IBA

ALGORITHM REPORT

Prepared by:
Ashal Ibrahim 26977

Content

Introduction

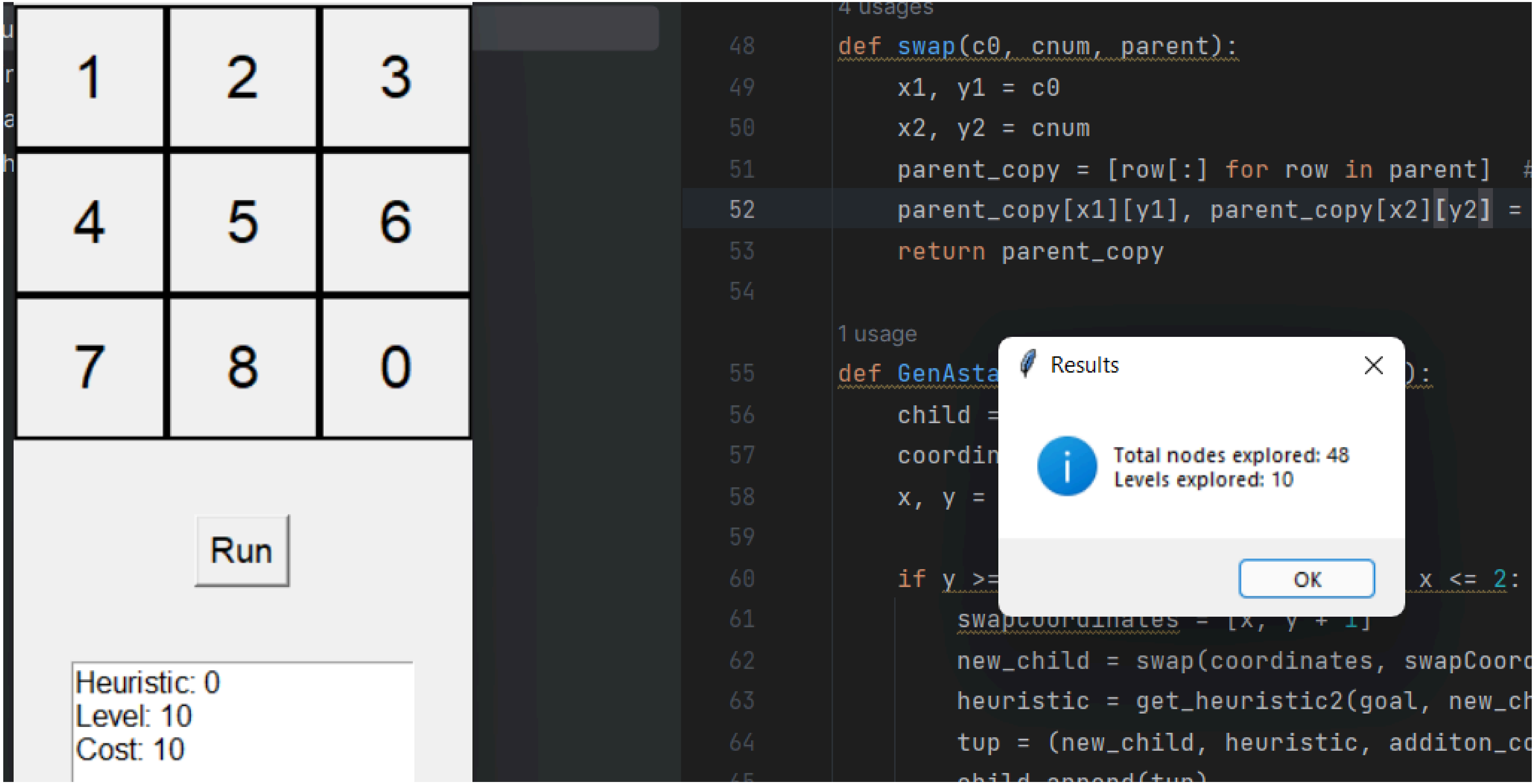
The presentation is about a puzzle problem known as 8 piece puzzle:
There are 9 tiles placed in a 3 by 3 matrix. The tiles are not in their original positions, each tile is a number from 1 to 8 and the 9th tile is a blank tile which is going to be used to swap positions of each tile adjacent to it either vertically or horizontally. The algorithm will be chosen by the user, it will run and give the correct solution(shortest)

Algorithms

A* Algorithm

It is an informed search technique. There were 2 types of heuristics used:

1) Sum of wrong tiles (get_heuristic1())
2) Manhattan distance of each tile (get_heuristic2())



```
1 2 3
4 5 6
7 8 0
```

```
Heuristic: 0
Level: 10
Cost: 10
```

```
4 usages
def swap(c0, cnum, parent):
    x1, y1 = c0
    x2, y2 = cnum
    parent_copy = [row[:] for row in parent]  #
    parent_copy[x1][y1], parent_copy[x2][y2] =
    return parent_copy

1 usage
def GenAsta                                 ):
    child =
    coordin
    x, y =

    if y >=                                 x <= 2:
        swapCoordinates = [x, y + 1]
        new_child = swap(coordinates, swapCoord
        heuristic = get_heuristic2(goal, new_ch
        tup = (new_child, heuristic, additon_co
        child.append(tup)
```

```
Total nodes explored: 48
Levels explored: 10
```

Analysis

For the same input: we tried running it through both the different heuristics
Surprisingly using both we got the same levels and nodes explored

No such analysis possible between the 2 heuristics, although with a different input heuristic 2 performed better than heuristic 1

Genetic Algorithm

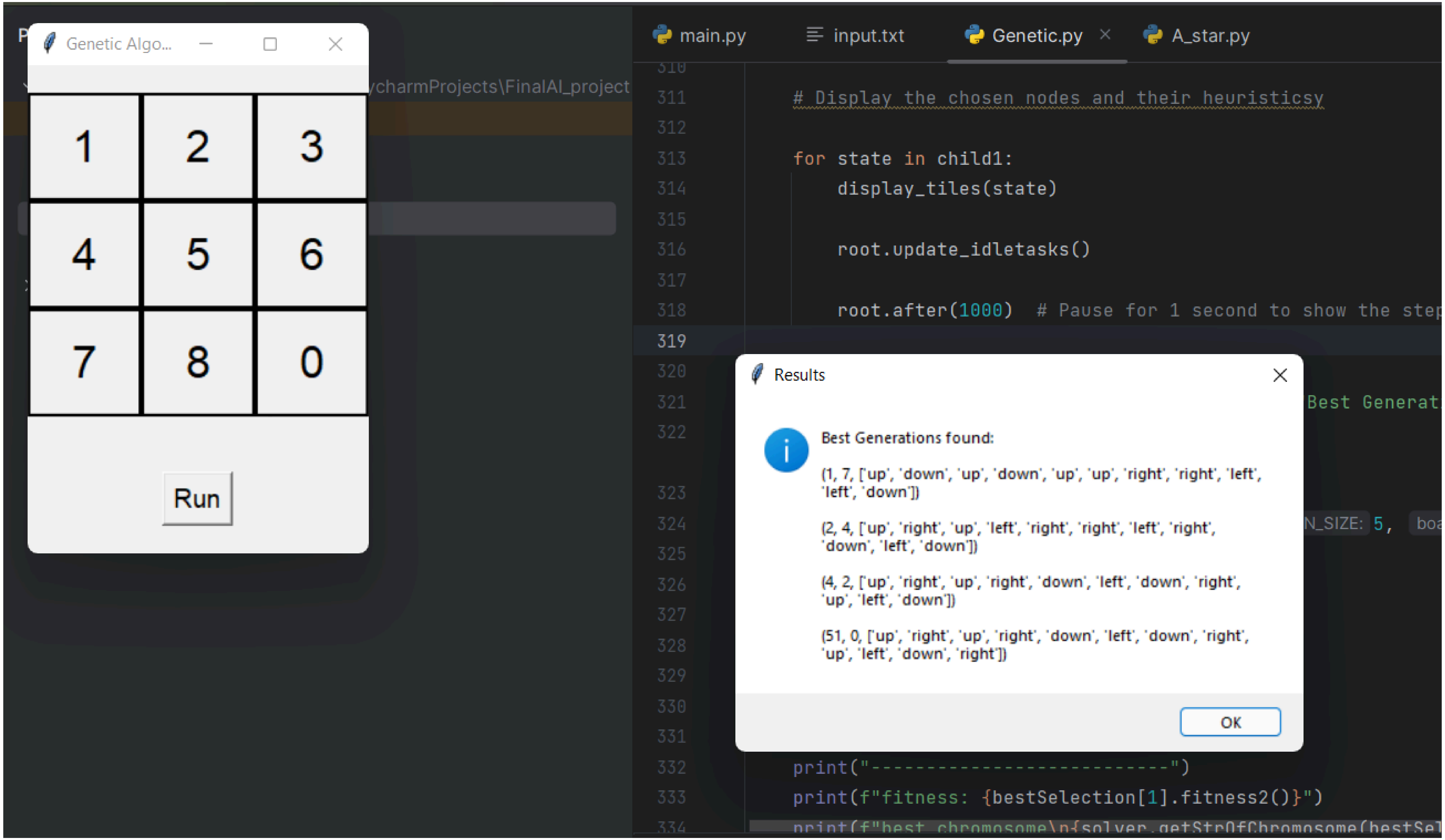
It is an evolutionary algorithm technique. There were 2 types of heuristics used:

1) Sum of wrong tiles (fitness2())
2) Manhattan distance of each tile (fitness())

Representation:

list of moves: [up,down,left,right,right...] this is one chromosome

this was then shuffled and then new chromosomes were created till they reached the initial population size



```
    # Display the chosen nodes and their heuristicsy

    for state in child1:
        display_tiles(state)

        root.update_idletasks()

        root.after(1000)  # Pause for 1 second to show the step
```

Results

Best Generations found:

(1, 7, ['up', 'down', 'up', 'down', 'up', 'up', 'right', 'right', 'left', 'left', 'down'])

(2, 4, ['up', 'right', 'up', 'left', 'right', 'right', 'left', 'right', 'down', 'left', 'down'])

(4, 2, ['up', 'right', 'up', 'right', 'down', 'left', 'down', 'right', 'up', 'left', 'down'])

(51, 0, ['up', 'right', 'up', 'right', 'down', 'left', 'down', 'right', 'up', 'left', 'down', 'right'])

OK

```
    print("--------------------------")
    print(f"fitness: {bestSelection[1].fitness2()}")
    print(f"best chromosome\n{solver.getStrOfChromosome(bestSel
```

Same generation different chromosome length,
Same generation and chromosome length, different heuristic used

- **MAX GENERATIONS (1ST COLUMN)**
- **MAX CHROMOSOME LENGTH (2ND COLUMN)**
- **NUMBER OF SELECTED CHROMOSOMES (3RD COLUMN)**
- **FITNESS FUNCTION USED (1. WRONG TILES OR 2. MANHATTAN) (4TH COLUMN)**
- **AVERAGE CHROMOSOME IN WHICH GENERATION (5TH COLUMN)**
- **AVERAGE CHROMOSOME LENGTH (6TH COLUMN**

100	20	2	2	27	19
100	11	2	2	50	14
100	11	2	1	60	15

For each row the algorithm was run 5 times and the average was taken for average best generation and average chromosome length in each best generation

Analysis

As the initial chromosome length in which the representation was was an array or list of moves to be applied on the blank tile until it reaches the goal state e.g [up, down, left, left, right, up] then the fitness was calculated of the final state after the set of moves had been applied to check if its zero or close to the goal state or not

1. **More initial chromosome length:**

- **lesser generations required to reach goal state**
- **lesser the moves required to reach goal state**
- **lesser the randomness and fluctuation in moves**

2.Manhattan is better than sum of wrong tiles

Initial chromosome length reduced

- MAX GENERATIONS (1ST COLUMN)
- MAX CHROMOSOME LENGTH (2ND COLUMN)
- NUMBER OF SELECTED CHROMOSOMES (3RD COLUMN)
- FITNESS FUNCTION USED (1. WRONG TILES OR 2. MANHATTAN) (4TH COLUMN)
- AVERAGE CHROMOSOME IN WHICH GENERATION (5TH COLUMN)
- AVERAGE CHROMOSOME LENGTH (6TH COLUMN

100	5	2	2	51	10
200	5	2	2	127	14
50	5	2	2	-	-

For each row the algorithm was run 5 times and the average was taken for average best generation and average chromosome length in each best generation

Analysis

For initial chromosome length=5

Generations=100

if the goal was being found within 100 generations then the average generations were 50 and the avg chromosome length was 10, which was the most accurate answer that could be achieved without any fluctuation

Generations=200

if the goal was being found within 100 to 200 generations then the average generations were 120 and the avg chromosome length was 14, with some fluctuations between the moves

Generations<=50

No goal state was found within 50 generations for the following parameters

Selection of chromosomes changed

- MAX GENERATIONS (1ST COLUMN)
- MAX CHROMOSOME LENGTH (2ND COLUMN)
- NUMBER OF SELECTED CHROMOSOMES (3RD COLUMN)
- FITNESS FUNCTION USED (1. WRONG TILES OR 2. MANHATTAN) (4TH COLUMN)
- AVERAGE CHROMOSOME IN WHICH GENERATION (5TH COLUMN)
- AVERAGE CHROMOSOME LENGTH (6TH COLUMN

200	20	2	2	13.4	15.2
200	20	5	2	3.6	17.6
200	20	10	2	3	16.4

For each row the algorithm was run 5 times and the average was taken for average best generation and average chromosome length in each best generation

Analysis

For initial chromosome length=20

Selected Chromosome = 2

The avg generation in which the goal state was found in 13.4 almost 13 and the average move list length was 15, sometimes wasnt found at all

Selected Chromosome = 5

The avg generation in which the goal state was found in 3.6 almost 4 and the average move list length was 17.6

Selected Chromosome = 10

The avg generation in which the goal state was found in 3 and the average move list length was 16.4

As selected chromosome increased, the generations in which the answer was found decreased, as more selected chromosomes meant higher chances of finding the move list closest to the goal state early

A* vs Genetic

A* was much faster than genetic, we get this idea by comparing average move list obtained

in A* for the same input we got 10 levels explored
in genetic with different parameters our answers varied, such as:
the most efficient move list we got was when initial chromosomes were 5 which was equivalent to A* , and the avg move list was 16 when initial chromosomes were 20

A* is consistent with the results, whereas genetic algorithm is not consistent but rather quite random
as the move list varies with each run

A star vs Genetic
To prove randomness

Heuristic 1 chosen

Trial	A star	Genetic
1st	Nodes Explored: 48 Moves: 10	51 Gens, 16 moves
2nd	Nodes Explored: 48 Moves: 10	3 Gens, 10 moves