|  |
| --- |
| TRABAJO FINAL  JUEGO DE MESA QUORIDOR |
| Team Members  Sinuiri Mozombite Giancarlo u201619022 |



**Universidad Peruana de Ciencias Aplicadas**

Ingeniería de Software

Ciclo 2020-02

## CONTENIDO

[CONTENIDO 2](#_Toc57320101)

[INTRODUCCIÓN 3](#_Toc57320102)

[ESTADO DEL ARTE 4](#_Toc57320103)

[METODOLOGÍA 5](#_Toc57320104)

[Espacio de búsqueda 5](#_Toc57320105)

[EXPERIMENTOS 7](#_Toc57320106)

[Experimentos 7](#_Toc57320107)

[RESULTADOS 8](#_Toc57320108)

[CONCLUSIONES 9](#_Toc57320109)

[BIBLIOGRAFÍA 10](#_Toc57320110)

CHAPTER

Ciclo 2020-02 | CC42 XXXX

## INTRODUCCIÓN

1

Hoy en día, dada la coyuntura de la pandemia covid-19 se ha reforzado el entretenimiento de juegos de mesa usando diferentes dispositivos, pues, la mayoría de personas no pueden salir de casa por el toque de queda impuesto por el país. A partir de lo anterior, en el presente proyecto se va a desarrollar un software de juego de mesa denominado 'Quoridor', que tiene como finalidad llegar a la base enemiga. Para lograr el objetivo de desarrollar el software se va a aprovechar los conocimientos adquiridos en la asignatura de complejidad algorítmica, al discernir de una serie de algoritmos cuáles serán los necesarios para llegar a la solución. Primero, se va a analizar diferentes proyectos de investigación relacionados al juego. A continuación, los métodos a usar. Finalmente, se mostrará los experimentos y resultados obtenidos.

CHAPTER

## ESTADO DEL ARTE

2

Quoridor es uno de los juegos de mesa con el espacio de búsqueda más amplias, pues, el tablero es mediana y las posibilidades de movimientos del jugador es grande en cada jugada, en estos últimos años aún se sigue haciendo estudios de complejidad del mismo y busca encontrar el desarrollo óptimo para su resolución.

*"Cuando esto es demasiado dificil de calcular, una cota superior a menudo se puede calcular mediante la inclusion de posiciones ilegales o posiciones que unnca puede surgir en el curso de un juego". (Montoya, 2019, p.1)*

CHAPTER

## METODOLOGÍA

3

### Espacio de búsqueda

#### Estado inicial

Para esta primera entrega el estado inicial será de dos jugadores ubicadas en el centro lateral izquierdo y el otro a conveniencia, además, se colocará 25 barreras (se puede definir el número de barreras en el código) de forma aleatorio en el tablero; tal como se muestra en la Figura 2. El objetivo de cada jugador es llegar a la base lateral del rival. El tablero será representado por una lista de adyacencia, además, se definir el punto de partida del jugador y los estados objetivos.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 9 | 18 | 27 | 36 | 45 | 54 | 63 | 72 |
| 1 | 10 | 19 | 28 | 37 | 46 | 55 | 64 | 73 |
| 2 | 11 | 20 | 29 | 38 | 47 | 56 | 65 | 74 |
| 3 | 12 | 21 | 30 | 39 | 48 | 57 | 66 | 75 |
| 4 | 13 | 22 | 31 | 40 | 49 | 58 | 67 | 76 |
| 5 | 14 | 23 | 32 | 41 | 50 | 59 | 68 | 77 |
| 6 | 15 | 24 | 33 | 42 | 51 | 60 | 69 | 78 |
| 7 | 16 | 25 | 34 | 43 | 52 | 61 | 70 | 79 |
| 8 | 17 | 26 | 35 | 44 | 53 | 62 | 71 | 80 |

goals = [72,73,74,75,76,77,78,79,80]

inicio = 20

G = [

[(1, 1), (9, 1)],

[(0, 1), (2, 1), (10, 1)],

[(1, 1), (3, 1)],

[(2, 1), (4, 1)],

[(3, 1), (5, 1)],

[(4, 1), (6, 1)],

[(5, 1), (7, 1), (15, 1)],

[(6, 1), (8, 1)],

[(7, 1)],

[(0, 1), (10, 1), (18, 1)],

[(1, 1), (9, 1), (11, 1)],

[(10, 1)],

[(21, 1)],

[(14, 1), (22, 1)],

[(13, 1), (23, 1)],

[(6, 1), (16, 1)],

[(15, 1), (17, 1)],

[(16, 1), (26, 1)],

[(9, 1), (19, 1)],

[(18,1), (20, 1)],

[(19, 1)],

[(12, 1)],

[(13, 1), (31, 1)],

[(14, 1), (32, 1)],

[(25, 1)],

[(24, 1), (26, 1)],

[(17, 1), (25, 1), (35, 1)],

[(28, 1), (36, 1)],

[(27, 1), (29, 1), (37, 1)],

[(28, 1), (38, 1)],

[(31, 1)],

[(22, 1)],

[(23, 1), (33, 1), (41, 1)],

[(32, 1), (42, 1)],

[(35, 1), (43, 1)],

[(26, 1), (34, 1), (44, 1)],

[(27, 1), (45, 1)],

[(28, 1), (38, 1), (46, 1)],

[(29, 1), (37, 1)],

[],

[(41, 1), (49, 1)],

[(32, 1), (40, 1), (42, 1), (50, 1)],

[(33, 1), (41, 1)],

[(34, 1), (44, 1)],

[(35, 1), (43, 1), (53, 1)],

[(36, 1)],

[(37, 1), (47, 1)],

[(46, 1), (48, 1), (56, 1)],

[(47, 1), (57, 1)],

[(40, 1), (58, 1)],

[(41, 1), (51, 1), (59, 1)],

[(50, 1), (52, 1), (60, 1)],

[(51, 1), (53, 1), (61, 1)],

[(44, 1), (52, 1), (62, 1)],

[(55, 1), (63, 1)],

[(54, 1), (56, 1), (64, 1)],

[(47, 1), (55, 1), (57, 1), (65, 1)],

[(48, 1), (56, 1), (58, 1), (66, 1)],

[(49, 1), (57, 1), (67, 1)],

[(50, 1), (60, 1), (68, 1)],

[(51, 1), (59, 1), (61, 1), (69, 1)],

[(52, 1), (60, 1), (62, 1)],

[(53, 1), (61, 1)],

[(54, 1), (64, 1)],

[(55, 1), (63, 1), (65, 1)],

[(56, 1), (64, 1), (66, 1), (74, 1)],

[(57, 1), (65, 1), (67, 1), (75, 1)],

[(58, 1), (66, 1), (68, 1)],

[(59, 1), (67, 1), (69, 1)],

[(60, 1), (68, 1), (70, 1), (78, 1)],

[(69, 1), (79, 1)],

[(80, 1)],

[(73, 1)],

[(72, 1), (74, 1)],

[(65, 1), (73, 1), (75, 1)],

[(66, 1), (74, 1), (76, 1)],

[(75, 1), (77, 1)],

[(76, 1), (78, 1)],

[(69, 1), (77, 1), (79, 1)],

[(70, 1), (78, 1)],

[(71, 1)]

]

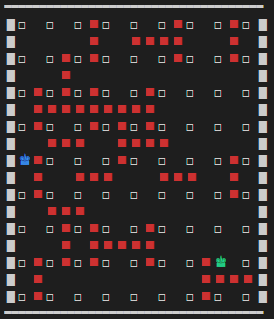
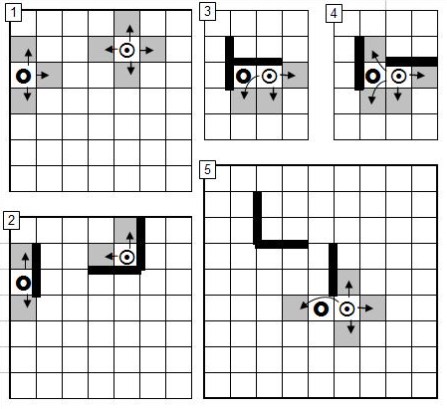


Figura 2: Tablero Quoridor con dos jugadores y 6 barreras colocadas de forma aleatoria. X e Y jugador 1 y jugador 2 respectivamente.

#### Reglas/Operadores/Acciones

Movimientos permitidos.

#### Estado objetivo

Dado que en esta primera entrega se va a analizar el recorrido de un jugador usando algunos algoritmos, el objetivo será de 9 estados posibles (Figura 3); por ejemplo, en la figura 3 se muestra los objetivos del jugador lateral izquierdo.

goals = [72,73,74,75,76,77,78,79,80]

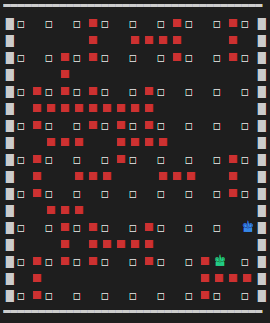


Figura 3: 9 posibles objetivos del jugador 1.

CHAPTER

## EXPERIMENTOS

4

Para llevar a cabo a los experimentos se ha procedido a ejecutar los algoritmos en 10 pruebas en la siguiente tabla se muestra experimentos.

|  |  |  |  |
| --- | --- | --- | --- |
| Pruebas | Tiempo segundos | | |
| UCS | Dijktra |  |
| 1 | 0.000999928 | 0.03299975 |  |
| 2 | 0.001000643 | 0.0319972 |  |
| 3 | 0.000998974 | 0.02299929 |  |
| 4 | 0.000967979 | 0.02899694 |  |
| 5 | 0.001000881 | 0.03200006 |  |
| 6 | 0.001025438 | 0.02698827 |  |
| 7 | 0.001000643 | 0.03099918 |  |
| 8 | 0.000996828 | 0.03401709 |  |
| 9 | 0.000972748 | 0.03400064 |  |
| 10 | 0.0010252 | 0.05603218 |  |

CHAPTER

## RESULTADOS

5

En esta grafica se puede observar la gráfica pruebas x tiempo en donde el algoritmo UCS (búsqueda costo uniforma) resuelve mucho más rápido el objetivo del jugador al llegar a la base enemiga del tablero. Siendo el color verde el algoritmo de Dijktra y el marron UCS.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | |  | |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

CHAPTER

## CONCLUSIONES

5

Se puede concluir que existen algoritmos idóneos para resolver problemas en específicos siendo uno de ellos el óptimo para resolver el juego de quoridor. Esto es muy útil como preámbulo para el aprendizaje de inteligencia artificial, además, se ha podido observar que aún falta otorgarle toma de decisiones al jugador en tiempo real.

## BIBLIOGRAFÍA

* Montoya Suárez Sebastián. (2019). LA COMPLEJIDAD DEL QUORIDOR. Recuperado de https://es.scribd.com/document/463995071/la-complejidad-de-quoridor