**Code:**

#include <bits/stdc++.h>

using namespace std;

typedef pair<int, int> pii;

void printpath(map<pii, pii> mp, pii u)

{

if (u.first == 0 && u.second == 0) {

cout << 0 << " " << 0 << endl;

return;

}

printpath(mp, mp[u]);

cout << u.first << " " << u.second << endl;

}

void DFS(int a, int b, int target)

{

map<pii, int> m;

bool isSolvable = false;

map<pii, pii> mp;

queue<pii> q;

q.push(make\_pair(0, 0));

while (!q.empty()) {

auto u = q.front();

q.pop();

if (m[u] == 1)

continue;

if ((u.first > a || u.second > b || u.first < 0

|| u.second < 0))

continue;

m[{ u.first, u.second }] = 1;

if (u.first == target || u.second == target) {

isSolvable = true;

printpath(mp, u);

if (u.first == target) {

if (u.second != 0)

cout << u.first << " " << 0 << endl;

}

else {

if (u.first != 0)

cout << 0 << " " << u.second << endl;

}

return;

}

// completely fill the jug 2

if (m[{ u.first, b }] != 1) {

q.push({ u.first, b });

mp[{ u.first, b }] = u;

}

// completely fill the jug 1

if (m[{ a, u.second }] != 1) {

q.push({ a, u.second });

mp[{ a, u.second }] = u;

}

// transfer jug 1 -> jug 2

int d = b - u.second;

if (u.first >= d) {

int c = u.first - d;

if (m[{ c, b }] != 1) {

q.push({ c, b });

mp[{ c, b }] = u;

}

}

else {

int c = u.first + u.second;

if (m[{ 0, c }] != 1) {

q.push({ 0, c });

mp[{ 0, c }] = u;

}

}

// transfer jug 2 -> jug 1

d = a - u.first;

if (u.second >= d) {

int c = u.second - d;

if (m[{ a, c }] != 1) {

q.push({ a, c });

mp[{ a, c }] = u;

}

}

else {

int c = u.first + u.second;

if (m[{ c, 0 }] != 1) {

q.push({ c, 0 });

mp[{ c, 0 }] = u;

}

}

// empty the jug 2

if (m[{ u.first, 0 }] != 1) {

q.push({ u.first, 0 });

mp[{ u.first, 0 }] = u;

}

// empty the jug 1

if (m[{ 0, u.second }] != 1) {

q.push({ 0, u.second });

mp[{ 0, u.second }] = u;

}

}

if (!isSolvable)

cout << "No solution";

}

int main()

{

int Jug1 = 4, Jug2 = 3, target = 2;

cout << "Path from initial state "

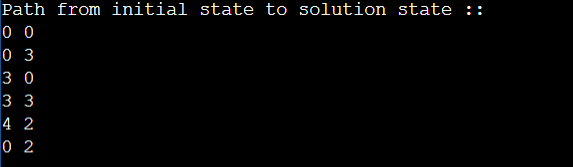
"to solution state ::\n";

DFS(Jug1, Jug2, target);

return 0;

}

**Output:**

****