**Dominoes Assignment 3 – Design**

**Introduction**

The idea for this assignment is to implement a number of different intelligent dominoes players that use a mixture of different tactics (blocking, winning when can, etc.). The most logical way to start implementing tactics for intelligent players is to think of tactics for the initial board (no dominoes played), the play before either player scores 53 (as this is the minimum possible score away from a one domino win) and finally, tactics for winning or blocking the opponent after scores of 53.

**Tactics**

**Domino (5,4) at Start**

*(5,4)* is the best first domino to play as it scores 3 points and the highest possible score in return is 2, instantly giving the first player a head start. This does not need to be a function as it can be done by simply checking whether *(5,4)* is in the hand and playing it, if it is.

**Start Play Hand**

If (*5,4)* is not in the player’s hand, a naïve way to start play would be to play the highest scoring domino from the hand. However, there are ‘dangerous’ dominoes with which the opponent could score 8 with. Because of this, ‘startPlayHand’ should be a function which checks whether the player has these dominoes (*(6,6),(5,5),(5,2),(6,3)*), returning a hand without these dominoes if the both dominoes for scoring 8 are not contained within the hand.

Logically

1. If all dangerous dominoes contained within hand; return hand passed in.
2. If two of the dangerous dominoes (that can score 8) are contained within hand; return hand minus any other dangerous dominoes.
3. Otherwise, if only one of each 8-scoring dominoes are in hand, return hand minus these dominoes.

**HSD Mid-Play**

Before either of the players score 53 points, the highest scoring domino will be played from the hand.

**Win with One Domino**

Once the intelligent player has hit a score of 53 or higher, it is possible to win with one domino. This tactic can be implemented by using a function called ‘get61’ by looking at the current hand and player’s score then finding and playing a domino that satisfies the remaining score to obtain 61.

Logically

1. Take in a hand of dominoes, scores and ends to play them.
2. Iterate through this list, checking whether any of these dominoes score the exact number to take the player’s score 61; return this domino.

Some extra helper functions will be needed to take the hand, get the viable left and right drops and zip these with their scores and ends on the current board:

getValidDrops

1. Take in a hand and dominoes board.
2. Get the left and right drops from hand.
3. Check that these are not null and zip these with their scores and ends (using getDomScoresAndEnds)
4. Concatenate these two new lists (left and right) and return this.

getDomScoresAndEnds

1. Take in a list of dominoes, and end and a board.
2. Map each of the dominoes in the list to be contained within a triple of domino, the score it obtains on the given board (using scoreDom) and the end to play it; return this new list of triples.

**Get to 59**

This tactic is useful to implement under the same conditions as get61 (once the player has hit a score of 53) but should be done only after no winning domino is found. The reason for getting to 59 is because 2 is the most common score from playing a domino, so if there is no way to win with one domino, it’s beneficial to obtain 59.

Logically

*(see ‘Win with One Domino’, replace 61 with 59)*

**Block When Possible**

It’s not only useful to think how to maximise the speed of winning but also minimising the risk of the opponent winning. This can be done by stopping the opponent from playing whenever possible. This tactic will be implemented using two conditions: the opponent is knocking on the current board and dominoes can be played to keep the ends of the board the same (keeping the opponent knocking).

Opponent knocking

1. Get the player of the last move played on board.
2. If this is the same as the player currently choosing a domino to play, opponent is knocking for current board.

Keep Board Ends Same

1. Check if any combination of the ends of the current board is a domino contained within the left and right drops from the current hand.
2. If one is found, return it to be played.

**Stop Opponent Winning**

It is useful to check the opponent’s score to see whether they are getting close to winning. If the opponent has hit a score of 53 or higher, then they are able to win with one domino. A good tactic to implement would be to stop the opponent from winning. This can be done by checking the opponents score, then if it is 53 or higher, only playing dominoes which do not allow the opponent to win.

Logically (do play bad dominoes)

1. Take in:
   1. A list of dominoes representing the remaining dominoes from the entire set, assumed to be the opponents hand (getRemDoms)
   2. All playable dominoes from the current hand.
   3. The board.
   4. The opponent’s score.
2. Iteratively play each playable domino onto the original board, observing the scores of the remaining dominoes
3. If any of the remaining dominoes take the opponent’s score up to 61, add the played domino to a list.
4. Return the list of dominoes not to play.

Some extra helper functions are needed to get the remaining dominoes and the opponents score.

getRemDoms

1. Take in a hand and a board.
2. Delete all dominoes in the given hand and board from the full set of dominoes; return this new list.

The remaining dominoes can also be modified each time the opponent knocks, as this is a tell that the opponent doesn’t have any dominoes containing the same number of pips as the pips on the ends of the board. Every domino containing exactly these pips can be deleted from the remaining dominoes list as it is representing the opponent’s hand.

**winPlayer**

This player will implement tactics (in this order):

* Domino (5,4) at Start
* Start Play Hand
* HSD Mid-Play
* Win with One Domino

To summarise, it will play (5,4) if it can, else play a non-dangerous domino. Then play highest scoring domino from hand until score >= 53, then try to win with one domino, otherwise play highest scoring domino.

*(hsdPlayer not included as assumed knowledge)*

**winPlayer**

Return (5,4) if can, play hsd, return winning dom if can

Hand,DomBoard,Player,Scores

(Dom,End)

**startPlayHand**

Returns hand without dangerous dominoes

Hand

Hand

**playerScore**

Gets the score of the current player

Player,Scores

Int

**getValidDrops**

Return all playable dominoes with their scores and ends

Hand,DomBoard

[(Dom,Int,End)]

**getDomScoresAndEnds**

Return a list of dominoes with their scores and ends

[(Dom,Int,End)]

Dom,End,DomBoard

**get61**

Find a domino that can take the player’s score to 61

[(Dom,Int,End)],Int

(Dom,End)

**getClosePlayer**

This player will implement tactics (in this order):

* Domino (5,4) at Start
* Start Play Hand
* HSD Mid-Play
* Win with One Domino
* Get to 59

To summarise, it will play (5,4) if it can, else play a non-dangerous domino. Then play highest scoring domino from hand until score >= 53, then try to win with one domino, otherwise play domino to get to 59, else play highest scoring domino.

*(hsdPlayer not included as assumed knowledge)*

**winPlayer**

Return (5,4) if can, play hsd, return winning dom if can

Hand,DomBoard,Player,Scores

(Dom,End)

**startPlayHand**

Returns hand without dangerous dominoes

Hand

Hand

**playerScore**

Gets the score of the current player

Player,Scores

Int

**getValidDrops**

Return all playable dominoes with their scores and ends

Hand,DomBoard

[(Dom,Int,End)]

**getDomScoresAndEnds**

Return a list of dominoes with their scores and ends

[(Dom,Int,End)]

Dom,End,DomBoard

**get61**

Find a domino that can take the player’s score to 61

[(Dom,Int,End)],Int

(Dom,End)

**get59**

Find a domino that can take the player’s score to 59

(Dom,End)

[(Dom,Int,End)],Int

Dom

**winPlayer**

Return (5,4) if can, play hsd, return winning dom if can

Hand,DomBoard,Player,Scores

(Dom,End)

**startPlayHand**

Returns hand without dangerous dominoes

Hand

Hand

**playerScore**

Gets the score of the current player

Player,Scores

Int

**getValidDrops**

Return all playable dominoes with their scores and ends

Hand,DomBoard

[(Dom,Int,End)]

**getDomScoresAndEnds**

Return a list of dominoes with their scores and ends

[(Dom,Int,End)]

Dom,End,DomBoard

**get61**

Find a domino that can take the player’s score to 61

[(Dom,Int,End)],Int

(Dom,End)

**orderDom**

Ensure domino is arranged highest pips first

Dom

**getEnds**

get the ends of the current board

DomBoard

(Int,Int)

**leftdrops**

get all dominoes playable on left of board

**rightdrops**

get all dominoes playable on right of board

Hand

Hand,

DomBoard

Hand,

DomBoard

Hand

**getLastMovePlayer**

get the player of the last move made

**getHistory**

get the history from the DomBoard

Player

History

History

DomBoard

**blockingDom**

get the domino which keeps the ends of the board the same

Dom,Hand,DomBoard

Dom,End

**blockPlayer**

This player will implement tactics (in this order):

* Domino (5,4) at Start
* Start Play Hand
* Block When Possible
* HSD Mid-Play
* Win with One Domino

To summarise, it will play (5,4) if it can, else play a non-dangerous domino. Then block the other player when possible, else play highest scoring domino from hand until score >= 53, then try to win with one domino, otherwise play highest scoring domino.

*(hsdPlayer not included as assumed knowledge)*

Dom

**winPlayer**

Return (5,4) if can, play hsd, return winning dom if can

Hand,DomBoard,Player,Scores

(Dom,End)

**startPlayHand**

Returns hand without dangerous dominoes

Hand

Hand

**playerScore**

Gets the score of the current player

Player,Scores

Int

**getValidDrops**

Return all playable dominoes with their scores and ends

Hand,DomBoard

[(Dom,Int,End)]

**getDomScoresAndEnds**

Return a list of dominoes with their scores and ends

[(Dom,Int,End)]

Dom,End,DomBoard

**get61**

Find a domino that can take the player’s score to 61

[(Dom,Int,End)],Int

(Dom,End)

**orderDom**

Ensure domino is arranged highest pips first

Dom

**getEnds**

get the ends of the current board

DomBoard

(Int,Int)

**leftdrops**

get all dominoes playable on left of board

**rightdrops**

get all dominoes playable on right of board

Hand

Hand,

DomBoard

Hand,

DomBoard

Hand

**getLastMovePlayer**

get the player of the last move made

**getHistory**

get the history from the DomBoard

Player

History

History

DomBoard

**blockingDom**

get the domino which keeps the ends of the board the same

Dom,Hand,DomBoard

Dom,End

[(Dom,Int,End)],Int

**get59**

Find a domino that can take the player’s score to 59

**superPlayer**

This player will implement tactics (in this order):

* Domino (5,4) at Start
* Start Play Hand
* Block When Possible
* HSD Mid-Play
* Win with One Domino
* Get to 59

To summarise, it will play (5,4) if it can, else play a non-dangerous domino. Then block the other player when possible, else play highest scoring domino from hand until score >= 53, then try to win with one domino, otherwise play domino to get to 59, else play highest scoring domino.

*(hsdPlayer not included as assumed knowledge)*

(Dom,End)

**blockOPWinPlayer**

This player will implement tactics (in this order):

* Domino (5,4) at Start
* Start Play Hand
* HSD Mid-Play
* Win with One Domino
* Stop Opponent Winning

To summarise, it will play (5,4) if it can, else play a non-dangerous domino. Then check if opponent’s score >= 53, if it is, play blocking dominoes, otherwise play highest scoring domino from hand until score >= 53, then try to win with one domino, otherwise play highest scoring domino.

*(hsdPlayer not included as assumed knowledge)*