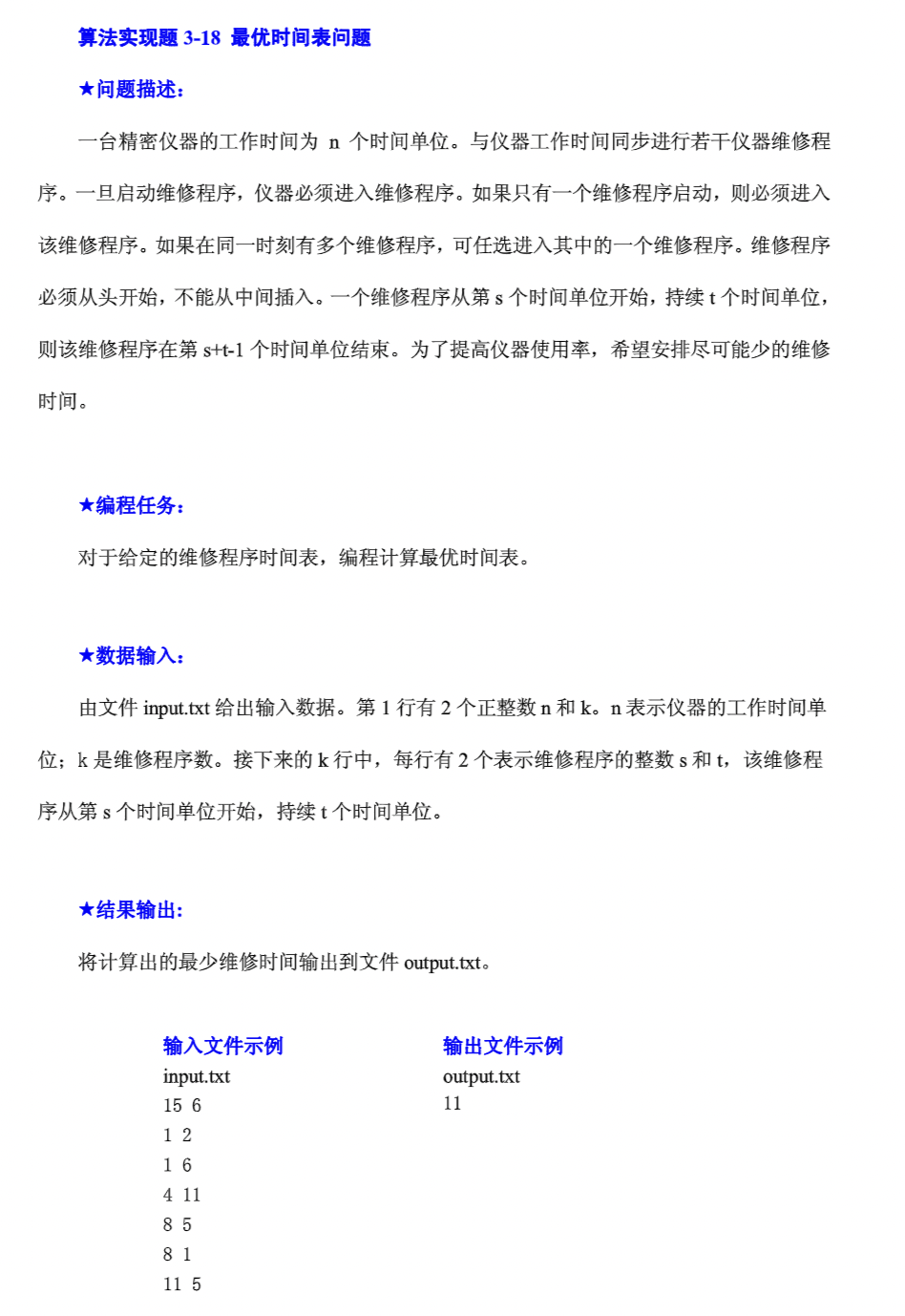
学号：201833050025

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;

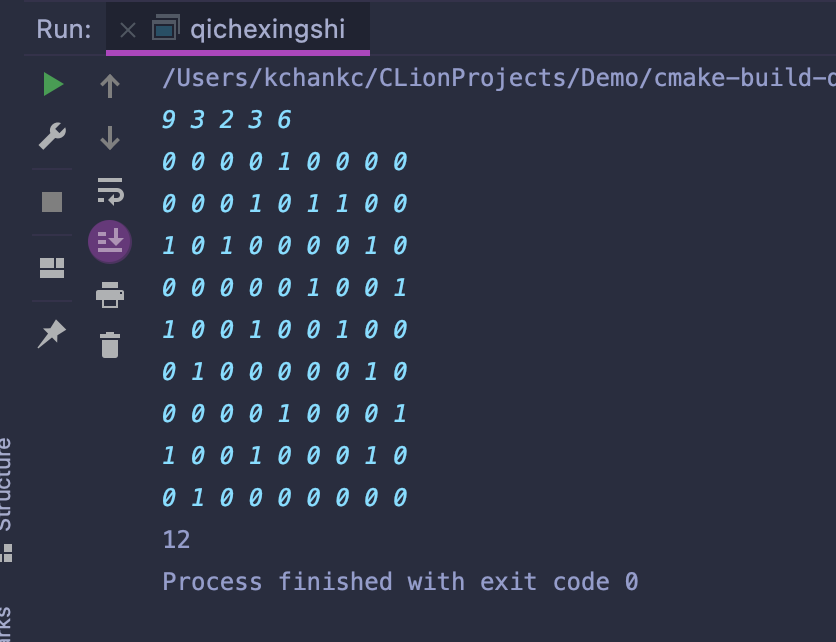
for (int i = 0; i <= k; i++) ans = min(ans, dis[getid(n, n, i)]);

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;

}

}

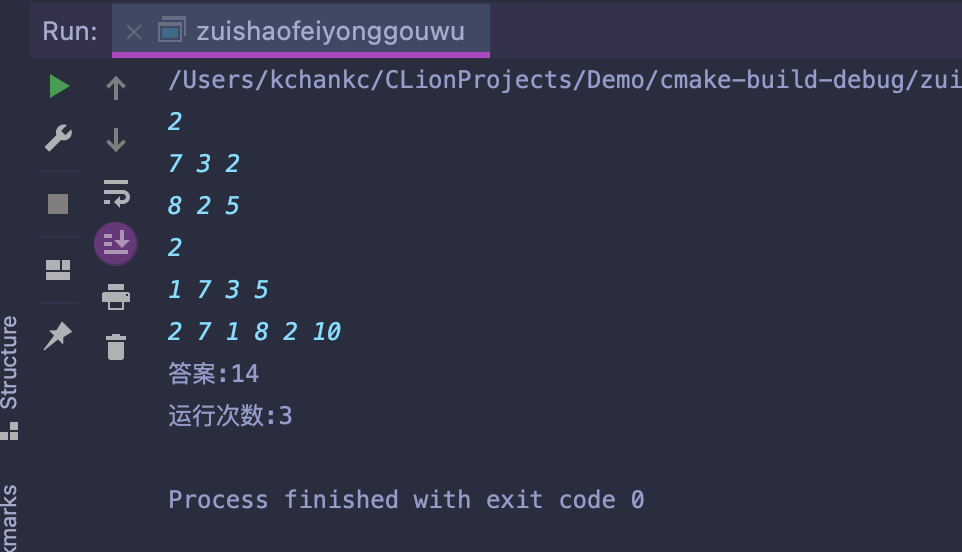
cout << "答案:" << dfs(nums) << endl;

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

}

}

}

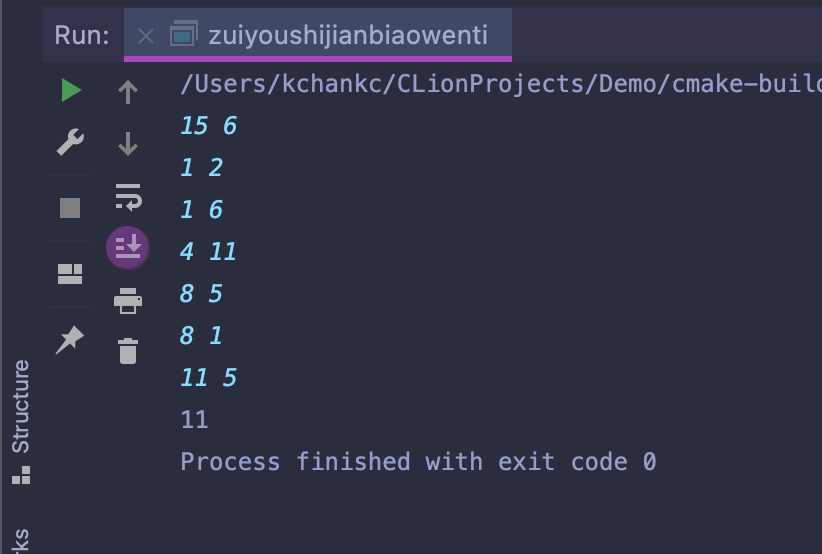
}

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。