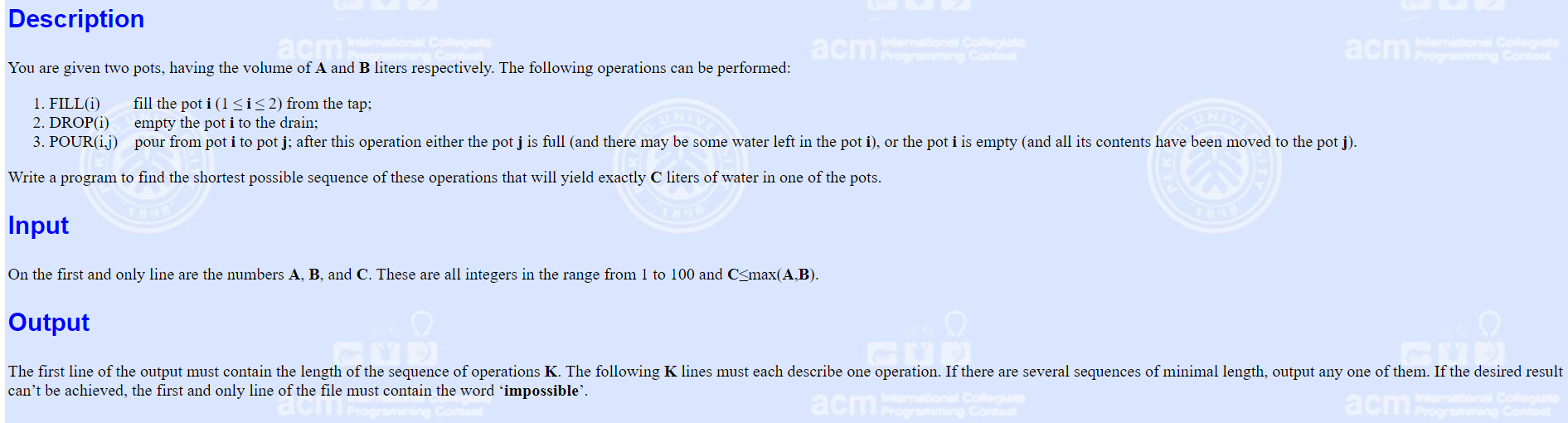
**Ai第四次实验**

前三次实验课，不管是dfs，A\*，迭代加深搜索，在我看来都是dfs的变形，或者说是一种拓展，虽然在这个过程中涉及到了一些bfs的内容，但是整体上来讲，对于bfs没有进行bfs深入的介绍，所以这次实验课，我决定选择一个专门针对bfs的题目，来写一写我对bfs的理解。(当然这个在acm竞赛中应该是一个比较浅显的内容，但是可以通过这次实验内容对bfs做一个更加深刻的分析，与此同时可以增加自己对这一算法思想的理解)。

我依然选择了北大oj上的一道acm题目，poj3414.

**题目**：



**题目大意：**

给出两个壶的容量A和B, 一个目标水量C，对A、B可以有３种操作，求最少经过几步操作能够在某个壶中得到目标水量C。输入A、B和C，输入最少操作数和操作过程。

我的分析：把A和B壶中水量作为状态，初始状态为<0,0>，每个操作都是状态变化的过程。因为有２个壶，所以总共有６种操作。使用ＢＦＳ搜索来找到最少的操作步数。同时需要考虑操作的条件，以减少操作来加快程序运行速度。

通过截图：



**我的ac代码：**

#include <iostream>

#include <queue>

#include <cstring>

#include <string>

using namespace std;

const int MAXN = 100;

int a, b, c;

bool notvist[MAXN+1][MAXN+1];

struct node {

int a, b, level;

char path[MAXN+1];

int plen;

};

string path[] = {

"FILL(1)"

,"FILL(2)"

,"DROP(1)"

,"DROP(2)"

,"POUR(1,2)"

,"POUR(2,1)"

};

void output\_result(int lvl, char p[], int n)

{

cout << lvl << endl;

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

cout << path[(int)p[i]] << endl;

}

void bfs()

{

queue<node> q;

memset(notvist, true, sizeof(notvist));

node f;

f.a = 0;

f.b = 0;

f.level = 0;

memset(f.path, 0, sizeof(f.path));

f.plen = 0;

q.push(f);

notvist[f.a][f.b] = false;

while(!q.empty()) {

f = q.front();

q.pop();

if(f.a == c || f.b == c) {

output\_result(f.level, f.path, f.plen);

return;

}

node v;

v = f;

v.level++;

v.plen++;

// FILL(a)

if(a - f.a > 0) {

v.a = a;

v.b = f.b;

if(notvist[v.a][v.b]) {

v.path[f.plen] = 0;

q.push(v);

notvist[v.a][v.b] = false;

}

}

// FILL(b)

if(b - f.b > 0) {

v.a = f.a;

v.b = b;

if(notvist[v.a][v.b]) {

v.path[f.plen] = 1;

q.push(v);

notvist[v.a][v.b] = false;

}

}

// DROP(a)

if(f.a) {

v.a = 0;

v.b = f.b;

if(notvist[v.a][v.b]) {

v.path[f.plen] = 2;

q.push(v);

notvist[v.a][v.b] = false;

}

}

// DROP(b)

if(f.b) {

v.a = f.a;

v.b = 0;

if(notvist[v.a][v.b]) {

v.path[f.plen] = 3;

q.push(v);

notvist[v.a][v.b] = false;

}

}

// POUR(a,b)

if(f.a && (f.b < b)) {

if(f.a > (b - f.b)) {

v.a = f.a -(b - f.b);

v.b = b;

} else {

v.a = 0;

v.b = f.b + f.a;

}

if(notvist[v.a][v.b]) {

v.path[f.plen] = 4;

q.push(v);

notvist[v.a][v.b] = false;

}

}

if(f.b && (f.a < a)) {

if(f.b > (a - f.a)) {

v.a = a;

v.b = f.b -(a - f.a);

} else {

v.a = f.a + f.b;

v.b = 0;

}

if(notvist[v.a][v.b]) {

v.path[f.plen] = 5;

q.push(v);

notvist[v.a][v.b] = false;

}

}

}

cout << "impossible" << endl;

}

int main()

{

cin >> a >> b >> c;

bfs();

return 0;

}