实验1.1

一、背景

春日午后，人工智能基础课上，你犯起了春困，不小心就睡着了。。。

醒来后，你发现自己处在浩瀚的α星系，你的边上都是不同的星球，你拨弄了几下遥控手杆，发现如果你朝某个方向驾驶宇宙飞船，对应位置的星球会受到特定的电磁力和你**互换位置**，同时，宇宙中存在不可逾越的黑洞，以及神秘的星际通道。

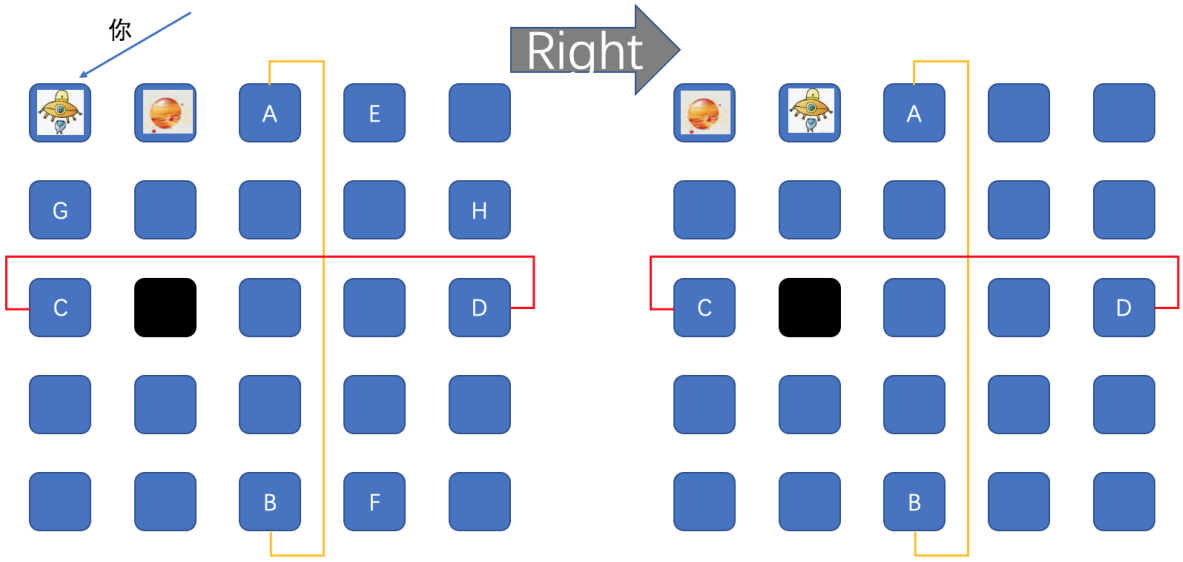
这时指挥部给你下了任务，需要你驾驶宇宙飞船达到某个位置，同时使其他星球呈现某种特定排列。

二、问题描述

输入

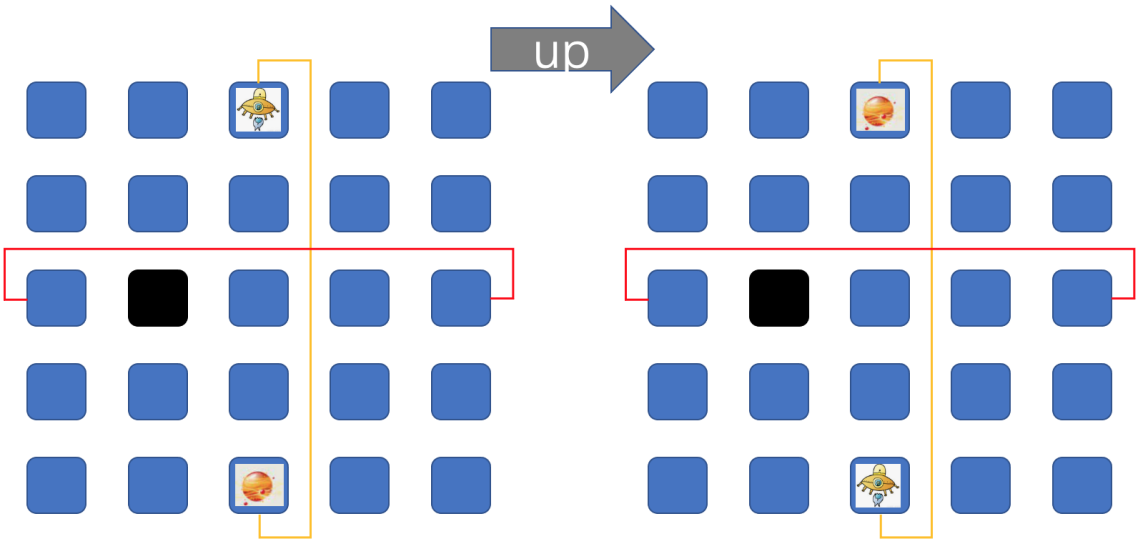
我们将宇宙空间简单化为5*5的方格，其分布所示

每个星球位于其中一个方格内



其中ABCD所在区域存在星际通道（红黄线），表示你在A点往上走可以到达B点，在B点往下走可以到达A点，同理在C点往左走可以到达D点，在D点往右走可以到达C点。而其他边缘方格如GH，EF并无星际通道。黑色方格表示黑洞，为不可行走的区域。

星际通道如下图示意，在左下图的状态下，执行向上（UP）动作，会变成右下图状态：



输入分为初始星球分布和目标星球分布

初始星球分布 对应input.txt：

```
0 2 3 4 5
1 7 8 9 10
6 12 13 14 11
16 17 18 19 -15
21 22 23 24 20
```

目标星球分布 对应target.txt：

```
1 2 3 4 5
6 7 8 9 10
11 12 13 19 14
16 17 18 24 -15
21 22 23 20 0
```

0 代表飞船，其他自然数表示不同编号的星球，黑洞是**负数**。所有数字互不相同，且绝对值均在[0-24]之间。

输出

输出一串动作系列（字符串形式），如 **DDLLDDR** 。

一共有4种动作，U表示向上，L表示向左，R表示向右，D表示向下。

A*算法实现

你需要实现A* 及迭代A *算法来让自己尽快完成任务。

迭代A*如下：

Algorithm 3 Iterative deepening A* search (IDA*)

```
1: d̂_limit ← d̂(s_0)
2: while d̂_limit < ∞ do
3:    next_d̂_limit ← ∞
4:    list ← {s_0}
5:    while list is not empty do
6:       s ← head(list)
7:       list ← rest(list)
8:       if d̂(s) > d̂_limit then
9:          next_d̂_limit ← min(next_d̂_limit, d̂(s))
10:      else
11:         if s is a goal then
12:            return s
13:         end if
14:         newstates ← apply actions to s
15:         list ← prepend(newstates, list)
16:      end if
17:   end while
18:   d̂_limit ← next_d̂_limit
19: end while
20: return fail
```

使用以下两种启发函数：

- h1(n) = number of misplaced stars(错位的星球数)
- 自己设计一种比 h1 更好的可采纳启发式，要求在实验报告中证明其可采纳性，注意曼哈顿距离不可采纳（提示:可以考虑错位的星球数、曼哈顿距离、欧式距离等的变种）。

三、作业要求

1. 编程语言限制为C/C++, 尽量写清核心注释。
2. 请将提交代码命名为a.c/a.cpp，在代码中实现如下定义的 `A_h1` `A_h2` `IDA_h1` `IDA_h2` 4个函数，分别代表A* 和 迭代A* 及2种启发函数。

```
// C++ 定义如下：
#include<vector>
using namespace std;
void A_h1(const vector<vector<int> > &start, const vector<vector<int> >
&target);
void A_h2(const vector<vector<int> > &start, const vector<vector<int> >
&target);
void IDA_h1(const vector<vector<int> > &start, const vector<vector<int> >
&target);
void IDA_h2(const vector<vector<int> > &start, const vector<vector<int> >
&target);
// C 定义如下
void A_h1(const int start[5][5], const int target[5][5]);
void A_h2(const int start[5][5], const int target[5][5]);
void IDA_h1(const int start[5][5], const int target[5][5]);
void IDA_h2(const int start[5][5], const int target[5][5]);
```

3. 请将3个实参传入主函数，第一个实参表示上诉4种函数，只可能为`A_h1` `A_h2` `IDA_h1` `IDA_h2`其中之一，第二个实参表示初始星球分布文件名，第三个实参表示目标星球分布文件名，终端执行时如`./a.out A_h1 input01.txt target01.txt`
4. 输入文件在`data/`目录下。
5. 输出文件共4个，命名为output_A_h1.txt，output_A_h2.txt，output_IDA_h1.txt，output_IDA_h2.txt，分别对应4个函数的运行结果。其中每个文件填入12行，按序对应12组输入，格式为“动作序列”+","+"运行时间(s)" 如`DDLLDDR,0.23`，如果对于某个样例你的代码无法得到结果，填入`x,x`，批改时我们会从初始状态开始检测每一步移动是否合法。
6. 实验报告主要内容包括以下几点：

   (a) 算法的主要思想

   (b) 请使用我们提供的测试用例进行分析，**列表登记**每个样例的运行结果，包括样例编号、运行时间、移动序列、总步数，如果对于某个样例你的代码无法得到结果，登记为（×）。`data/难度说明.txt`里的参考数值不一定是最佳。

   (c) 介绍你的优化方法，对运行空间和时间的优化都会有加分。优化有利于让长步数的样例跑出结果。
7. **严禁抄袭**，批改时会进行代码查重，抄袭以 0 分记。一共要实现 4个核心函数，每缺一个算法此部分将扣25%分数。

实验1.2：作业调度问题

一、问题描述

CSP技术可以用来解决需要满足各种约束的日常生活中的调度问题。考虑下面这个场景，在一个车间中，共有 n 名工人：工人1，工人2， ...， 工人n，每个工人根据工作年限都有自己的级别，比如工人1的级别是：senior，工人2的级别是：junior，现在需要 n 名工人设计一个工作表，比如将工人1和工人2安排到星期一，工人3和工人4安排到星期二，... ，对于车间的工作安排，有以下5条约束要求：

1. 每个工人每周必须休息2天或2天以上
2. 每个工人每周不可以连续休息3天(不考虑跨周情况)
3. 周六周天也需要有人值班，即一周7天每天都要安排工人值班
4. 每天至少要有3个人值班

5. 每天至少要有一名级别为 senior 的工人值班

下面是一个小例子：
车间有5名工人：

- 工人1：senior
- 工人2：junior
- 工人3：junior
- 工人4：senior
- 工人5：junior

为了满足上述的约束，可以将工作表设计如下：

周一	周二	周三	周四	周五	周六	周天
----	工人1	工人1	工人1	工人1	工人1	----
----	工人2	工人2	工人2	工人2	----	工人2
工人3	工人3	----	工人3	----	工人3	工人3
工人4	-----	工人4	----	工人4	工人4	工人4
工人5	工人5	工人5	----	工人5	工人5	----

但由于工人间的个人矛盾，某些人不想在同一天值班，如上面的例子中，工人2不想和工人4一起工作，那么工作表就可以设计成：

周一	周二	周三	周四	周五	周六	周天
----	工人1	工人1	工人1	工人1	工人1	----
----	工人2	----	工人2	----	工人2	----
工人3	工人3	----	工人3	工人3	----	工人3
工人4	----	工人4	----	工人4	----	工人4
工人5	工人5	工人5	----	----	工人5	工人5

二、实验要求

下面请大家为以下两个车间设计工作表

1. 车间1：有7名工人
    - 工人1：senior
    - 工人2：senior
    - 工人3：junior
    - 工人4：junior
    - 工人5：junior
    - 工人6：junior
    - 工人7：junior

由于车间规模较大，需要每天安排至少 **4** 个人值班，其余约束同上，其中工人1和工人4，工人2和工人3，工人3和工人6不想在同一天工作。

2. 车间2：有10名工人

- 工人1：senior
- 工人2：senior
- 工人3：junior
- 工人4：junior
- 工人5：junior
- 工人6：junior
- 工人7：junior
- 工人8：senior
- 工人9：junior
- 工人10：senior

由于车间规模较大，需要每天安排至少 **5** 个人值班，其余约束同上，其中工人1和工人5，工人2和工人6，工人8和工人10不想在同一天工作。

实验任务：

1. 实现一个经过优化的 CSP 搜索算法来给出一个合理的工作安排，可选择的优化方案有：
   - MRV启发式
   - 前向检验
   - 约束传播
2. 上面这个问题能否通过模拟退火算法、遗传算法或者其它你能想到的 local search 算法进行解决，如果有请给出描述思路的伪代码。

三、作业要求：

1. 用C/C++实现上述实验，源码命名为csp.c/csp.cpp
2. 实验的输入与输出：
   (a)输入请按照上面的约束要求自行定义
   (b)输出格式为：
   1 2 3 ...
   1 2 3 ...
   1 2 3 ...
   1 2 3 ...
   1 2 3 ...
   1 2 3 ...
   1 2 3 ...
   其中数字代表从周一到周天的工人编号
   输出文件分别命名为 output1.txt 和 output2.txt
3. 实验报告包括以下几点：
   (a) 描述实验中的变量集合、值域集合以及约束集合
   (b) 算法的主要思路
   (c) 使用的优化方法，并分析使用该优化方法带来的优化效果

实验提交

1. 提交方式：bb系统中提交
2. 截止日期：5月6日晚11:59
3. 提交的目录树结构如下所示：

   SA10000000_张三_exp1\

```
|---digit
        |---src
            |---(your_code)
        |---output
            |---output_A_h1.txt
            |---output_A_h2.txt
            |---output_IDA_h1.txt
            |---output_IDA_h2.txt
|---CSP
        |---src
            |---(your_code)
        |---output
            |---output1.txt
            |---output2.txt
|---report.pdf
```

最后将文件压缩成 .zip 格式进行提交，**注意**，请大家务必按照目录树结构组织文件，否则可能导致检查脚本读取结果失败。

3. 请务必按时提交实验。