Number Klotski Report

STUDENT1: 司旭　STUDENT ID：12111206
STUDENT2: 胡强　STUDENT ID: 12111214

该项目工程基于贪心的行进策略，根据标准格式的输入，能够产生标准格式的输出。Proj 小组由司旭、胡强组成，胡强负责工程的后端代码书写，司旭负责前端与 report 的书写。

一．变量解释

该项目工程的代码部分包含四个类，分别是 Klotski, MainJframe, Search，和 GenData

在介绍该项目工程的基本情况前，我们先解释类 Search 里的一个内部类 State（report 后续内容会提到 State，在此解释，便于读者后续理解）而各个类中的其他变量将会在代码中用注释加以解释

State 类实现了 Comparable 接口,主要用于存储棋盘(本篇 report 将二维数组统称为棋盘)上数据的分布状态，该类中的变量如下：

board 是一维 int 数组，将二维的棋盘数据以一维数组的形式保存

depth 是 int 类型变量，表示从 start 状态走到当前状态所用步数

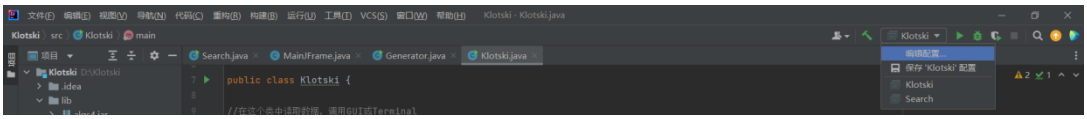
how 是 String 类型变量，表示从上一步走到当前状态的方法，其格式为“移动的数字 移动的方向”,例如“16 U”,表示将左上角数字为 16 的 block(大小可以为 1*1)向上移动一个单位

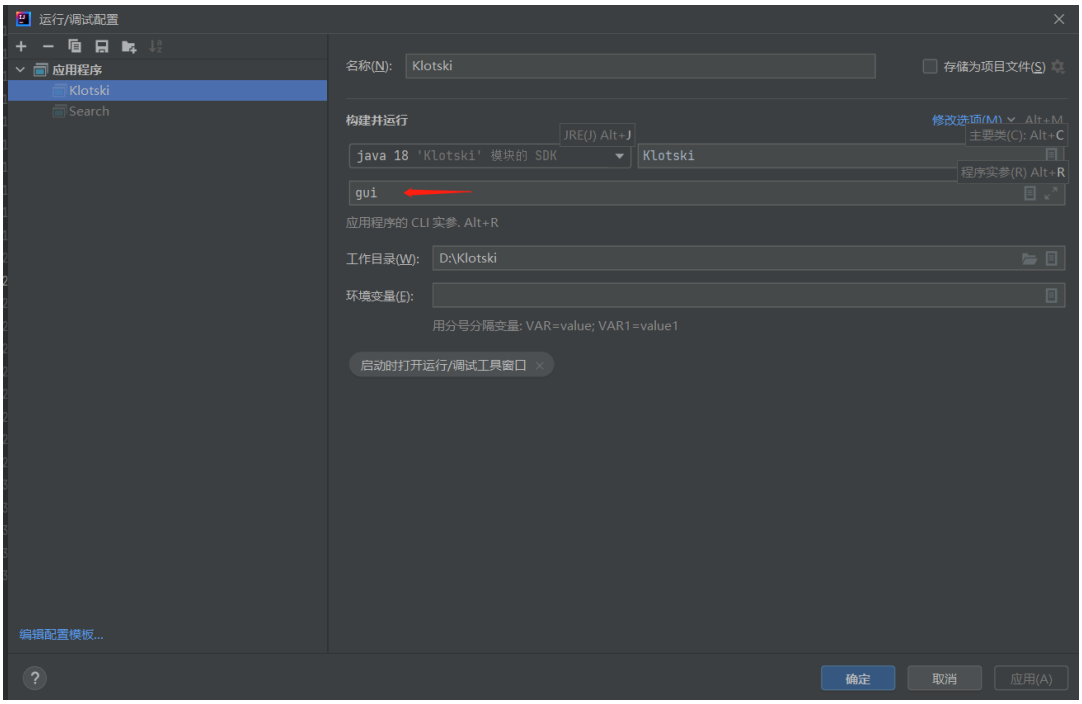
parent 是 State 变量，表示前一步的棋局状态

二．基本结构

（一） 在类 Klotski 中，实现了标准数据的读入；通过命令行传参的方式来判断是通过以 GUI 的形式呈现结果还是在 terminal 中打印输出标准结果；MainJframe 和 Search 均需在 Klotski 中例化，使用整个程序时，只需运行 Klotski 中的 main 方法，然后手动输入标准的数据格式，便可呈现结果

下面是在 IDEA 中实现命令行传参的步骤：





在箭头处输入“gui”或者“terminal”以选择最终呈现结果的方式

（二） 在类 MainJframe 中，实现了图形化界面功能，界面左半部分为数字滑块，右半部分为按钮“GO”，逐次点击便可呈现移动数字滑块的过程

界面如下：

（三） 在类 Search 中，实现了该工程的后端运行逻辑，使用了贪心算法来优化每一次数字滑块的移动，尽可能使总步数更少；并判断了对于每一种输入最终是否可解

（四）在类 Generator 中，通过调用该类中的 main 方法，并指定数据的行数、列数、块状结构的个数以及存储块状结构左上角数字和规模（如“2*1、1*2、2*2）的 String 数组 blocks，便可随机生成可解的输入数据

三．数据结构

（一） 使用了 MaxPQ（在类 Search 中的变量名是 toCheck），用于贪心算法中（即 Search 类里的 greedy 方法）；在说明 MaxPQ 的用途前，先简要陈述找寻最优的下一个状态的依据——位于 State 类中的一个方法 distance，其功能是对于当前状态棋盘上的每一个数字，计算该数字与 end（一维 int 数组，表示棋盘的最终状态）对应的棋盘里的相同数字的曼哈顿距离，再将所有数字得到的距离求和，该和值即为该方法的输出。我们将所有可行的下一状态保存于 MaxPQ 中，重写 compareTo 方法，通过比较 distance 的大小来比较可行的多个下一状态的优劣；据此，MaxPQ 中的最大值即为下一最优状态，通过 delMax 方法可去取出这一最优状态，然后循环此步骤，就可保证我们所走的每一步都尽量朝着 end 靠近

（二）使用了 Queue，用于 bfs（广度优先搜索）；Queue 中存储的是 state 变量，搜索过程中，从队头取出 state，生成下一步骤里包含的所有可能的 state，将得到的 state 插入队列末尾

（三）使用了 ST（符号表），Search 类中的 numer_block 即为一个符号表，用于存储块状结构的信息

（四）使用了 hashSet，用于贪心算法和 bfs 中，其功能是存储此前已经生成的所有 state，用于剪枝，避免重复的 state 出现

四．优化方法

（一）Search 里的 greedy 方法是基于贪心的行进策略，并在 bfs 的基础上做了一些改进，比如将 BFS 所使用的队列更改为了优先队列，提升了运行速度

（二）基于以下定理：

图形 A 与图形 B 等价的充要条件：图形 A 的排列的逆序数加上 0 元素行号和 0 元素列号的奇偶性等于图形 B 的排列的逆序数加上 0 元素行号和 0 元素列号的奇偶性。为方便表述，把图形排列的逆序数加上 0 元素行号和 0 元素列号的奇偶性称为图形的奇偶性。

可以直接判断所给的标准输入是否可解，从而避免了不必要的搜索，提升了效率

五．工程的优点

（一）该工程的最大特点是搜索效率高。相较于 BFS 算法，该工程所实现的 greedy 方法不会出现爆栈的情况，且根据目前的运行情况，对于 4*5 及以下规模数据，可以立刻得出结果，对于 5*5 规模的数据，基本保持在 10s 左右得出结果

（二）GUI 较为美观

（三）可以生成随机的可解数据

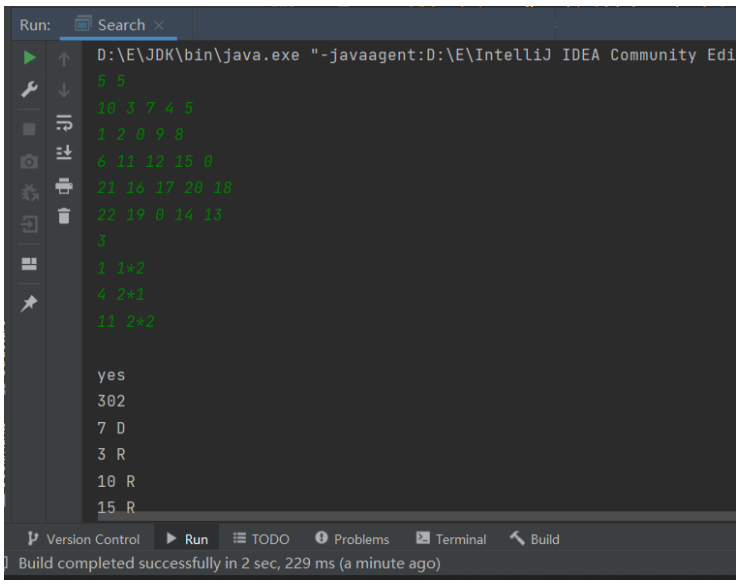
（四）采用了多种搜索方法

六．得分点完成情况

（一） 使用 StdIn 进行标准数据的读入（10 points）

（二） 采用上述定理判断是否可解，如果可解就输出结果，若不能，就输出“NO”（详情请看后续对于一些标准输入的运行结果 10 points）

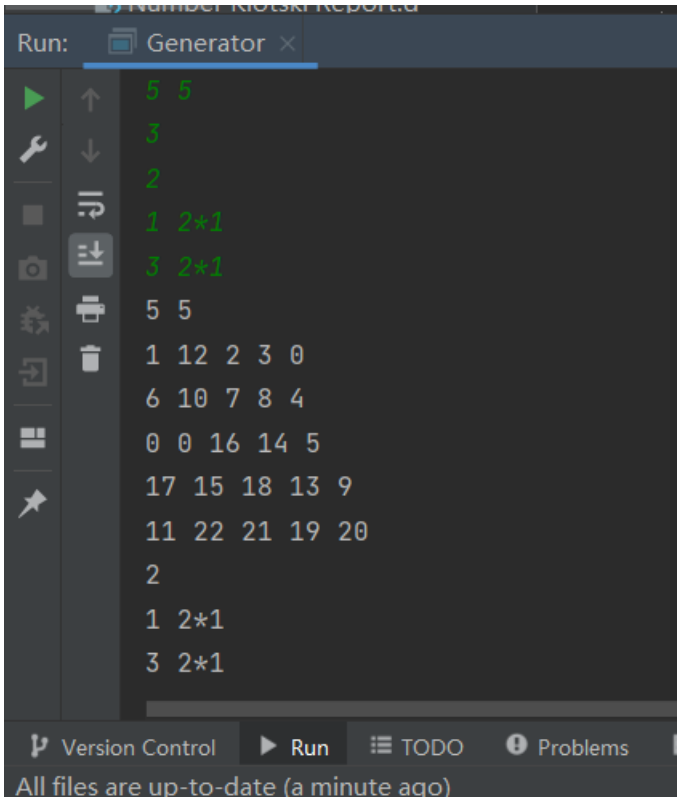
（三） 代码可以在终端上呈现标准输出或者通过 GUI 展现（10 points）



（四） 代码可以处理类似于 2*1、1*2、1*1、2*2 这样的块状结构（10 points）

（五） 程序可以运行 GUI 模块，并且能在一个窗口里展现数字的移动过程，并且有一个按钮可供点击以移动数字（10 points）

（六） 该工程提供了 Generator 类，运行其中的 main 方法就可生成随机数据（10 points）



（七） 该工程除了使用 greedy 方法，还提供了 BFS 方法用于搜索，即使用了不同的方法用于解决问题（10 points）

七．伪代码

Search.greedy:

1）利用相关定理判断盘面是否可解，不可解输出“no”，可解则进行下

述步骤；

2）创建 maxPQ 和 hashSet,将初始盘面状态插入；

3）每次从 maxPQ 中取出最优的盘面（distance 最小），对于每个空格，遍历其上下左右四个方向，判断是否可走以及如何走（是否需要联合周 围的空格），生成下一步的所有盘面；

4）判断下一步的每个盘面是否在 hashSet 中（之前已经生成过），若没有，则将其插入 maxPQ 和 hashSet；

5）循环 3，4 直到全部数字归位。

Search.bfs:

1） 利用相关定理判断盘面是否可解，不可解输出“no”，可解则进行下述步骤；

2） 创建 Queue 和 hashSet,将初始盘面状态插入；

3） 每次从 Queue 中取出队首，对于其中的每个空格，遍历其上下左右四个方向，判断是否可走以及如何走（是否需要联合周围的空格），生成下一步的所有盘面；

4）判断下一步的每个盘面是否在 hashSet 中（之前已经生成过），若没有，则将其插入 Queue 和 hashSet；

5）循环 3，4 直到全部数字归位。

Generator:

1） 读取基本参数(行列数、空格数、大块数、块的第一个数字以及规格)；

2） 构建归位后的盘面；

3） 利用 search 中的移动方法随机移动空格，得到乱序的盘面；

4） 构建标准输出格式，输出样例

八．一些示例的运行结果