Artificial Intelligence

ECS629U-759P

Coursework1

Spyridon Roumpis

181004877

Question 1: It's on the tip of my tongue! (Genetic Algorithm)

(a) Following the UCS algorithm, manually execute it for three iterations for this problem.

```
frontier []

explored ['london']

frontier [(110, ['london', 'birmingham']), (116, ['london', 'bristol']), (54, ['london', 'cambridge']), (161, ['london', 'car
diff']), (302, ['london', 'carlisle']), (71, ['london', 'dover']), (172, ['london', 'exeter']), (396, ['london', 'glasgow']),
(172, ['london', 'hull']), (198, ['london', 'leeds']), (198, ['london', 'liverpool']), (57, ['london', 'oxford'])]

explored ['london', 'brighton']

frontier [(110, ['london', 'birmingham']), (116, ['london', 'bristol']), (161, ['london', 'cardiff']), (302, ['london', 'carl
isle']), (71, ['london', 'dover']), (172, ['london', 'exeter']), (396, ['london', 'glasgow']), (172, ['london', 'hull']), (19
8, ['london', 'leeds']), (198, ['london', 'liverpool']), (57, ['london', 'oxford']), (353, ['london', 'brighton', 'aberystwyt
h']), (282, ['london', 'brighton', 'nottingham']), (381, ['london', 'brighton', 'penzance']), (153, ['london', 'brighton', 'p
ortsmouth']), (320, ['london', 'brighton', 'sheffield']), (340, ['london', 'brighton', 'swansea']), (354, ['london', 'brighto
n', 'york'])]

explored ['london', 'brighton', 'cambridge']
```

(b) Implement the UCS, run it with the provided data

frontier [(396, ['london', 'glasgow']), (565, ['london', 'hull', 'edinburgh']), (732, ['london', 'cambridge', 'sheffield', 'aberdeen'])]
explored ['london', 'brighton', 'cambridge', 'oxford', 'dover', 'birmingham', 'bristol', 'portsmouth', 'cardiff', 'exeter', 'hull', 'leeds', 'liverpool', 'nottingham', 'sheffield', 'manchester', 'york', 'swansea', 'aberystwyth', 'carlisle', 'penzance', 'newcastle']
path_till_now ['london', 'glasgow'] cost 396

frontier [(565, ['london', 'hull', 'edinburgh']), (732, ['london', 'cambridge', 'sheffield', 'aberdeen'])]
explored ['london', 'brighton', 'cambridge', 'oxford', 'dover', 'birmingham', 'bristol', 'portsmouth', 'cardiff', 'exeter', 'hull', 'leeds', 'liverpool', 'nottingham', 'sheffield', 'manchester', 'york', 'swansea', 'aberystwyth', 'carlisle', 'penzance', 'newcastle', 'glasgow']
path_till_now ['london', 'hull', 'edinburgh'] cost 393

frontier [(508, ['london', 'cambridge', 'sheffield', 'aberdeen'])]
explored ['london', 'brighton', 'cambridge', 'oxford', 'dover', 'birmingham', 'bristol', 'portsmouth', 'cardiff', 'exeter', 'hull', 'leeds', 'liverpool', 'nottingham', 'sheffield', 'manchester', 'york', 'swansea', 'aberystwyth', 'carlisle', 'penzance', 'newcastle', 'glasgow', 'edinburgh']
path_till_now ['london', 'cambridge', 'sheffield', 'aberdeen'] cost 508

```
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Final Path ['london', 'cambridge', 'sheffield', 'aberdeen'] with cost 508

All explored nodes ['london', 'brighton', 'cambridge', 'oxford', 'dover', 'birmingham', 'bristol', 'portsmouth', 'cardiff', 'ex
eter', 'hull', 'leeds', 'liverpool', 'nottingham', 'sheffield', 'manchester', 'york', 'swansea', 'aberystwyth', 'carlisle', 'pe
nzance', 'newcastle', 'glasgow', 'edinburgh', 'aberdeen']

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

There is an issue with the code, bug, it doesn't run optimally for this part, but due to lack of time, it could not be resolved. I hope it will not affect the over mark. The last part,c, was not tried to be achieved, again because of limited time.

Question 2: It's on the tip of my tongue! (Genetic Algorithm)

(a) What did you get for the two passwords?

In the first case, where the index is 0 the password is PASSWORD: 9_FR33D0M8 while in the second case where the index is 1 the password is PASSWORD: WE1L_D0N3_

(b) Explain briefly how specifically you chose to do the state encoding, selection, crossover, and mutation.

- Initial population, different numbers of population initialization tried.
- Fitness function was given
- Selection, for the selection process we selected the fittest individuals and let them pass their genes to the next generation. Two pairs of individuals (parents) are selected based on their fitness scores. The selection_type is the 'elite' type.
- Crossover, the crossover point was the first 5 digits for the first parent while for the second parent the last 5.
- Mutation, a mutation rate of 20% was selected. The mutation type was 'bit_flip.

c) Report on the number of reproductions you had to do to converge to the password?

Parameters		
Index/Password	**1st password**	**2nd password**
Population	100	100
Elite Size	2	2
Mutation Rate	0.2	0.2
Iterations	**Generations**	
1	4328	2021
2	2634	1673
3	4978	805
4	4230	1061
5	1547	605
6	1202	690
7	6137	447
8	4033	2927
9	1135	3701
10	616	1120
Mean	**3084**	**1505**

(c) Explore the effect of at least one hyper-parameter on the performance of your algorithm.

Experiments were conducted for 6 different values of the population(10,50,100,500,1000,5000). For each of those, we tried two different values for the elite size and the mutation rate.

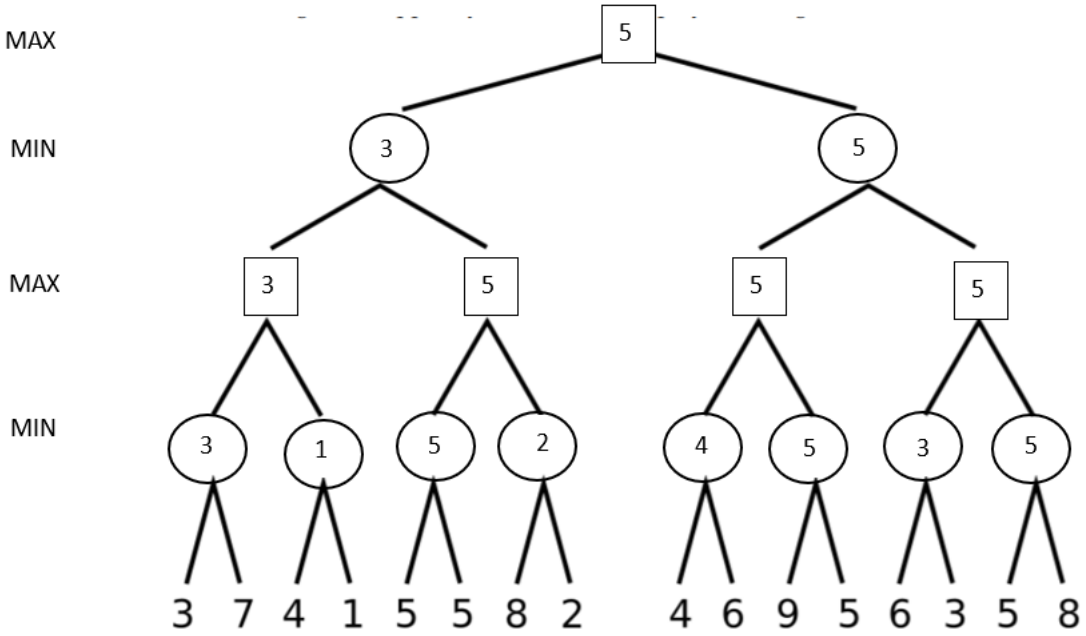
Population	10				50				100			
Elite Size	2		4		2		4		2		4	
Mutation Rate	0.2	0.5	0.2	0.5	0.2	0.5	0.2	0.5	0.2	0.5	0.2	0.5
Iterations	Generations				Generations				Generations			
1	26917	100001	8024	100001	2134	100001	769	100001	518	100001	653	100001
2	35678	100001	20473	100001	5245	100001	362	100001	3985	100001	352	100001
3	23150	100001	5245	100001	9454	100001	951	100001	1378	100001	380	100001
4	46340	100001	17229	100001	1134	100001	1024	100001	971	100001	956	100001
5	40305	100001	9517	100001	4995	100001	690	100001	624	100001	358	100001
Mean	34478.0	100001.0	12097.6	100001.0	4592.4	100001.0	759.2	100001.0	1495.2	100001.0	539.8	100001.0

Population	500				1000				5000			
Elite Size	2		4		2		4		2		4	
Mutation Rate	0.2	0.5	0.2	0.5	0.2	0.5	0.2	0.5	0.2	0.5	0.2	0.5
Iterations	Generations				Generations				Generations			
1	27127	100001	338	12945	41956	100001	934	100001	67980	100001	100001	100001
2	28621	100001	1580	100001	100001	45227	3198	54920	100001	100001	58698	100001
3	24928	100001	392	100001	100001	100001	736	100001	100001	100001	100001	100001
4	25940	100001	1344	100001	25310	100001	927	100001	83610	100001	100001	100001
5	30305	100001	1526	100001	100001	100001	1078	67140	97520	100001	100001	100001
Mean	27384.2	100001.0	1036.0	82589.8	73453.8	89046.2	1374.6	84412.6	89822.4	100001.0	91740.4	100001.0

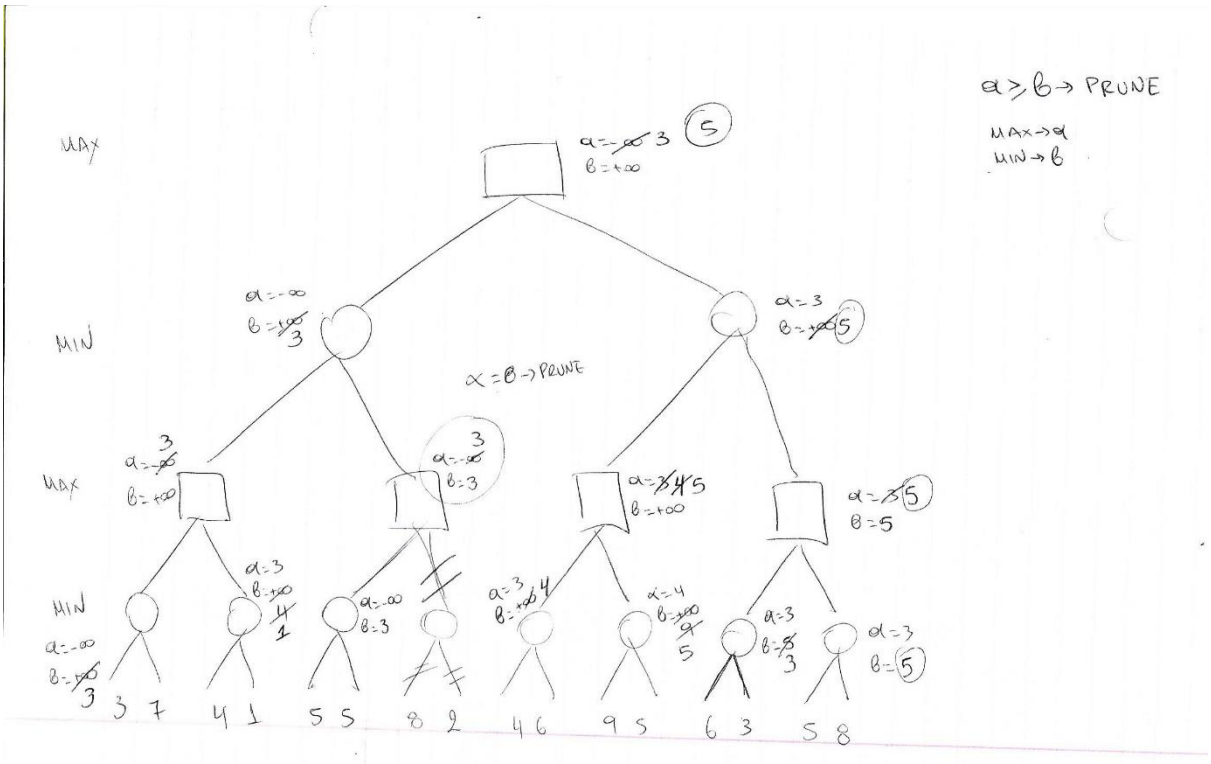
Overall it can be seen the smallest the population and for elite size=2 the fastest the algorithm is. Also for a big number of the population, the algorithm performs better with big elite size. For population=5000 the algorithm performed badly. Finally, the mutation rate worked better for the 0.2 value.

Question 3: Let's play a game! (Adversarial search)

(a) Play optimally (MINIMAX algorithm)



(b) Play optimally, quicker thinking! (α-β pruning)



The (3,2) node was pruned. Beta = 3 = alpha, the condition to prune a node.

(c) Suppose the entry fee to play the game is x units (for either player)

c-(1)

In the MINIMAX algorithm, the maximizer works to get the highest score, while the minimizer tries to get the lowest score by trying to counter moves. From the tree, it can be seen that the best score for both the MAX and the MIN player is 5.
We know that if the MIN player plays optimally it will get a score of 5, so that means that the MAX player should pay at most 5. By paying 5 it will not earn any money, by paying x money where x=[0,4) it will earn 5-x.

c-(2) It would make no difference at all to be either the MAX or the MIN player. Both of them would get the same score of 5. The maximum score of the MAX player would be 5 same as the score of the MIN player.

References:

https://towardsdatascience.com/ai-search-algorithms-implementations-2334bfc59bf5