Assignment2

2.1

1. Explain why problem formulation must follow goal formulation.

The goal formulation is the first step of the problem solving. We can analyze what aspects are needed and what are redundant in goal formulation. Problem formulation is the process of deciding what actions and states to consider, given a goal. Thus, if we do not do goal formulation first, it will be hard to find an accurate and efficient solution.

2. The textbook says that we would not consider problems with negative path costs. In this exercise, we explore this in more depth.

(a) Suppose that actions can have arbitrarily large negative costs; explain why this possibility would force any optimal algorithm to explore the entire state space.

For the optimization algorithm, no matter which path is chosen, it will produce very large results. So if you want to find the optimal solution, you may need to exhaust all states.

(b) Does it help if we insist that step costs must be greater than or equal to some negative constant c? Consider both trees and graphs.

If the search space is a tree, and we know the depth of it is d. So the negative cost action can add at most cd to the result. As a result, if a solution is worse than the best solution minus cd, we can ignore it. But if the search space is a graph, there may be some negative loops and we can never know what the depth it is, so it is helpless.

(c) Suppose that there is a set of operators that form a loop, so that executing the set in some order results in no net change to the state. If all of these operators have negative cost, what does this imply about the optimal behavior for an agent in such an environment?

The agent will always move along this loop.

(d) One can easily imagine operators with high negative cost, even in domains such as route finding. For example, some stretches of a road might have such beautiful scenery as to far outweigh the normal costs in terms of time and fuel. Explain, in precise terms, within the context of state-space search, why humans do not drive around scenic loops indefinitely, and explain how to define the state space and operators for route finding so that artificial agents can also avoid looping.

In the real world, the path with high negative cost will change over time. Take the beautiful road mentioned in the problem for example, the beautiful scenery it has may worth the cost of time and fuel if we first visit it. However, if we visit it at the second time, third time and thousands of time, we will get tired, it will never worth the cost of the time and the fuel. So we can define that if the artificial travel along the negative loop for the second, third, or more time, the absolute value of the negative-cost path will become less and less, and finally, it will get rid of the negative loop.

(e) Can you think of a real domain in which step costs are such as to cause looping?

Real domains like that, if you do not like sports, you will get fat, and it will be more hard for you to do sports, and you will do less sports.

2.2

The missionaries and cannibals problem is usually stated as follows: Three missionaries and three cannibals are on one side of a river, along with a boat that can and must hold either one or two people. Find a way to get everyone to the other side without ever leaving a group of missionaries in one place outnumbered by the cannibals in that place. This problem is famous in AI because it was the subject of the first paper that approached problem formulation from an analytical viewpoint.

1. Formulate the problem precisely, making only those distinctions necessary to ensure a valid solution. Draw a diagram of the complete state space (without repeated states).

States: We use an array of three elements(m,c,b) to describe a state. In the array, m means the number of missionaries in the initial side of river, c means the number of cannibals in the initial side, b means the location of the boat. If the boat locate in the initial side, then b=0, else b=1.

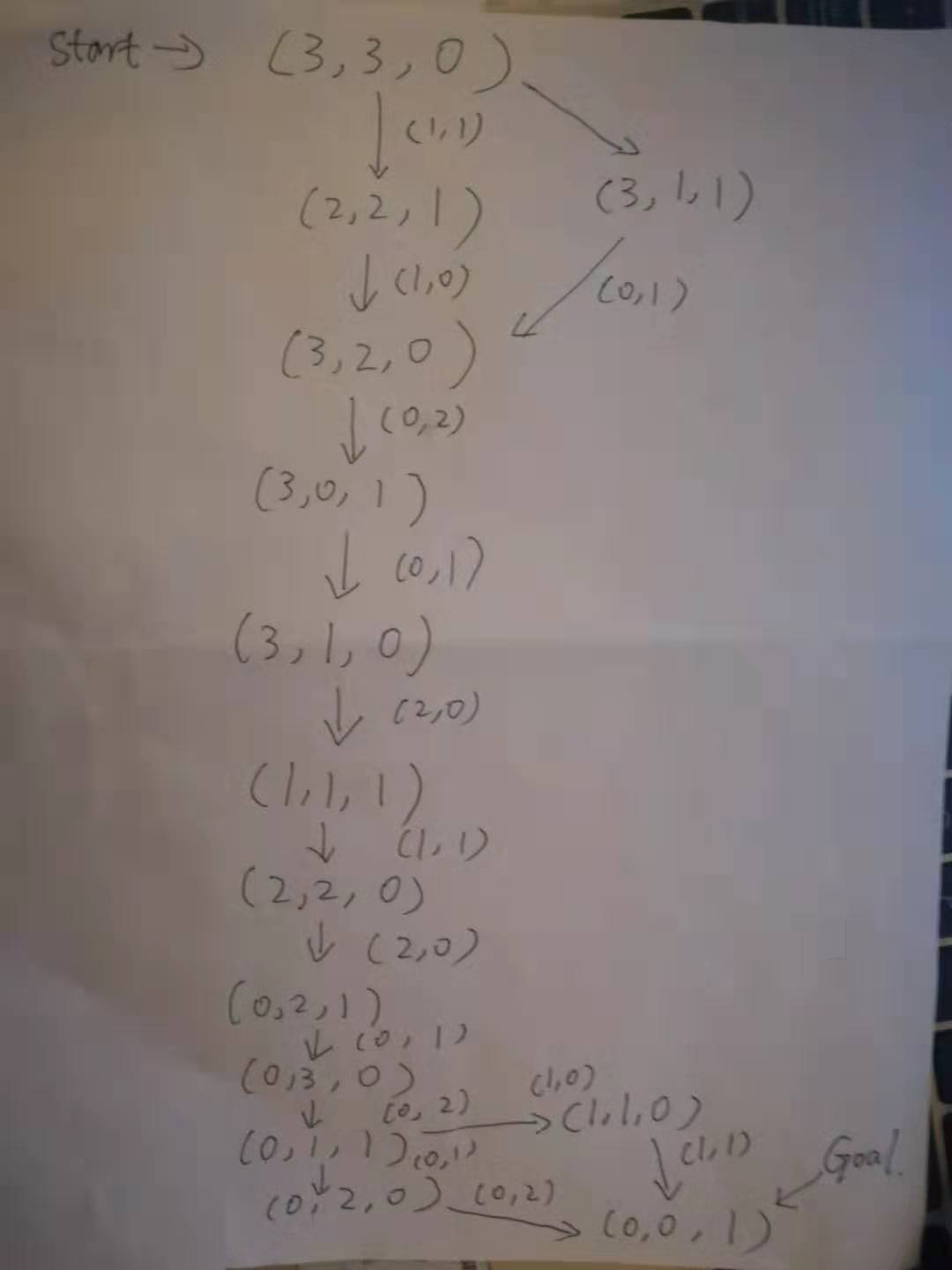
Initial state: there are three missionaries and three cannibals in the initial side, and the boat locates in the initial side. (3,3,0)

Actions: the boat carry one cannibal and one missionary to the other side; the boat carry two cannibals to the other side; the boat carry two missionaries to the other side; the boat carry one cannibals to the other side; the boat carry one missionary to the other side.

Transition model: As the actions of the boats, the number changes in the array.

Goal test: there are three missionaries and three cannibals in the goal side, and the boat locates in the goal side. (0,0,1)

State space:



2. Implement and solve the problem optimally using an appropriate search algorithm. Is it a good idea to check for repeated states?

I implemented the search algorithm in C++ programming, and my strategy for excluding duplicate nodes is to check if they are traversing the queue whenever they traverse to a node, and if so, do not continue searching.

This is my code:

#include <iostream>

using namespace std;

int n=3, space=2,step=0;

int action1[]={1,2,0,0,1};

int action2[]={1,0,2,1,0};

int num1[100], num2[100], num3[100];

int time=0;

void printans()

{

int i;

time++;

cout<<"Solution "<<time<<":"<<endl;

for(i=1;i<=step;i++)

{

cout<<"Step"<<i<<": "<<"("<<num1[i]<<","<<num2[i]<<","<<num3[i]<<")"<<endl;

}

cout<<endl;

return;

}

void search(int m, int c, int boat)

{

int i;

step++;

//cout<<"("<<m<<","<<c<<","<<boat<<")"<<endl;

num1[step]=m;num2[step]=c;num3[step]=boat;

for(i=1;i<=step-1;i++)

{

if((m==num1[i])&&(c==num2[i])&&(boat==num3[i]))

{

step--;

return;

}

}

if ((m==0) && (c==0) && (boat==1))

{

printans();

step--;

return;

}

if (((m>0) && (c>0) & (m<c)) || ((3-m>0) && (3-c>0) && (3-m<3-c)))

{

step--;

return;

}

if (boat==0)

{

for(i=0;i<=4;i++)

{

if ((m-action1[i]>=0)&&(c-action2[i]>=0)) search(m-action1[i],c-action2[i],1);

}

}

else

{

for(i=0;i<=4;i++)

{

if ((m+action1[i]<=3)&&(c+action2[i]<=3)) search(m+action1[i],c+action2[i],0);

}

}

step--;

return;

}

int main(int argc, char\*\* argv) {

search(3,3,0);

return 0;

}

3. Why do you think people (humans) have a hard time solving this puzzle, given that the state space is so simple?

For searching in this state space, humans often forget certain nodes during the search process and cannot perform complex calculations and comparisons.