Assignment 2 - Queries on the Othello Database

Team #50
- George Paul (2021121006)
- Amal Sunny (2021121011)

Amendments

The essence of the previous page design is maintained. An addition to each row is made over the previous page design - The position of the disc placed in that move is appended at the end. A segment of an example final page design:

```
...
16e11b1e6w6e2w2b3e3w5e4b1e1w1e1w1e16;12,34
17e11b1e6w6e1b6e3w5e4b1e1w1e1w1e16;43,21
//                                   ^latest turn
...
```

Similarity and KNB

Defining Similarity

Consider a board state P and another board state Q.
Define a Match Matrix M where

$$m_{ij} = \begin{cases} 2 & \text{if } p_{ij} = q_{ij} \\ 1 & \text{if either } p_{ij} \text{ or } q_{ij} \text{ are empty and the other is filled} \\ 0 & \text{if both } p_{ij} \text{ and } q_{ij} \text{ are filled and differ (ie black and white)} \end{cases}$$

Let

$$S(P, Q)$$

be the similarity between the two states.
Now, for every element m_{ij} calculate

$$x_{ij} = \sum \text{surrounding 8 elements}$$

Finally, $S(P, Q)$ is calculated.

$$S(P, Q) = \sum x_{ij}$$

KNB

The entire database is first indexed using the number of moves completed at a particular board state.
An example index block segment would look like

```
...
13;<record pointer>
13;<record pointer>
14;<record pointer>
...
```

Now for a board P on the i^{th} turn, to get K_{NB} we navigate to the row in the index with i turns completed and for each subsequent board Q listed in the index, calculate $S(P, Q)$ and store the boards in an intermediate result as:

```
<similarity score>;<record pointer>
```

Next, sort this list and return the top K boards.
The boards will be returned in a sequence of blocks where block B_n will contain board states that have completed n turns more than board P.

```
---Block 1---
<board states that have completed 1 more turn than P>

---Block 2---
<board states that have completed 2 more turns than P>

---Block 3a---
<board states that have completed 3 more turn than P>
<pointer to block 3b>
---Block 3b--- (in case of an overflow)
<more board states that have completed 1 more turn than
P>
```

Query 1 - Longevity

Given, the board configuration P and the k-nearest boards K_{NB} along with a position (i, j) on the board configuration P.
Assumption: The given position (i,j) would always be coloured. i.e.
$P(i, j) = W \text{ or } B$

a) Whether colour gets flipped in next move

- We query for the K_{NB} and then iterate through all the board states in B_1. Let that board be J_i.
- We compare $J_i(i, j)$ and $P(i, j)$ and check whether the colour has changed (i.e only when both of them are coloured, and of the opposite/same colours).
- If it has, or is the same colour - we count the vote of the J_i board and count across all boards.
- The majority of the vote is used to return the answer of whether the piece will get flipped or not.

b) Guarantee how long the position will not be flipped

- We repeat the same procedure as above, however this time we iterate across all B_i until we reach the end of all boards.
- However unlike above, we do not check for votes - if we encounter an $J_i(i, j)$ & $P(i, j)$ such that $J_i(i, j)$ = opposite colour of $P(i, j)$ we terminate the loop there and return the index of the block we reached.

This gives us the furthest we had to go to get one board where the piece gets flipped, i.e how long it will not be flipped.

c) All potentially flippable positions in next move

- All pieces that are going to become flippable implies the pieces which are currently not flippable but will be flippable in the next turn.
- Essentially, pieces on the current board which have a longevity of 1 - i.e are guaranteed to not be flippable for the next turn but there exists some move to flip them in the turn after that.
- For this, we need to only run the algorithm for the b) part of this question on every piece currently on the board. Those pieces which are guarenteed to not be flipped only for the next move (longetivity = 1) are returned, as they become flippable in the next turn in some board.

Query 2 - Diff in Flips

For every board state in the K_{NB}:

1. Go to that entry in the database.
2. Traverse through the next board state and calculate $(k - l)$ and store the ones that are at least b in an intermediate result, along with the turn number.
3. This intermediate result is returned for query 2a.
4. The list is traversed again to find the longest sequence of moves. This interval of move numbers is returned as the result for 2b.

Query 3 - Max Flips

For every board state in the K_{NB}:

1. Access the record in the database.
2. Traverse to the next board state and check the move that's been played.
3. Check only the rows for which there can be a change. i.e. horizontally, vertically and on both diagonals.
4. Tally the number of times a cell has changed in a intermediate matrix.
5. Return the highest value in the matrix after the game has been gone through.

Query 4 - Unchanged

- One easy heuristic to find some easy positions that would not change is the corners - if they are occupied, it is a known fact that those positions would not change. So if corners are marked with some colour, we can return those positions and subsequently other pieces along the edges that are of the same colour as these edge pieces.
- For the remaining pieces, we have to check the longetivity. We can find the furthest block B_k, that houses boards that are furthest away in number of pieces from our current board among the K_{NB}. So, we know what that k is.
- Thus, we need to now find all the pieces that have longetivity = k, i.e. that piece doesn't change for all of the board states in our k-nearest boards.
- So, barring the positions in the first point - we run longevity for all the other pieces in the current board (as done in Query 1b. If the result = k, store that piece's position and return along with all such pieces at the end.

Open Ended

Some additional queries that can be added:

Earliest Corner

A query that returns the sequence of moves that leads to the earliest corner disc placed among the K_{NB}.

Execution

For every $n > i$ where i is the turn number P is at, consider the n^{th} board state from block B_n and check whether any corner discs have been placed. At the first occurrence where a corner disc is present, return the sequence starting from the i^{th} move of that board till turn n.

Shortest game

Among the K_{NB} boards, return the sequence of moves that lead to the earliest end to a a game.

Execution

For every $n > i$ where i is the turn number P is at, consider the n^{th} board state from block B_n and check whether there are any legal moves. At the first occurrence where there are no more legal moves, return the sequence starting from the i^{th} move of that board.