Moises Alexander Gama Espinosa A00817566
Carlos Roberto Cueto Zumaya A01209474

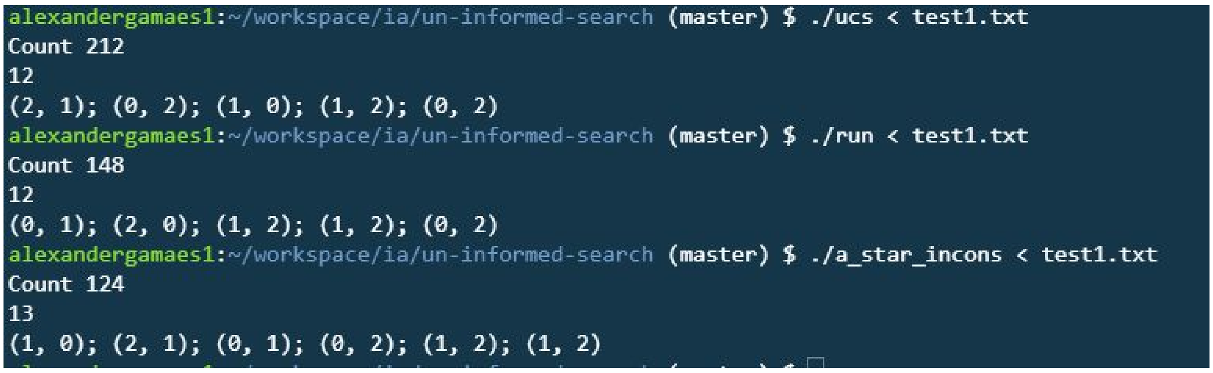
(Un)informed Search Lab

- **Which heuristics did you use for the A* algorithm?**

  For the UCS algorithm we used a value of 0, whereas for the consistent heuristic we calculate the misplaced blocks and for the inconsistent heuristic we multiply the misplaced blocks by 5 because it surpass the consistent heuristic.

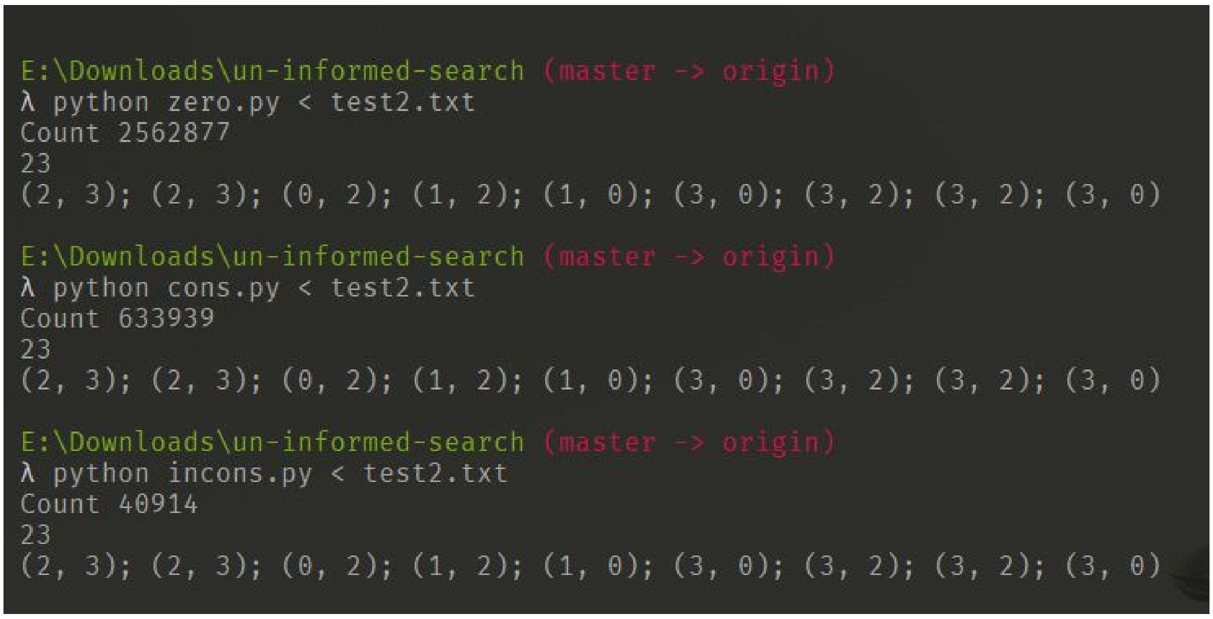
- Test your program with a couple of different problems. Increase the size of the problem to test the limits of your program. Make a table comparing **how many nodes are searched** to find the answer for each problem. For this table, you should compare a number of different problems (at least 3) to avoid a statistical bias. Which of the three algorithms (UCS, A *with consistent and and A* with an inconsistent heuristic) searches the least nodes and which one take the most?

Test1



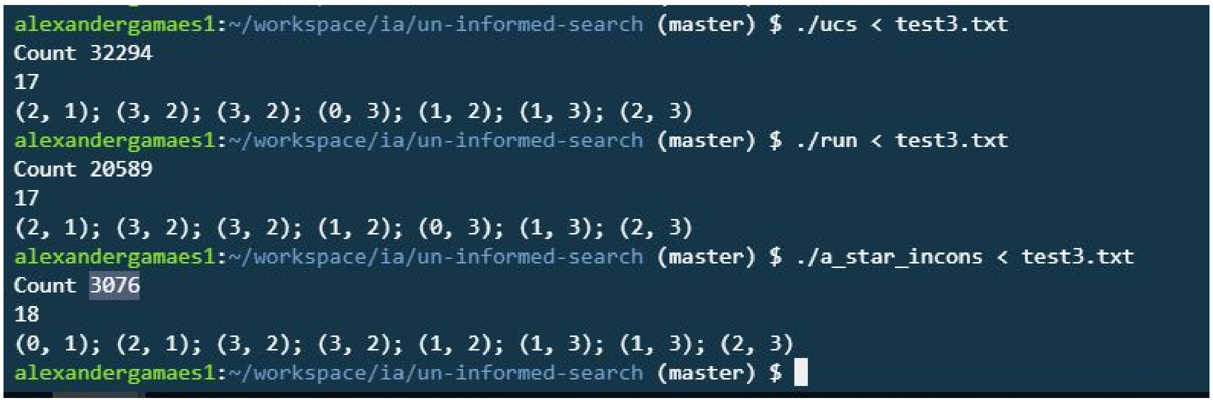
```
alexandergamaes1:~/workspace/ia/un-informed-search (master) $ ./ucs < test1.txt
Count 212
12
(2, 1); (0, 2); (1, 0); (1, 2); (0, 2)
alexandergamaes1:~/workspace/ia/un-informed-search (master) $ ./run < test1.txt
Count 148
12
(0, 1); (2, 0); (1, 2); (1, 2); (0, 2)
alexandergamaes1:~/workspace/ia/un-informed-search (master) $ ./a_star_incons < test1.txt
Count 124
13
(1, 0); (2, 1); (0, 1); (0, 2); (1, 2); (1, 2)
```

Test2



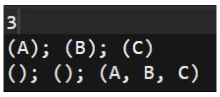
```
E:\Downloads\un-informed-search (master -> origin)
λ python zero.py < test2.txt
Count 2562877
23
(2, 3); (2, 3); (0, 2); (1, 2); (1, 0); (3, 0); (3, 2); (3, 2); (3, 0)

E:\Downloads\un-informed-search (master -> origin)
λ python cons.py < test2.txt
Count 633939
23
(2, 3); (2, 3); (0, 2); (1, 2); (1, 0); (3, 0); (3, 2); (3, 2); (3, 0)

E:\Downloads\un-informed-search (master -> origin)
λ python incons.py < test2.txt
Count 40914
23
(2, 3); (2, 3); (0, 2); (1, 2); (1, 0); (3, 0); (3, 2); (3, 2); (3, 0)
```

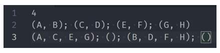
Test3

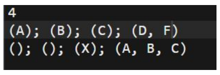


```
alexandergamaes1:~/workspace/ia/un-informed-search (master) $ ./ucs < test3.txt
Count 32294
17
(2, 1); (3, 2); (3, 2); (0, 3); (1, 2); (1, 3); (2, 3)
alexandergamaes1:~/workspace/ia/un-informed-search (master) $ ./run < test3.txt
Count 20589
17
(2, 1); (3, 2); (3, 2); (1, 2); (0, 3); (1, 3); (2, 3)
alexandergamaes1:~/workspace/ia/un-informed-search (master) $ ./a_star_incons < test3.txt
Count 3076
18
(0, 1); (2, 1); (3, 2); (3, 2); (1, 2); (1, 3); (1, 3); (2, 3)
alexandergamaes1:~/workspace/ia/un-informed-search (master) $
```

Test	UCS	A* (consistent heuristic)	A* (inconsistent heuristic)

1.  `3`
`(A); (B); (C)`
`(); (); (A, B, C)`
	212	148	124

2.  `4`
`(A, B); (C, D); (E, F); (G, H)`
`(A, C, E, G); (); (B, D, F, H); ()`
	2, 562, 877	633, 939	40, 914

3  `4`
`(A); (B); (C); (D, F)`
`(); (); (X); (A, B, C)`
	32, 294	20, 589	3, 076

- **Why does this happen?**

  The USC is the worst because the queue is always considering the same cost, this lead to a lot of nodes even though the solution is at the beginning of the graph, whereas the A* algorithm with inconsistent heuristic is always trying to achieve the quickest path sometimes sacrificing optimality resulting in maybe a non optimal solution, but, with an consistent heuristic, the node is expand it choice the lowest cost possible resulting in a optimal solution sacrificing time and memory.

- **Which algorithms are optimal? Why?**

  The UCS and the A* algorithm with a consistent heuristic are optimal. The UCS because its definition guarantees that if it finds the goal, it will have the optimal cost, because the next states are stored in a priority queue.

  The A* is optimal because it also uses priority queue, but only when the heuristic is consistent. This is because this ensure that the estimation is valid: that we are not overestimating in any point (admisible), and that an action will not have a bigger effect in the heuristic that the one that is supposed to be (consistent).

- **In your opinion, what are the benefits of simpler algorithms versus more complex ones?**

  The advantage of a simpler algorithm versus a complex one is that they are more generic, so you don’t need to rewrite so much code when you are trying to resolve a different problem. So, the maintainability is better. The disadvantage could be that the computation time or memory space could be not the best one. For example, there is an algorithm that resolves the hanoi’s tower much faster that the A* algorithm, but it won't be easy to make it to resolve a labyrinth.

Bibliography

Peter Norvig, Stuart J. Russell. (1994). Artificial Intelligence: A Modern Approach. Berkeley: Prentice Hall.