



| Name | Code |
|-------------------------|---------|
| Yara Ismail Fekry | 1501696 |
| Hajar Mohamed Adel Rizk | 1601633 |
| Hadeer Fawzy Ahmed | 1501677 |
| Nouran Khaled | 1601602 |
| Hassan Mostafa Mohamed | 15E0052 |

جامعة عين شمس

1839

كلية الهندسة - جامعة عين شمس

Code Main Functions

- Print game

This function's purpose is to represent the console based UI, and to print the game array in a form that is recognizable to the player.

```
1 def printgame(game):
2     for i in range(12, 6, -1):
3         print(' ', game[i], ' ', end='')
4
5     print('\n')
6     print(game[13], ' ',
7           game[6])
8     print('\n')
9     for i in range(0, 6, 1):
10        print(' ', game[i], ' ', end='')
11    print('\n')
```

```
4      4      4      4      4      4
0                                0
4      4      4      4      4      4 ← Player's Pits
```

- Final

This function's purpose is to check whether or not the game is still going.

It takes the current game array as an input

Returns 1 if the player opposing to the bot finished all the stones in his pits.

Returns 2 if bot finished all the stones in all his 6 pits

Returns 0 if the game is still going.

```
1 def final(game):
2     if sum(game[0:6]) == 0:
3         game[13] += sum(game[7:13])
4         for i in range(14):
5             if (i != 13 and i != 6):
6                 game[i] = 0
7         return 1
8
9     elif sum(game[7:13]) == 0:
10        game[6] += sum(game[0:6])
11        for i in range(14):
12            if (i != 13 and i != 6):
13                game[i] = 0
14        return 2
15    else:
16        return 0
```

- Move Game

```
1 def movegame(borad_mancala, num_pit, stealing):
2     num_seed = 0
3     another_turn = False
```

```

4     num_pit_sent = num_pit
5     if num_pit_sent < 6:
6
7         num_seed = borad_mancala[num_pit]
8         borad_mancala[num_pit] = 0
9         while (num_seed != 0):
10             num_pit += 1
11             num_pit = num_pit % 14
12             if num_pit == 13:
13                 continue
14             borad_mancala[num_pit] += 1
15             num_seed -= 1
16             if (borad_mancala[num_pit] == 1 and borad_mancala[
17                 -num_pit + 12] != 0 and num_pit != 6 and num_seed == 0 and stealing == 0):
18                 borad_mancala[6] += borad_mancala[num_pit] + borad_mancala[-num_pit + 12]
19                 borad_mancala[num_pit] = 0
20                 borad_mancala[-num_pit + 12] = 0
21             if (num_pit == 6 and num_seed == 0):
22                 another_turn = True
23
24     if num_pit_sent > 6:
25         num_seed = borad_mancala[num_pit]
26         borad_mancala[num_pit] = 0
27         while (num_seed != 0):
28             num_pit += 1
29             num_pit = num_pit % 14
30             if num_pit == 6:
31                 continue
32             borad_mancala[num_pit] += 1
33             num_seed -= 1
34             if (borad_mancala[num_pit] == 1 and borad_mancala[
35                 5 - (num_pit - 7)] != 0 and num_pit != 13 and num_seed == 0 and stealing == 0):
36                 borad_mancala[13] += borad_mancala[num_pit] + borad_mancala[5 - (num_pit - 7)]
37                 borad_mancala[num_pit] = 0
38                 borad_mancala[5 - (num_pit - 7)] = 0
39             if (num_pit == 13 and num_seed == 0):
40                 another_turn = True
41
42     return another_turn

```

When changing
player pits.

When changing bot
pits.

This function's purpose is to change the values of the game array following the rules of mancala, it is adaptive to work in both stealing and no stealing case, it takes the current board, the pit chosen, and whether or not stealing will be applied.

Returns a bool to indicate whether or not a player will play again (in case the player's last stone falls into player's home).

- Heuristic

```
1 def heuristic(game):
2     if final(game):
3         if game[13] > game[6]:
4             return 49
5         elif game[13] == game[6]:
6             return 0
7         else:
8             return -49
9     else:
10        return game[13] - game[6]
```

This function is responsible for heuristic value calculation, we chose the heuristic function to be the difference between the number of stones in both players' homes, we chose a maximum value of 49, because the best case (all stones in one home) is 48.

There were other possible functions like looking at the stones in the bot's home and maximizing them but this function neglects the number of stones in the opponents home, another function is to minimizing the opponent's stones, but it neglects the number of stones in the bot's home.

Other possible functions such as aiming for boards that give another turn or aiming for boards that makes bot have maximum stones in his pits(not home).

But after some research and thinking, we decided that the best function is the one we chose.

- AlphaBeta

```
1 def alphabeta(currentgame, depth, alpha, beta, MinorMax, steal):
2     global num_of_nodes
3     if depth == 0 or final(currentgame) != 0:
4         hursvalue = heuristic(currentgame)
5         verbose['NumofLeaf'] += 1
6         verbose['values'].append(hursvalue)
7         return hursvalue, -1
8     if MinorMax:
9         v = -1000
10        move = -1
11        for m in range(7, 13, 1):
12            num_of_nodes +=1
13            if currentgame[m] == 0: continue
14            a = currentgame[:]
15            minormax = movegame(a, m, steal)
16            newv, _ = alphabeta(a, depth - 1, alpha, beta, minormax, steal)
17            if v < newv:
18                move = m
```

Supports Difficulty Levels

Changes depending on the turn/level we are now in.

When evaluating a Max node.

```

19         v = newv
20         alpha = max(alpha, v)
21         if alpha >= beta:
22             verbose['numofCut'] += 1
23             verbose['levels'].append(depth)
24             break
25         return v, move
26     else:
27         v = 1000
28         move = -1
29         for n in range(0, 6, 1):
30             if currentgame[n] == 0: continue
31             a = currentgame[:]
32             minormax = movegame(a, n, steal);
33             num_of_nodes +=1
34             newv, _ = alphabeta(a, depth - 1, alpha, beta, not minormax, steal)
35             if v > newv:
36                 move = n
37                 v = newv
38             beta = min(beta, v)
39             if alpha >= beta:
40                 verbose['numofCut'] += 1
41                 verbose['levels'].append(depth)
42                 break
43         return v, move

```

When evaluating a
Min node.

This function is the main decision-making algorithm, it is a recursive function that supports iterative depth allowing for multiple difficulty levels, supporting stealing and verbose mode, saving the info in a verbose.txt file.

• Verbose functions

```

1 def storeVerbose(i):
2     try:
3         global num_of_nodes
4         verbose['AverageBF'] = (verbose['NumofLeaf'] +
num_of_nodes) / (num_of_nodes+1)
5         verbose_file = open('verbose.txt', 'a')
6         verbose_file.write('Turn' + str(i) + ': ' + str(verbose)
+str(num_of_nodes)+ '\n')
7         verbose_file.close()
8         verbose['AverageBF'] = 0
9         verbose['NumofLeaf'] = 0
10        verbose['values'].clear()
11        verbose['numofCut'] = 0
12        verbose['levels'].clear()

```

```

• 13         num_of_nodes = 0
• 14     except:
• 15         print("Unable to write to file")
• 16
• 17
• 18 def delete_verbose():
• 19     # open file
• 20     f = open("verbose.txt", "w+")
• 21
• 22     # absolute file positioning
• 23     f.seek(0)
• 24
• 25     # to erase all data
• 26     f.truncate()
• 27     f.close

```

These two functions are responsible for all data recording and documentation for every turn, all data is recorded in a dictionary and saved in a txt file.

• Main Code

```

1 loop1=1
2 Newgame = [0 for i in range(14)]
3 for i in range(0, 6, 1): Newgame[i] = 4
4 for i in range(7, 13, 1): Newgame[i] = 4
5 delete_verbose()
6

```

Flag

Game array initialization and clearing previously recorded data.

```

7 while True:
8     try:
9         resume = input("New game or Resume ? 0:Newgame, 1:Resume\n")
10        if resume.isdigit():
11            resume = int(resume)
12        else:
13            raise ValueError()
14        if resume==0:
15            break
16        if resume==1:
17            try:
18                if(path.exists("save.txt")):
19                    filesize = os.path.getsize("save.txt")
20                    if filesize == 0:
21                        print("the file is empty")
22                        resume = input("Enter the value 0 for a new game \n")
23                    else:
24                        raise ValueError()
25            except ValueError:
26                print("There is no saved game, you will now play a new game")

```

This loop is responsible for all input possibilities after the bots asks the player whether or not he wants to resume.

```

27         resume=0
28         loop1=0
29         raise ValueError()
30     except ValueError:
31         if loop1==0:
32             break
33         print("Input must be 0 or 1.\n")
34
35 if resume==1:
36     if (path.exists("save.txt")):
37         with open('save.txt','rb') as f:
38             n,steal,diff=pickle.load(f)
39             Newgame = n
40             turn = 0
41             print('If the result is 0:With Stealing 1:Without Stealing\n',steal)
42             print('If the difficulty is 0:easy 1:medium 2:hard\n',diff)
43     else:
44         print("file does not exist")
45
46 elif resume == 0:
47
48     while True:
49
50         try:
51             turn = input("Who do you want to start? 0=YOU, 1=BOT\n")
52             if turn.isdigit():
53                 turn = int(turn)
54             else:
55                 raise ValueError()
56             if turn == 0 or turn == 1:
57                 break
58             raise ValueError()
59         except ValueError:
60             print("Input must be 0 or 1.\n")
61
62     while True:
63         try:
64             steal = input("Do you want stealing? 0=YES, 1=NO\n")
65             if steal.isdigit():
66                 steal = int(steal)
67             else:
68                 raise ValueError()
69             if steal == 0 or steal == 1:
70                 break
71             raise ValueError()
72         except ValueError:
73             print("Input must be 0 or 1.\n")
74

```

Data extraction from pre saved file

Taking input from user in case user chooses to play a new game.

```

75     while True:
76         try:
77             diff = input("Which difficulty level do you want? 0=easy, 1=medium,
2=hard\n")
78             if diff.isdigit():
79                 diff = int(diff)
80                 if diff == 0:
81                     depth = 4
82                 elif diff == 1:
83                     depth = 8
84                 elif diff == 2:
85                     depth = 12
86                 verbose['Maxdepth'] = depth
87             else:
88                 raise ValueError()
89             if diff == 0 or diff == 1 or diff == 2:
90                 break
91             raise ValueError()
92         except ValueError:
93             print("Input must be 0 or 1 or 2.\n")
94
95 i = 1
96 count = 2
97 printgame(Newgame)
98 while (True):
99     if final(Newgame):
100         whowon = Newgame[13] - Newgame[6]
101         break
102
103     while (turn == 1):
104         if final(Newgame):
105             whowon = Newgame[13] - Newgame[6]
106             break
107         print("MY TURN ")
108         _, k = alphabeta(Newgame, depth, -1000, 1000, True, steal)
109         print('I choose to move pit number ', k)
110         t = movegame(Newgame, k, steal)
111         printgame(Newgame)
112         storeVerbose(i)
113         i += 1
114         if (not t):
115             turn = 0
116         break

```

Main loop

Bot's turn


```

117
118     while (turn == 0):
119         if final(Newgame):
120             break
121         while (turn == 0):
122             try:
123                 save = input("Do you want to save the game? 0:NO 1:YES.\n ")
124                 if save.isdigit():
125                     save = int(save)
126                 else:
127                     raise ValueError()
128                 if save == 0 or save == 1:
129                     break
130                 raise ValueError()
131             except ValueError:
132                 print("Input must be 0 or 1.\n")
133         if save == 1:
134             with open("save.txt", "wb") as f:
135                 pickle.dump([Newgame, steal, diff], f)
136             sys.exit()
137         else:
138             while (turn == 0):
139                 try:
140                     h = input("YOUR TURN\nYou choose pit number:")
141                     if h.isdigit():
142                         h = int(h)
143                     else:
144                         raise ValueError()
145                     if h == 0 or h == 1 or h == 2 or h == 3 or h == 4 or h == 5:
146                         break
147                     raise ValueError()
148                 except ValueError:
149                     print("You must choose one of your own pits")
150             if h > 5 or Newgame[h] == 0:
151                 print('you can\'t play')
152                 break
153             t = movegame(Newgame, h, steal)
154             printgame(Newgame)
155             if (not t):
156                 turn = 1
157                 break
158
159 printgame(Newgame)
160 if (whowon > 0):
161     print('I WIN. HA HA')
162 elif (whowon < 0):
163     print('You won..')
164 elif (whowon == 0):
165     print('Its a Tie')

```

Player's Turn

Ending phrases

User guide and snapshots

Snapshot

```
In [37]: runfile('D:/College/4th Computer/Term 2/Artificial Intelligence/Project/HagoorFinal.py', wdir='D:/College/4th Computer/Term 2/Artificial Intelligence/Project')
```

```
New game or Resume ? 0:Newgame, 1:Resume  
5  
Input must be 0 or 1.
```

Wrong input

```
New game or Resume ? 0:Newgame, 1:Resume  
1  
There is no saved game, you will now play a new game
```

Wrong input because there's no previously saved game

```
Who do you want to start? 0=YOU, 1=BOT  
0
```

Normal Dialogue

```
Do you want stealing? 0=YES, 1=NO  
0
```

```
Which difficulty level do you want? 0=easy, 1=medium, 2=hard  
0
```

```
4      4      4      4      4      4  
0                                     0
```

```
4      4      4      4      4      4
```

```
Do you want to save the game? 0:NO 1:YES.  
0
```

```
YOUR TURN  
You choose pit number:|
```

```
YOUR TURN  
You choose pit number:0  
4      4      4      4      4      4  
0                                     0  
0      5      5      5      5      4  
MY TURN  
I choose to move pit number 11  
5      0      4      4      4      4  
1                                     0  
1      6      5      5      5      4  
Do you want to save the game? 0:NO 1:YES.  
|
```

```
MY TURN
I choose to move pit number 11
  5      0      4      4      4      4

1                                     0

  1      6      5      5      5      4

Do you want to save the game? 0:NO 1:YES.
2
Input must be 0 or 1.

Do you want to save the game? 0:NO 1:YES.
1
```

```
New game or Resume ? 0:Newgame, 1:Resume
5
Input must be 0 or 1.

New game or Resume ? 0:Newgame, 1:Resume
1
If the result is 0:With Stealing 1:Without Stealing
0
If the difficulty is 0:easy 1:medium 2:hard
2
  6      0      0      0      4      4

9                                     0

  5      0      5      5      5      5

Do you want to save the game? 0:NO 1:YES.
0

YOUR TURN
```

```

26                                     3

0      0      0      0      0      7

MY TURN
I choose to move pit number 7
0      3      3      4      2      0

26                                     3

0      0      0      0      0      7

Do you want to save the game? 0:NO 1:YES.
0

YOUR TURN
You choose pit number:5
1      4      4      5      3      1

26                                     4

0      0      0      0      0      0

0      0      0      0      0      0

44                                     4

0      0      0      0      0      0

I WIN. HA HA

```

MY TURN

I choose to move pit number 7

7 2 1 7 7 0

3 2

2 6 5 5 0 1

MY TURN

I choose to move pit number 11

8 0 1 7 7 0

4 2

2 6 5 5 0 1

MY TURN

I choose to move pit number 10

8 0 0 7 7 0

11 2

2 0 5 5 0 1

Work division

| | Alpha Beta Algorithm | Movement Algorithm | Main Loop | Heuristic Calculation | Final function | Printing function | Report | Verbose | Save and load |
|-------------------|----------------------------|-----------------------|--------------|--------------------------|-------------------|----------------------|--------|---------|---------------------|
| Hajr Rizk | ✓ | | ✓ | | | ✓ | | ✓ | |
| Yara Ismail | ✓ | | ✓ | | | ✓ | ✓ | | |
| Hadeer Fawzy | | ✓ | ✓ | | | ✓ | | | ✓ |
| Nouran Khaled | | ✓ | ✓ | | | ✓ | | | ✓ |
| Hassan Mostafa | | | ✓ | ✓ | ✓ | | ✓ | | |

Youtube link

<https://youtu.be/xS8gG63y7m8>

Github link

<https://github.com/YaraFekri/MancalaAI>