GamesCrafters 2006

Mancala

Documentation for Developers

Alvin Chyan (dev)
Spencer Ray (dev)
Keith Ho (dev)

Last revised 2006 Dec. 6

Table of Contents

1. Overview

Mancala is a fairly unique game in its board configuration. Unlike other boards with one piece in every position, any number of pieces could be placed in one bin. It also features numerous variants. Some of these variants remain to be implemented. It is currently tierified to solve larger boards though.

Because of the later tierification, the hash functions differ on first glance. The original un-tierified version uses a lot of combinations and mathematics to construct its hash. It is basically the same code as the underlying implementation of `generic_hash`. In the tierified version, `generic_hash` is called instead.

2. Game Board

One of the confusing aspects of the current implementation is the representation of the game board. When playing the game, the spaces on the board are labeled in the following manner:

```
**          [1] [2] [3]
**        [0]         [4]
**          [5] [6] [7]
```

Internally, the integer array is managed like so:

```
**          [1] [2] [3]
**        [0]         [4]
**          [7] [6] [5]
```

The 0 indexed bin is always the left mancala, and the index for the right mancala is determined as the total boardsize/2, which in this case, is 8 divided by 2—4. Finally, the turn is kept track of in an extra index allotted at the end of this integer array. With the boardsize in this example, the turn number 0 or 1 would be stored in the 8th slot of the array.

3. Hash

`array_unhash`

`array_unhash` is the counterintuitive name for the hashing function that returns a `POSITION`. As with `array_hash`, this function does not unhash a position to a board, but does the opposite. Before it can use `generic_hash_hash` on the board, it converts the board (integer array form) to a sequence of x's and o's. X's represent a divider between bins, and each 'o' represents a stone. For example, "ooxoxx" represents a board with 2 pieces in the first bin, 1 in the second, and 0 in the third and fourth bins.

4. Unhash

`array_hash`

It does not hash to a `POSITION` here in normal terms. It first takes a `POSITION` as a parameter and then unhashes that using `generic_hash_unhash`. It then converts that resulting board (o's and x's) to a game board in integer array form as described in the game board section above. The function is so named because of the "hash" to a more easily viewable integer array format.

5. Tierification

The tier number denotes how many pieces are in the left mancala and the right mancala. It is equal to 100*left mancala pieces + right mancala pieces, so the pieces in each mancala are easy to view as the two left digits and two right digits in the tier number.

5.1 InitTierStuff

The bulk of the code is in the initialization of the respective hash windows. The array size for the remainder of the tierified mancala is boardsize – 2 – 1 + numOfPieces – mancL – mancR. The two is to adjust for the removed two outer bins. The one is there for mathematical reasons. In the `generic_hash` representation, the 'o' pieces can be partitioned into 6 bins using 5 dividers, so we don't actually need boardsize – 2 'x's', but boardsize – 2 – 1 x's to adequately divide the pieces. The numOfPieces – mancL – mancR represents the number of remaining pieces in the middle bins, or in other words, not in the mancalas.

5.2 TierChildren

In general, only one player should win pieces in one turn. However, due to the numerous variants, including the winning of all pieces on one's side of the board when the other player is out of pieces, it is possible for multiple players to win pieces in one move. Therefore, a tier's children includes not just the tiers in which one player's mancala increases in pieces, but also those where one player gains pieces, and the other player gets the remaining stones.