COMP 424 Assignment 1

By Ziqiao Ringgold Lin, 260561361

Question 1

a)

Part 1 & 2: (Sequence Puzzle from initial to the goal state)

```
1 4 2
5 3 0

1 4 2
5 0 3

1 0 2
5 4 3

0 1 2
5 4 3
Since it is unit step cost, BFS = UCS, cost =3
```

For both of BFS and UCS, the total explored states number is 9, and cost is 3.

Part 3 DFS: (Sequence Puzzle from initial to the goal state) From **Top to bot, left to right**

```
1 4 2
5 3 0

1 4 0
5 3 2

1 0 4
5 3 2

0 1 4
5 3 2

5 1 4
0 3 2

5 1 4
3 0 2

5 0 4
3 1 2

5 4 0
3 1 2

5 4 2
3 1 0

5 4 2
3 0 1

5 4 2
0 3 1
```

```
0 4 2
5 3 1

4 0 2
5 3 1

4 2 0
5 3 1

4 2 1
5 3 0

4 2 1
5 0 3

4 0 1
5 2 3

4 1 0
5 2 3

4 1 3
5 2 0

4 1 3
5 0 2

4 0 3
5 1 2

4 3 0
5 1 2

4 3 2
5 1 0
```

```
4 3 2
5 0 1

4 3 2
0 5 1

0 3 2
4 5 1

3 0 2
4 5 1

3 2 0
4 5 1

3 2 1
4 5 0

3 2 1
4 0 5

3 0 1
4 2 5

3 1 0
4 2 5

3 1 5
4 2 0

3 1 5
4 0 2

3 0 5
4 1 2
```

```
0 3 5
4 1 2

4 3 5
0 1 2

4 3 5
1 0 2

4 3 5
1 2 0

4 3 0
1 2 5

4 0 3
1 2 5

4 2 3
1 0 5

4 2 3
0 1 5

0 2 3
4 1 5

2 0 3
4 1 5

2 1 3
4 0 5

2 1 3
0 4 5
```

```
0 1 3
2 4 5

1 0 3
2 4 5

1 3 0
2 4 5

1 3 5
2 4 0

1 3 5
2 0 4

1 3 5
0 2 4

0 3 5
1 2 4

3 0 5
1 2 4

3 2 5
1 0 4

3 2 5
0 1 4

0 2 5
3 1 4

2 0 5
3 1 4
```

```
2 1 5
3 0 4

2 1 5
0 3 4

0 1 5
2 3 4

1 0 5
2 3 4

1 5 0
2 3 4

1 5 4
2 3 0

1 5 4
2 0 3

1 5 4
0 2 3

0 5 4
1 2 3

5 0 4
1 2 3

5 2 4
1 0 3

5 2 4
0 1 3
```

```
0 2 4
5 1 3

2 0 4
5 1 3

2 1 4
5 0 3

2 1 4
5 3 0

2 1 0
5 3 4

2 0 1
5 3 4

0 2 1
5 3 4

5 2 1
0 3 4

5 2 1
3 0 4

5 0 1
3 2 4

0 5 1
3 2 4

3 5 1
0 2 4
```

Figure 1. The first half of sequences of DFS

```
3 5 1
2 0 4

3 5 1
2 4 0

3 5 0
2 4 1

3 0 5
2 4 1

0 3 5
2 4 1

2 3 5
0 4 1

2 3 5
4 0 1

2 0 5
4 3 1

0 2 5
4 3 1

4 2 5
0 3 1

4 2 5
3 0 1

4 2 5
3 1 0
```

```
4 2 0
3 1 5

4 0 2
3 1 5

4 1 2
3 0 5

4 1 2
0 3 5

0 1 2
4 3 5

1 0 2
4 3 5

1 2 0
4 3 5

1 2 5
4 3 0

1 2 5
4 0 3

1 0 5
4 2 3

0 1 5
4 2 3

4 1 5
0 2 3
```

```
4 1 5
2 0 3

4 0 5
2 1 3

0 4 5
2 1 3

2 4 5
0 1 3

2 4 5
1 0 3

2 4 5
1 3 0

2 4 0
1 3 5

2 0 4
1 3 5

0 2 4
1 3 5

1 2 4
0 3 5

1 2 4
3 0 5

1 0 4
3 2 5
```

```
0 1 4
3 2 5

3 1 4
0 2 5

3 1 4
2 0 5

3 0 4
2 1 5

0 3 4
2 1 5

2 3 4
0 1 5

2 3 4
1 0 5

2 3 4
1 5 0

2 3 0
1 5 4

2 0 3
1 5 4

0 2 3
1 5 4

1 2 3
0 5 4
```

```
1 2 3
5 0 4

1 0 3
5 2 4

0 1 3
5 2 4

5 1 3
0 2 4

5 1 3
2 0 4

5 0 3
2 1 4

5 3 0
2 1 4

5 3 4
2 1 0

5 3 4
2 0 1

5 3 4
0 2 1

0 3 4
5 2 1

3 0 4
5 2 1
```

```
3 2 4
5 0 1

3 2 4
5 1 0

3 2 0
5 1 4

3 0 2
5 1 4

3 1 2
5 0 4

3 1 2
5 4 0

3 1 0
5 4 2

3 0 1
5 4 2

0 3 1
5 4 2

5 3 1
0 4 2

5 3 1
4 0 2

5 3 1
4 2 0
```

```
5 3 0
4 2 1

5 0 3
4 2 1

5 2 3
4 0 1

5 2 3
4 1 0

5 2 0
4 1 3

5 0 2
4 1 3

5 1 2
4 0 3

5 1 2
0 4 3

0 1 2
5 4 3
```

Figure 2. The second half sequences of DFS

And the total explored states number is 190.

Part 4 ID: (Sequence Puzzle from initial to the goal state)

```
1 4 2
5 3 0

1 4 2
5 0 3

1 0 2
5 4 3

0 1 2
5 4 3
The ID stopped and found goal state at depth 3. The cost is: 3
```

Figure 3. IDS result after stopped and finished at depth of 3

b)

<u>Yes, Manhattan distance has the h value is still admissible.</u>

Justify: As the transition cost is equal to the number of the piece that is moved. The actual minimum cost for a travel becomes the following equation while n is the number of piles we are going to slide:

$\boldsymbol{New\ Min\ Cost} = \sum_{k=0}^{n} \boldsymbol{k}$ and the previous cost $\boldsymbol{h1} = \boldsymbol{Old\ Min\ Cost} = \boldsymbol{n}$

New Min Cost >= Old Min Cost (***<u>h1</u>***), which means the Manhattan distance is still admissible.

c)

Make the New Cost as shown below as the new heuristic value ***<u>h2</u>***:

$$\boldsymbol{h2} = \boldsymbol{New\ Min\ Cost} = \sum_{k=0}^{n} \boldsymbol{k}$$

Since the New Min Cost (***<u>h2</u>***) <= The Actual Min Cost, this heuristic value dominates h1 (proven above), and is admissible.

d)

No, if the setup is not like the initial state and goal state shown on this question, but just need us to swap the top and bottom pile to get the goal state, then the actual cost will be 0.5, but the heuristic value will be 1 (bigger than 0.5). Thus the Manhattan distance value as heuristic value at this case will not be admissible.

Question 2

a) **Example of ID much worse than DFS**
If the state space is a simple linked list with size n, this will happen.
1) With DFS, the search will be simply go through n of linked nodes with a max time complicity of only O(n).
2) With IDS, the total search cost will be 1 + 2 + ... + n = O(n^2) since it is a iterative type.

Thus in this case IDS is much worse than DFS

b) **BFS is a special case of UCS**
Uniform Cost Search reduces to Breadth First Search when the step cost is a unit cost (just like the question 1).

c) **DFS is a special case of best-first tree search**
When the evaluation function f(n) = -depth(n), the search behavior will be DFS.

d) **UCS is a special case of A***
A* reduces to UCS when the heuristic function h(n) = 0 for all n.

Question 3

a) **The following tables shows the result of hill-climbing of X_0 from 0 to 10**

x0	StepSize	max(X, Y)	Steps
00	.01	(1.74, +0.396)	175
00	.02	(1.74, +0.396)	88
00	.03	(1.74, +0.396)	59
00	.04	(1.72, +0.396)	44
00	.05	(1.75, +0.396)	36
00	.06	(1.74, +0.396)	30
00	.07	(1.75, +0.396)	26
00	.08	(1.76, +0.396)	23
00	.09	(1.71, +0.396)	20
00	.10	(1.70, +0.395)	18

x0	StepSize	max(X, Y)	Steps
01	.01	(1.74, +0.396)	75
01	.02	(1.74, +0.396)	38
01	.03	(1.75, +0.396)	26
01	.04	(1.72, +0.396)	19
01	.05	(1.75, +0.396)	16
01	.06	(1.72, +0.396)	13
01	.07	(1.77, +0.395)	12
01	.08	(1.72, +0.396)	10
01	.09	(1.72, +0.396)	9
01	.10	(1.70, +0.395)	8

x0	StepSize	max(X, Y)	Steps
02	.01	(1.74, +0.396)	27
02	.02	(1.74, +0.396)	14
02	.03	(1.73, +0.396)	10
02	.04	(1.72, +0.396)	8
02	.05	(1.75, +0.396)	6
02	.06	(1.76, +0.396)	5
02	.07	(1.72, +0.396)	5
02	.08	(1.76, +0.396)	4
02	.09	(1.73, +0.396)	4
02	.10	(1.70, +0.395)	4

x0	StepSize	max(X, Y)	Steps
03	.01	(1.74, +0.396)	127
03	.02	(1.74, +0.396)	64
03	.03	(1.74, +0.396)	43
03	.04	(1.72, +0.396)	33
03	.05	(1.75, +0.396)	26
03	.06	(1.74, +0.396)	22
03	.07	(1.74, +0.396)	19
03	.08	(1.72, +0.396)	17
03	.09	(1.74, +0.396)	15
03	.10	(1.70, +0.395)	14

x0	StepSize	max(X, Y)	Steps
04	.01	(3.96, +0.334)	5
04	.02	(3.96, +0.334)	3
04	.03	(3.97, +0.334)	2
04	.04	(3.96, +0.334)	2
04	.05	(3.95, +0.334)	2
04	.06	(3.94, +0.333)	2
04	.07	(3.93, +0.332)	2
04	.08	(3.92, +0.330)	2
04	.09	(4.00, +0.330)	1
04	.10	(4.00, +0.330)	1

x0	StepSize	max(X, Y)	Steps
05	.01	(5.32, +0.310)	33
05	.02	(5.32, +0.310)	17
05	.03	(5.33, +0.310)	12
05	.04	(5.32, +0.310)	9
05	.05	(5.30, +0.310)	7
05	.06	(5.30, +0.310)	6
05	.07	(5.35, +0.305)	6
05	.08	(5.32, +0.310)	5
05	.09	(5.36, +0.302)	5
05	.10	(5.30, +0.310)	4

x0	StepSize	max(X, Y)	Steps
06	.01	(6.39, +0.296)	40
06	.02	(6.38, +0.296)	20
06	.03	(6.39, +0.296)	14
06	.04	(6.40, +0.295)	11
06	.05	(6.40, +0.295)	9
06	.06	(6.36, +0.291)	7
06	.07	(6.42, +0.291)	7
06	.08	(6.40, +0.295)	6
06	.09	(6.36, +0.291)	5
06	.10	(6.40, +0.295)	5

x0	StepSize	max(X, Y)	Steps
07	.01	(7.31, +0.286)	32
07	.02	(7.30, +0.285)	16
07	.03	(7.30, +0.285)	11
07	.04	(7.32, +0.285)	9
07	.05	(7.30, +0.285)	7
07	.06	(7.30, +0.285)	6
07	.07	(7.28, +0.280)	5
07	.08	(7.32, +0.285)	5
07	.09	(7.27, +0.275)	4
07	.10	(7.30, +0.285)	4

x0	StepSize	max(X, Y)	Steps
08	.01	(8.12, +0.278)	13
08	.02	(8.12, +0.278)	7
08	.03	(8.12, +0.278)	5
08	.04	(8.12, +0.278)	4
08	.05	(8.10, +0.273)	3
08	.06	(8.12, +0.278)	3
08	.07	(8.14, +0.275)	3
08	.08	(8.16, +0.265)	3
08	.09	(8.09, +0.269)	2
08	.10	(8.10, +0.273)	2

x0	StepSize	max(X, Y)	Steps
09	.01	(8.86, +0.271)	15
09	.02	(8.86, +0.271)	8
09	.03	(8.85, +0.270)	6
09	.04	(8.88, +0.268)	4
09	.05	(8.85, +0.270)	4
09	.06	(8.88, +0.268)	3
09	.07	(8.86, +0.271)	3
09	.08	(8.84, +0.266)	3
09	.09	(8.82, +0.253)	3
09	.10	(8.90, +0.256)	2

x0	StepSize	max(X, Y)	Steps
10	.01	(10.00, -0.069)	1
10	.02	(10.00, -0.069)	1
10	.03	(10.00, -0.069)	1
10	.04	(10.00, -0.069)	1
10	.05	(10.00, -0.069)	1
10	.06	(10.00, -0.069)	1
10	.07	(10.00, -0.069)	1
10	.08	(10.00, -0.069)	1
10	.09	(10.00, -0.069)	1
10	.10	(10.00, -0.069)	1

As shown on the 11 tables attached, the **left most side states initial X_0 value**, and the following are the **increasing step size**, and corresponding local maximum (x, y), then at the right most is the **steps(cost)** needed to find it.

b) The following table is the simulated annealing

Simulated Annealing
T= 4000.0, alpha = 0.95
Total steps: 252, step size: 0.1

x0	max(x, y)
00	(1.85, 0.396)
01	(1.93, 0.396)
02	(1.83, 0.396)
03	(3.95, 0.334)
04	(3.93, 0.334)
05	(3.91, 0.334)
06	(6.43, 0.310)
07	(8.13, 0.286)
08	(8.14, 0.278)
09	(9.50, 0.271)
10	(8.90, 0.271)

All of the parameters and results are shown on the left. My overall parameter defining logic is to “try as many as possible”. From the **part a)** analysis, I found out that the general shape of the graph is quite bumpy and not linear based. Thus, I chose the step size to be 0.1 (the maximum step size I can choose) so that the stepping action can make the cursor jump out of the local maximum as best as it can. Since the difference between E(i) and E(i*) is usually small, I picked a large T = 4000 to balance the p value, and make alpha rather big to cover more area.

Question 4

a)

Variables: k rooks (can be filled from [1][1] to [n][n])

Domains: {nxn spaces | each space may and may not have one rook}

Constraints:

For each rook with coordinates [x][y] ($x \in [1,n]$, $y \in [1,n]$)

from [1][y] to [n][y], there is no another rook

from [x][1] to [x][n], there is no another rook

b)

numberToAssign = 3

assignedCount = 0

initially: all of the spaces are not filled

For space [1][1] to [3][3]

if (the potential placement satisfy that from [1][y] to [3][y] there is no another rook,

and from [x][1] to [x][3] there is no another rook) then{

1. assign the space [x][y] with one rook k_i (marked as filled, 1<=i<=3)
2. assignedCount++
3. delete [x][y] from the available space

if (assignedCount == numberToAssign){

break and finish;

}

}

End for

(next page continue for part c)

c)

numberToAssign = 3

assignedCount = 0

initially: all of the spaces are not filled

For space [1][1] to [3][3]

if (the potential placement satisfy that from [1][y] to [3][y] there is no another rook,

and from [x][1] to [x][3] there is no another rook) then{

1. assign the space [x][y] with one rook k_i (marked as filled, 1<=i<=3)
2. assignedCount++
3. delete [1][y] , [2][y], [3][y] from the available space
4. delete [x][1] , [x][2], [x][3] from the available space

if (assignedCount == numberToAssign){

break and finish;

}

}

End for

END OF THE ASSIGNMENT 1