| \* Grupo 5  \* Luis Gabriel Román Santillan  \* Mateo González de Miguel  \* Víctor Tello Carrascal  8,5  Ej 1. Ok  Ej 2. Bien el puzle de 16.  Ej 3. Bien las pruebas de misioneros y puzle de 8. Faltan las pruebas del puzle de 16 del ej3.  ¿por qué la llamada breadth\_first\_tree\_search(Ocho\_Puzzle((2, 8, 3, 1, 6, 4, 7, 0, 5))).solution() se queda en un bucle infinito? Las conclusiones del ej3 regular. “Como podemos ver, la búsqueda en anchura del árbol del puzle de 8 se queda buscando eternamente ya que no hay control de repetidos” que no termine no tiene que ver con el control de repetidos (ya que la busqueda en anchura es completa tanto en treesearch como en graph search) Tiene que ver con que el estado inicial no tiene solución (check solvability)  ¿ uno sería el de anchura por ser un algoritmo más veloz, pero si carecemos de recursos sería más conveniente el de profundidad.?  El de anchura NO es más veloz.  Ej4. Bien las heurísticas. Completo.  Ej5.  En el enunciado se pide que hagáis pruebas también con los algoritmos que correspondan a búsqueda\_coste\_uniforme y busqueda\_primero\_el\_mejor . Solo habéis hecho búsqueda\_a\_estrella Las pruebas están en el ejercicio 6.  “Teóricamente al ser la heurística Manhattan más informada los tiempos deberían ser mejores, pero en este caso la heurística linear da mejores tiempos, ya que el número de nodos que genera es menor.” Esto ocurre en el puzle de 16 solo si el estado inicial está casi colocado.. si hacéis pruebas con el puzle descolocado se ven las diferencias.  EJ6. Bien las pruebas y los comentarios en general. |
| --- |