**Algorithm**

### Zobrist hashing

"Zobrist hashing is a hash function construction used in computer programs that play abstract board games, such as chess and Go, to implement transposition tables, a special kind of hash table that is indexed by a board position and used to avoid analyzing the same position more than once."

To use Zobrist hashing, we must initialize the hashing table. Basically, we need to generate a random number for each possible state inside a single cell of the board. The game has 20 cells (5 rows x 4 columns), and each cell can be empty or occupied by 4 different types of blocks. In program, we use a three-dimensional array to store the initial random numbers.

1. Create empty state of the board, fill all the border line array elements with -1 and empty cells with 0.

Game board= create 2D array of size (row+2) × (column+2)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 0 | 0 | 0 | 0 | -1 |
| -1 | 0 | 0 | 0 | 0 | -1 |
| -1 | 0 | 0 | 0 | 0 | -1 |
| -1 | 0 | 0 | 0 | 0 | -1 |
| -1 | 0 | 0 | 0 | 0 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 |

1. Create hash table.

constant block indices

SMALL\_BLOCK = 1

VERTICAL\_BLOCK = 2

HORIZONTAL\_BLOCK = 3

BOX\_BLOCK = 4

function init\_zobrist():

# fill a table of random numbers/bitstrings

Zobrist\_hash := a 3D array of size 5×4×4

for i from 1 to 5: # loop over each row of the board

for j from 1 to 4: # loop over each column of the board

for k type from 1 to 4 # loop over each block type

Zobrist\_hash[i][j][k] = random\_bitstring()

|  |  |  |  |
| --- | --- | --- | --- |
| 4 hash values for each block type | 4 hash values for each block type | 4 hash values for each block type | 4 hash values for each block type |
| .  .  . | .  .  . | .  .  . | .  .  . |
| .  .  . | .  .  . | .  .  . | .  .  . |
| .  .  . | .  .  . | .  .  . | .  .  . |
| .  .  . | .  .  . | .  .  . | .  .  . |

1. Add initial Game state

Check for available position

If true

Take\_Position() //add index of block to state board at position

Push positioned block to state.blocks

Else false

Return;

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| -1 | -1 | -1 | -1 | -1 | -1 |
| -1 | 1 | 1 | 2 | 2 | -1 |
| -1 | 1 | 1 | 3 | 4 | -1 |
| -1 | 0 | 0 | 3 | 4 | -1 |
| -1 | 5 | 6 | 6 | 7 | -1 |
| -1 | 5 | 7 | 8 | 9 | -1 |
| -1 | -1 | -1 | -1 | -1 | -1 |

1. Get Zobrist hash for current state

h=0

for i from 1 to 5: # loop over each row of the state

for j from 1 to 4: # loop over each column of the state

type = blocks[i][j].type

h = h XOR Zobrist\_hash[i][j][type]

return h;

1. Store state in Game.states array with hash value for that state
2. Store unique hashkeys in Game.Zhash array
3. Check for Goal state
   1. If Yes

PrintMoves()

* 1. If No
     1. Search for new Game state
     2. Check for all possible direction for each block
        1. If Yes
           1. Update Zobrist hash

1. Update Zobrist Hash
   1. Clear hash for old position
   2. Create hash for new position

h = old\_hash

//clear hash for old position

h = h XOR Zobrist\_hash[old\_row][old\_col][block\_type\_index]

//generate hash for new position

h = h XOR Zobrist\_hash[new\_row][new\_col][block\_type\_index]

1. Check for updated Hash
   1. Is it available in Zhash array
      1. If true then go to step 7
      2. If False then go to step 5