**CS 440 MP2 Report (3 credit report)**

Chongye Wang chongye2

Zhengdai Hu zh6

Ruian Pan ruianp2

Report Checklist

* 1. Least stops and shortest distance

1.3.1 random problem generator

1.3.2 Dijkstra for both cases

1.1

**For 1.1 with equal stops**

File: planning\_least\_stops.py

Number of stops: 11

Number of expanded nodes: 2290

Path found: BDAEDCBACDE

**State representation:**

state = (curr\_place, w1, w2, w3, w4, w5)

where w1, w2, w3, w4, w5 are tuples of the remaining factories of each widget

**Actions:** When a path is taken, the curr\_place is changed, the five widget representations change accordingly

**Heuristics:** The number of types of factories that requires visiting at the time for all five widgets. This is clearly admissible and consistent, due to the fact that one cannot possibly take less stops than the number of factories that need visiting.

**For 1.1 with equal distance**

Miles: 5473

Number of expanded nodes: 21115

Path found: AAEBDEAEBAEBBDBEACEBBEA

**State representation:**

state = (curr\_place, w1, w2, w3, w4, w5)

where w1, w2, w3, w4, w5 are tuples of the remaining factories of each widget

**Actions:** **:** When a path is taken, the curr\_place is changed, the five widget representations change accordingly

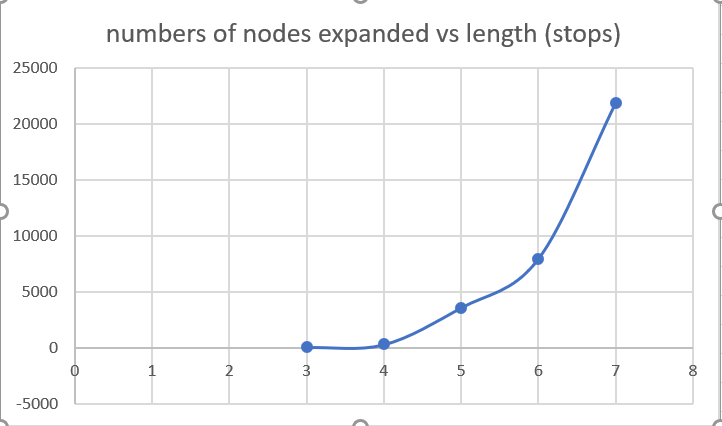
**Heuristics:** the shortest distance from any remaining factory to visit to any of the other factories. This is clearly admissible, because the shortest distance from any factory to the remaining factory is smaller than any potential path.

1.3

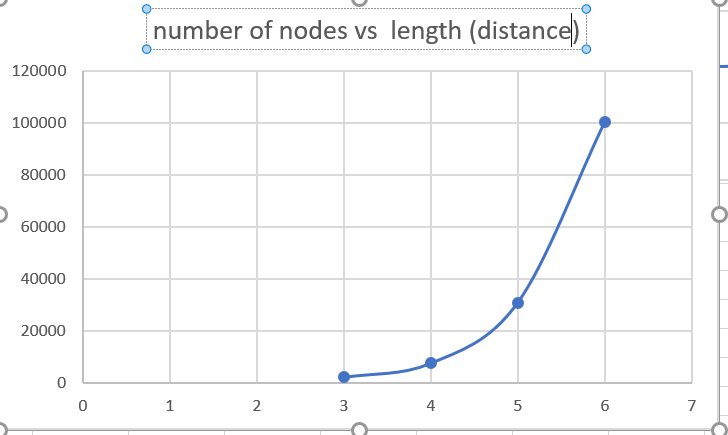
**1.3.1 random problem generator**

Files: mp1311.py mp1312.py

For the least stops cases, the expanded nodes versus length shows:



For the shortest distance cases, the expanded nodes versus length shows:



**1.3.2**

Files: dijkistra\_stops.py dijkstra\_distance.py

For both cases, we changed the Astar algorithm so that it does not use the heuristics anymore.

For Dijkstra stops algorithm, the stops is :11 stops

The expanded nodes is 65592 nodes.

For Dijkstra distance algorithm, the distance is: 5473 miles

The expanded nodes is 24410 nodes.

MP2.1

Path : MP2\_1.mov video in zip file

Initial state : place piece at the center

The rest: following the instructions on the website

Result of reflex agent self play : there is a tie

File : Game folder , Game.java class reflexAgentMove function

MP2.2

Path : MP2\_2.mov video in zip file

In video : black side reflex agent

white side minimax agent

winner : minimax agent

File : Game folder , Game.java class minimaxMove function and alphaBetaMove function

Minimax agent evaluation function:

Check winner : max turn: score 100000

min turn: score -100000

Check continuing 4 pieces having empty spaces on both ends: max turn 3000

min turn -3000

Check continuing 3 pieces having empty spaces on both ends: max turn 200

min turn -200

Check continuing 2 pieces having empty spaces on both ends: max turn 50

min turn -50

Check 1 piece:

Give more weights to pieces at the center areas: max turn +8

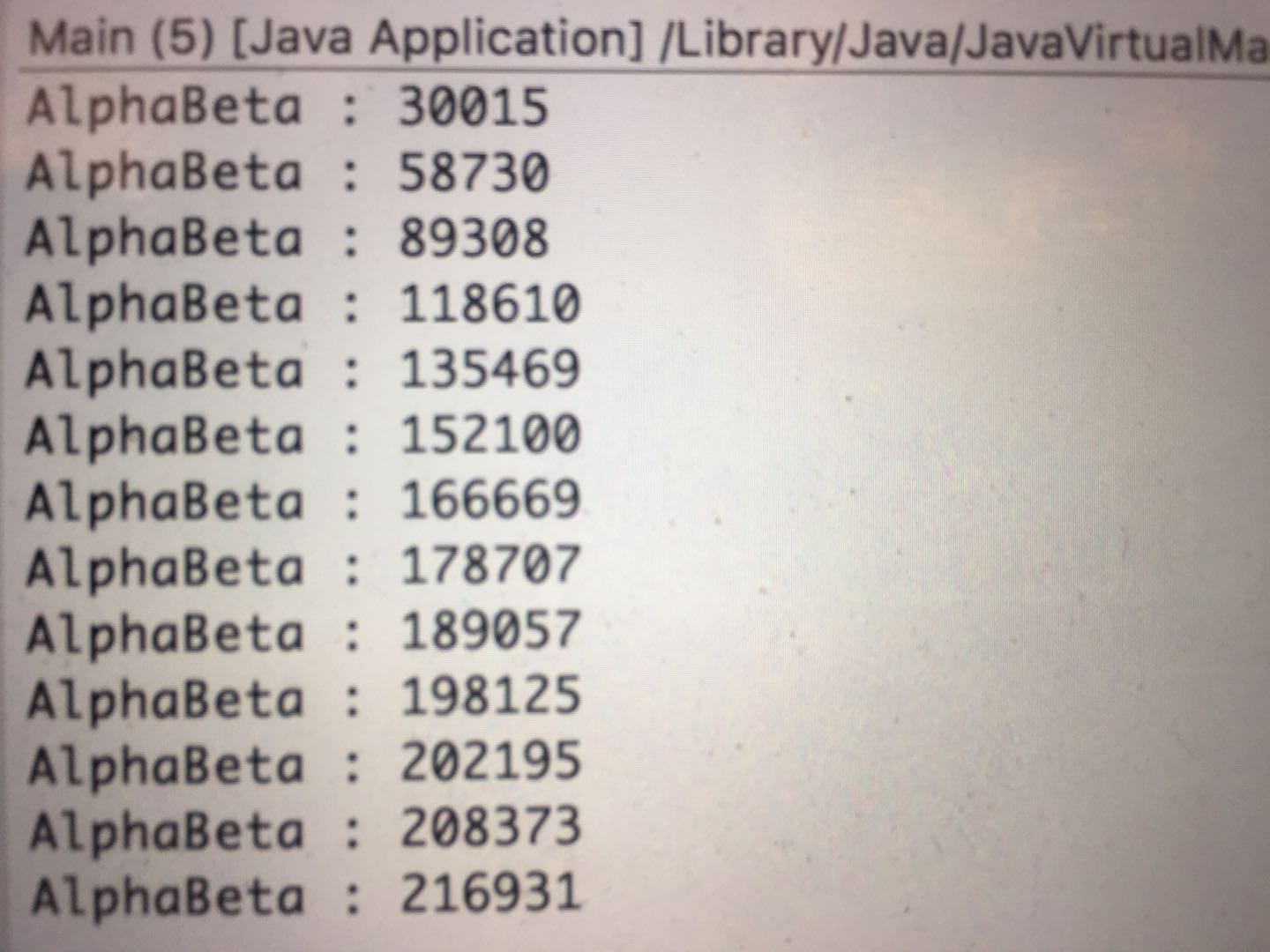
min turn -8

Pieces at the edge : max turn +4

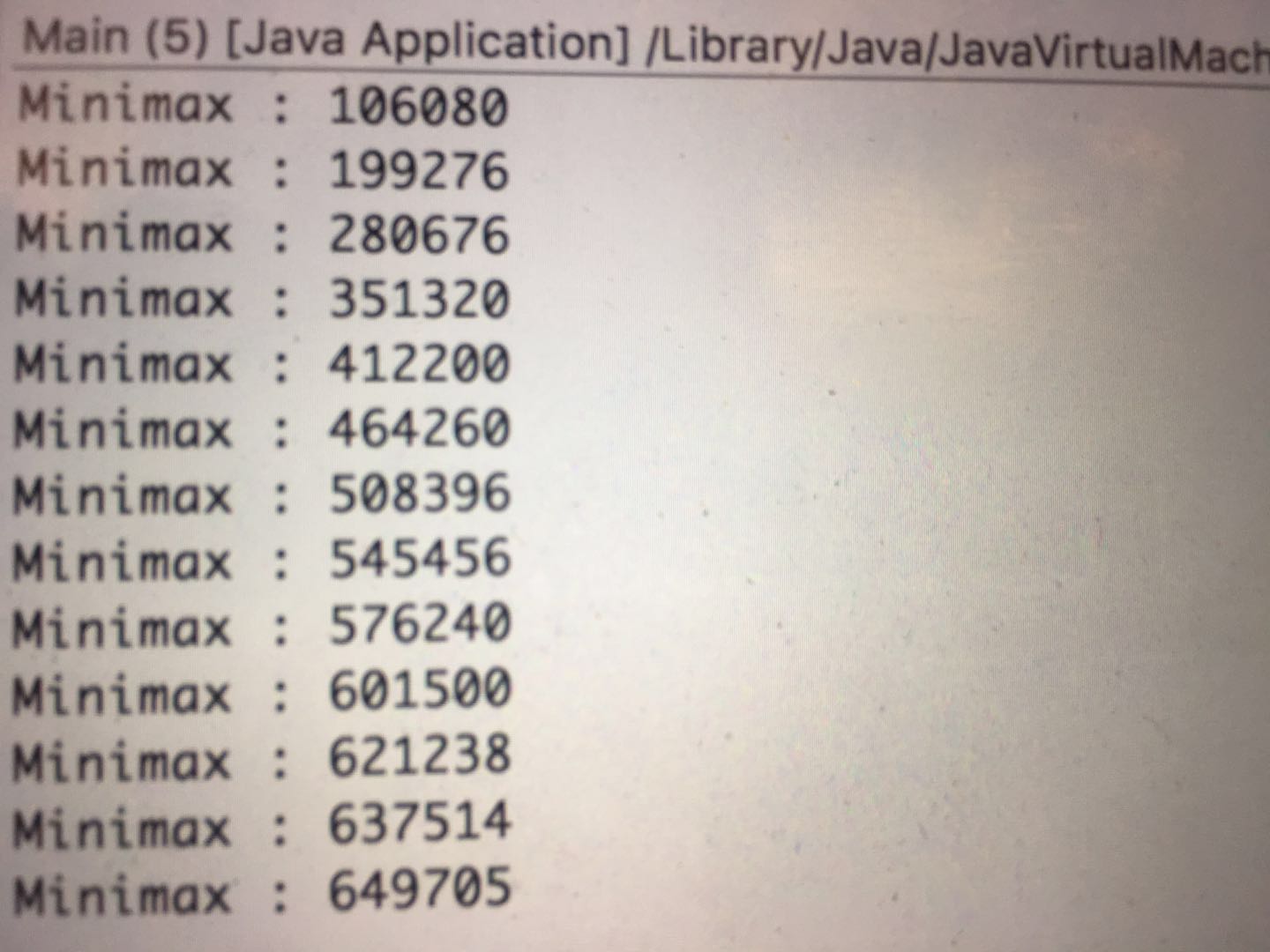
min turn -4

Expanding nodes:

AlphaBeta:



minimax:



**Extra Credits:**

2.4.1 User Interface

Path: MP2\_1.mov video in zip file

Environment : Eclipse

Tool : Java GUI

AI : minimax agent

User : allows the user to play on the board implemented with mouselistener

File : GUI folder uwer\_vs\_AI.java file

2.4.2 Supervised Learning

Method : linear regression

Tool: Using java common math library

Regressor : Generate 3000 matrix representing board conditions

Feature vector:

number of continuing five

number of continuing four having empty spaces on both sides

number of continuing three having empty spaces on both sides

number of continuing two having empty spaces on both sides

Regressand: a vector of size 3000

whether there are wining conditions. If there are, then the entry will be 1

else entry will be zero

File: Game folder SupervisedLearning.java file