AI-lab1

PB19111675 德斯别尔

实验1.1 A* 算法

主要思想

- **A* 主要思想**

  使用可采纳启发式函数 h 与深度 g 之和 f = g + h 作为每个结点的代价, 不断从 open_list 中取出代价最 小的结点进行扩展，新结点加入 open_list ,若是之前出现过的状态，则不加入，直至找到解即为最优解

- **IDA 主要思想**

  使用迭代加深的深度优先搜索, 每确定一个新的代价的界限 limit 后, 不断从栈中弹出结点进行扩展, 将其扩展后的代价不超过 limit 的新结点加入栈中，把扩展过程中扩展到的最小的超过 limit 的结点的代价作为新的界限, 重复以上过程, 直至找到解即为最优解

- **h2 主要思想**

  使用曼哈顿距离的变种:所有的墙壁都有时空隧道可以穿梭，并将所有不在正确位置的星球离正确位置的曼哈顿距离相加。这将必然比只考虑所给的时空隧道的路径要短，也要比`h1()`大，既提升效率，又确保能够找到最优解。

样例测试

void A_h1(const vector > &start, const vector > &target);

样例编号	移动序列	总步数	运行时间
00	DDRUR	5	0.001
01	ULLUULDD	8	0.001
02	DDLUULLURR	10	0.001
03	DLDRRURRRUUURR	14	0.005
04	LUUURULLURDDRDR	15	0.017
05	LLUURRRUURDDDDLUURDD	10	0.033
06	DRDLLULULUUURDRURDRDRRR	23	0.056
07	URRRRDLLLLDRRRRDLLLLDRRRR	25	0.029
08	DLLLDRUUUULDRRRRULDDDDRULDR	27	0.282
09	RDRDLUUUURRUUURDRUUULDLDDDRR	28	3.499
10	DDRRUUUULLULLUULLLLLUURRDDDDRR	30	0.135
11	DRUURDRRDRUULDLULDLDRDLDRURDRURD	32	15.967

样例编号	移动序列	总步数	运行时间
11	DRUURDRRDRUULDLULDLDRDLDRURDRURD	32	15.967

void A_h2(const vector > &start, const vector > &target);

样例编号	移动序列	总步数	运行时间
00	DDRUR	5	0.001
01	ULLUULDD	8	0.001
02	DDLUULLURR	10	0.001
03	DLDRRURRRUUURR	14	0.003
04	LUUURULLURDDRDR	15	0.004
05	LLUURRRUURDDDDLUURDD	10	0.002
06	DRDLLULULUUURDRURDRDRRR	23	0.012
07	URRRRDLLLLDRRRRDLLLLDRRRR	25	0.005
08	DLLLDRUUUULDRRRRULDDDDRULDR	27	0.015
09	RRRRDRUUULDLDLLDRDLUUUURRURR	28	0.254
10	DDRRUUUULLULLUULLLLLUURRDDDDRR	30	0.01
11	DRUURDRRDRUULDLULDLDRDLDRURDRURD	32	0.128

void IDA_h1(const vector > &start, const vector > &target);

样例编号	移动序列	总步数	运行时间
00	DDRUR	5	0.001
01	ULLUULDD	8	0.001
02	DDLUULLURR	10	0.001
03	DLDRRURRRUUURR	14	0.005
04	LUUURULLURDDRDR	15	0.019
05	LLUURRRUURDDDDLUURDD	10	0.033
06	DRDLLULULUUURDRURDRDRRR	23	0.056
07	URRRRDLLLLDRRRRDLLLLDRRRR	25	0.029
08	DLLLDRUUUULDRRRRULDDDDRULDR	27	0.282
09	RDRDLUUUURRUUURDRUUULDLDDDRR	28	3.499
10	DDRRUUUULLULLUULLLLLUURRDDDDRR	30	0.135
11	DRUURDRRDRUULDLULDLDRDLDRURDRURD	32	20.492

void IDA_h2(const vector > &start, const vector > &target);

样例编号	移动序列	总步数	运行时间
00	DDRUR	5	0.001
01	ULLUULDD	8	0.001
02	DDLUULLURR	10	0.001
03	DLDRRURRRUUURR	14	0.002
04	LUUURULLURDDRDR	15	0.004
05	LLUURRRUURDDDDLUURDD	10	0.002
06	DRDLLULULUUURDRURDRDRRR	23	0.056
07	URRRRDLLLLDRRRRDLLLLDRRRR	25	0.009
08	DLLLDRUUUULDRRRRULDDDDRULDR	27	0.031
09	RDRDLUUUURRUUURDRUUULDLDDDRR	28	0.345
10	DDRRUUUULLULLUULLLLLUURRDDDDRR	30	0.004
11	DRUURDRRDRUULDLULDLDRDLDRURDRURD	32	0.148

优化方法

- **优先队列**

  使用优先队列来选取F最小的节点，对其进行探索。
- **MAP**

  标记已经出现过的状态，再次重复出现时，不再考虑。
- **vector**(?)

  由二维数组转换成vector之后，速度快了很多。

实验1.2 作业调度问题

CSP问题的描述方式

- **变量集合**

  每个工人在一周中的某一天是否值班。
- **值域集合**

  {0，1}
- **约束集合**

```
bool workday_check()    //检查是否有工人休息不足
bool day_off_check()    //检查是否有工人过的太舒服
bool num_enough_check()     //检查每天人手是否充足
bool senior_check(int y)     //是否每天都有老师傅镇场
bool dislike_check()        //不和不喜欢的人呆在一起
```

主要思想

选取未取值的变量, 分别为其赋不同的值, 然后继续选取下一个变量, 所有变量均已经赋值或者与约束条件冲突时, 要进行回溯, 重新为上一层变量选取新的值

优化方法

可以使用MRV启发式、前向检验、约束传播。我时间没来得及，所以并没有进行较好的优化。

局部搜索算法

- **模拟退火**

```
function SIMULATED_ANNEALING(csp, schedule)
    current = an initial compelete assignment for csp
    for t = 1 to INFINITE do
        T = schedule(t)
        if (T=0) then return current
        if current is a solution then return current
        var = a randomly chosen conflicted variable from csp.variables
        dE = CONFLICTS(current) - CONFLICTS(next)
        if dE > 0 then current = next
        else current = next only with probability exp{dE/T}
```