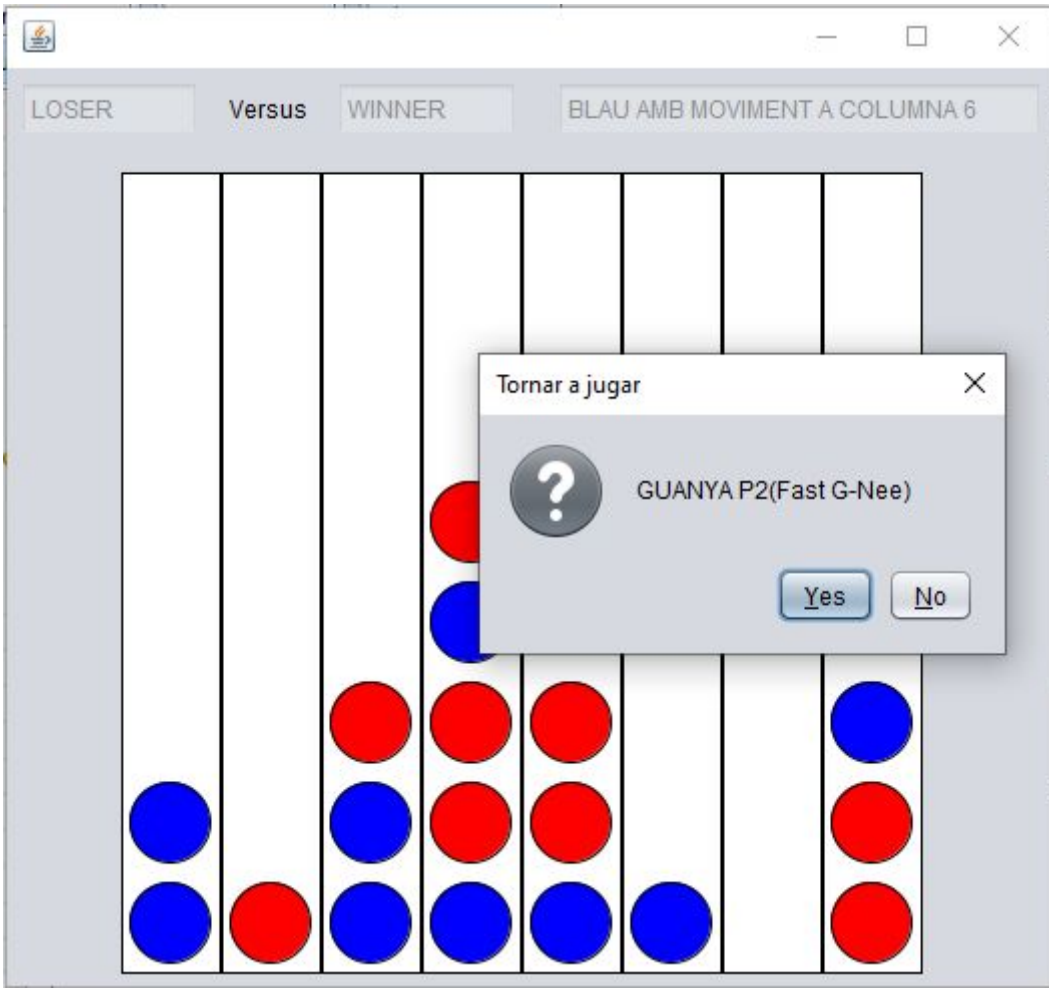
Connecta 4



Introducción

Esta actividad se trata de crear un jugador que gane al jugador aleatorio y profe utilizando el algoritmo de MiniMax.
Lo primero que hicimos es revisar los ficheros que nos dieron para trabajar y ver como funcionaba. Después comenzamos a pensar en cómo calcular la heurística.

Para esta actividad hemos utilizado el algoritmo de MiniMax con poda alfa-beta.

▪Explicación de la Heurística diseñada y detalles de la implementación.

Lo primero que hacemos es calcular la heurística para nuestro jugador y después el de nuestro rival. Restamos el valor de la heurística del rival a la nuestra y lo multiplicamos por la diferencia en la profundidad final y la actual. De esta manera tendrá más importancia los valores en los nodos más cercanos a la raíz y por lo tanto, eso significa que faltan menos turnos para llegar .

Para poder calcular la heurística, revisamos una por una todas las fichas que son del mismo tipo en el tablero y calculamos para cada una de ellas la heurística. Entonces una vez calculada nos quedamos con la que tenga el valor máximo de entre ellas.

Por otro lado, para calcular la heurística de una casilla, lo que hacemos es quedarnos con el máximo número de fichas que tiene adyacentes que tiene esta casilla (este número máximo estará entre 0 y 4), entonces a este numero maximo le damos un valor exponencial (elevando al cuadrado).

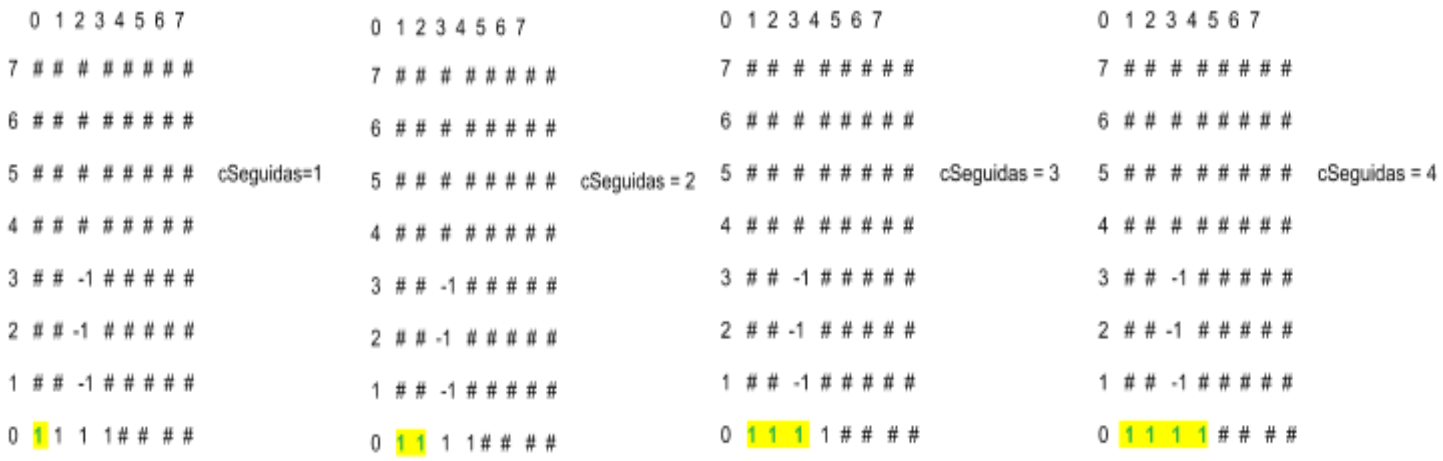
Para finalizar, no tenemos en cuenta las casillas adyacentes si están bloqueadas (hay una ficha del rival en esa casilla) o no podemos ganar en esa dirección, es decir tiene menos de 4 fichas seguidas.

Ejemplo de cómo revisa las casillas adyacentes:

Ya sea para la horizontal, vertical o las diagonales lo que hacemos es un bucle para cada una y revisamos si las casillas son del mismo tipo.
Por ejemplo en el caso de la diagonal:

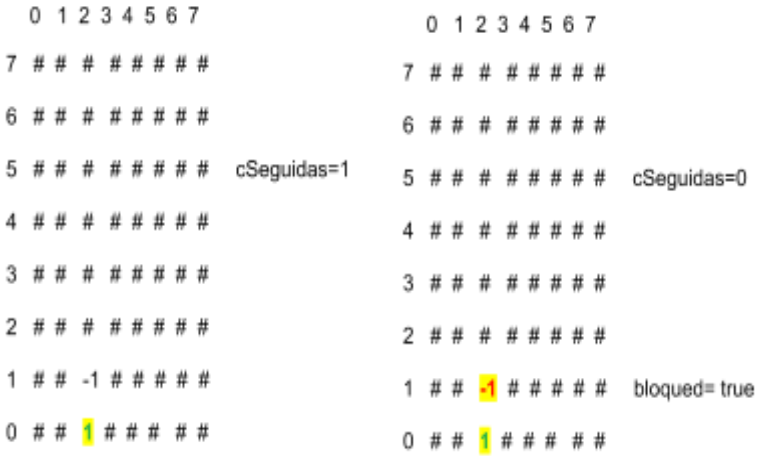
- Tenemos nuestra ficha (1) en una casilla entonces lo primero que hacemos es mirar si tiene alguna ficha en horizontal que sea igual y si es así aumenta el contador de cSeguidas+1.

- En el caso que encuentre una ficha del rival (-1), significa que esta casilla está bloqueada, si no seguida hasta que el número de casillas visitadas sea más grande que 4.



En el caso de la vertical es lo mismo que la horizontal:

- Como la casilla está en la posición 0 revisamos si hay alguna ficha por encima de ella que sea del mismo jugador en caso de que sea la ficha del rival esta casilla quedaría bloqueada el número de casillas seguida vuelve a ser 0.

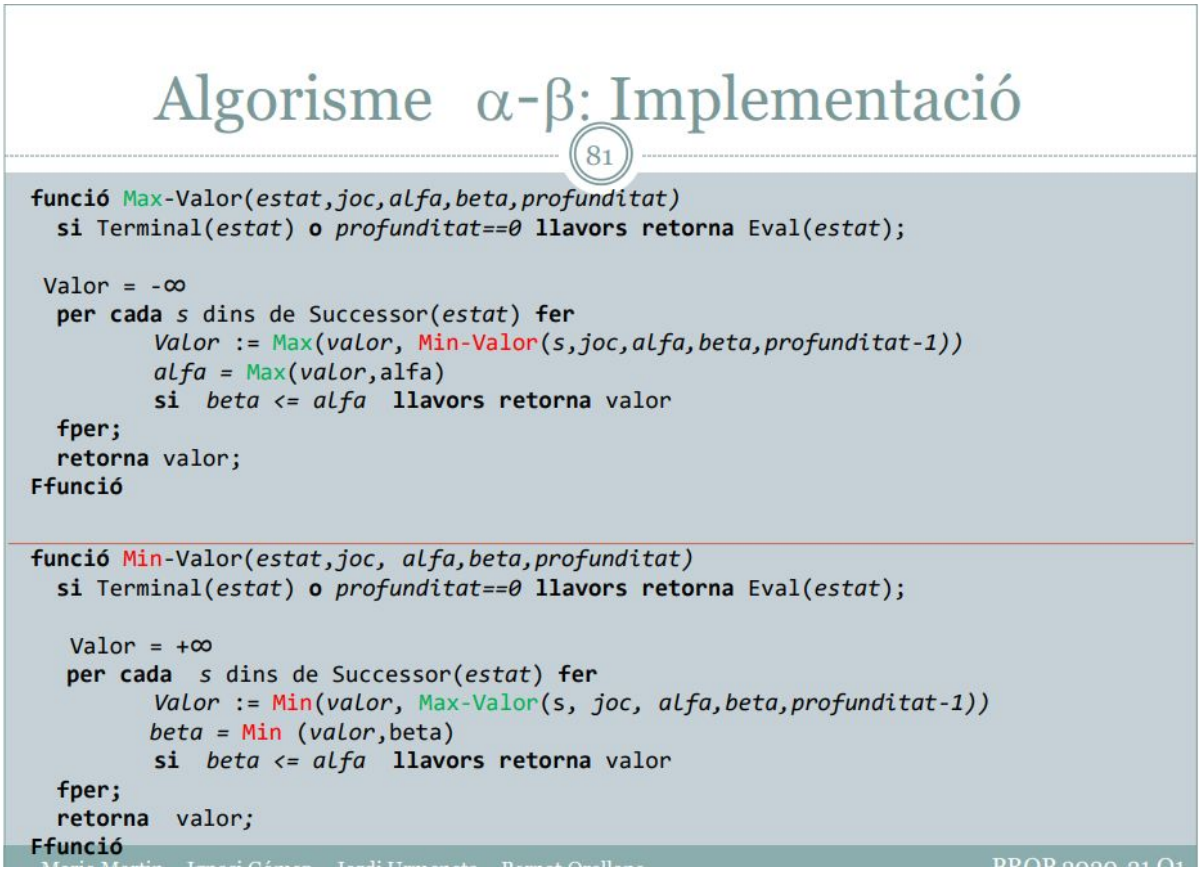


- Si la ficha que encontramos es nuestra aumentaría el contador de casillas seguidas y seguirá así hasta que el número de casillas visitadas (cVistas) sea más grande que 4.

Y para las diagonales hacemos lo mismo, revisamos si es la misma ficha o no y vamos aumentando los contadores

▪Estructuración del código e implementación de minimax y poda

Inspirandonos en la teoría de la asignatura:



Algorisme α-β: Implementació

81

```
funció Max-Valor(estat,joc,alfa,beta,profunditat)
  si Terminal(estat) o profunditat==0 llavors retorna Eval(estat);

 Valor = -∞
  per cada s dins de Successor(estat) fer
         Valor := Max(valor, Min-Valor(s,joc,alfa,beta,profunditat-1))
         alfa = Max(valor,alfa)
         si  beta <= alfa  llavors retorna valor
  fper;
  retorna valor;
Ffunció
```

```
funció Min-Valor(estat,joc, alfa,beta,profunditat)
  si Terminal(estat) o profunditat==0 llavors retorna Eval(estat);

   Valor = +∞
   per cada  s dins de Successor(estat) fer
         Valor := Min(valor, Max-Valor(s, joc, alfa,beta,profunditat-1))
         beta = Min (valor,beta)
         si  beta <= alfa  llavors retorna valor
  fper;
  retorna  valor;
Ffunció
```

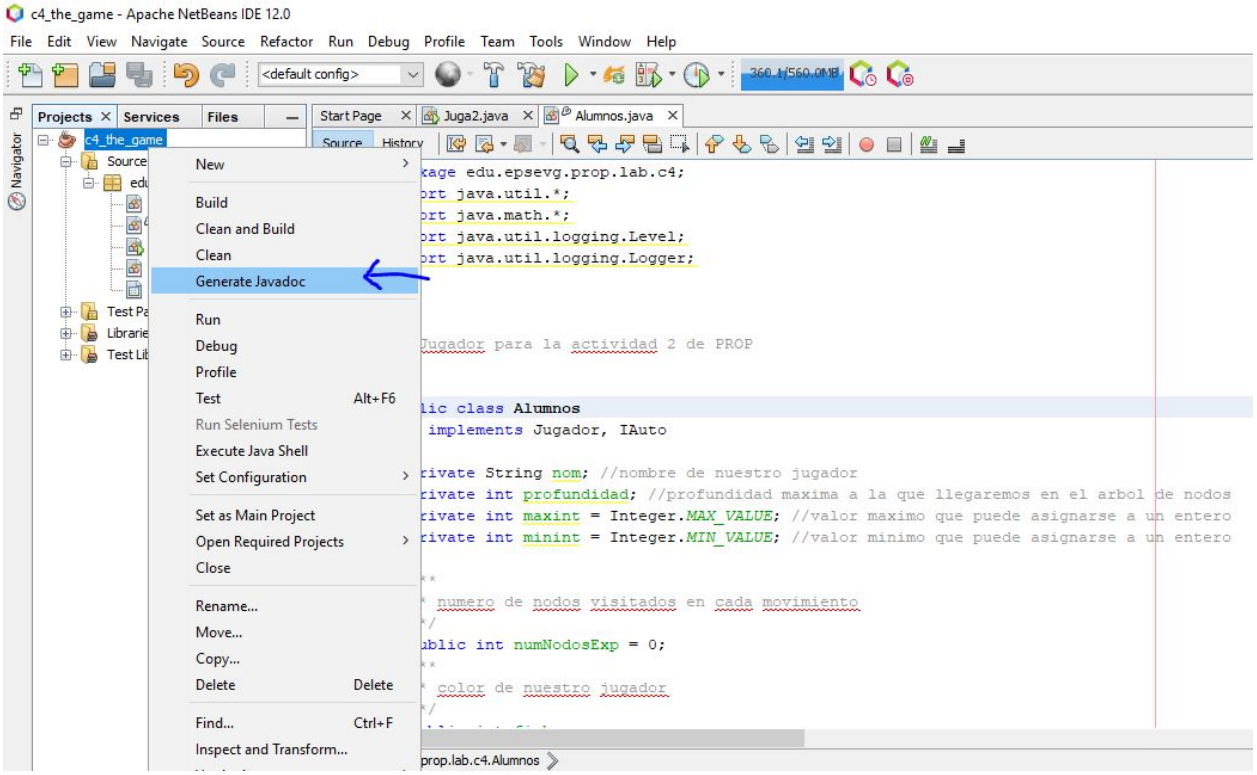
implementamos algo similar adaptándolo a nuestro algoritmo Min-Max.

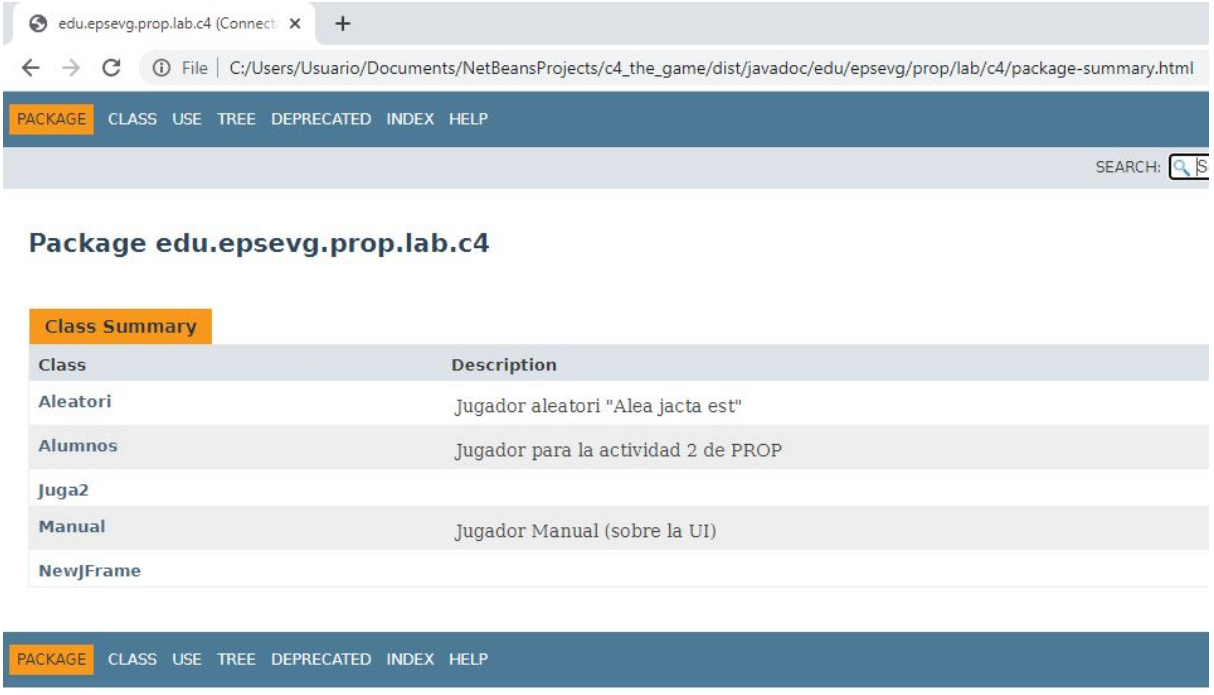
¿Qué incidencias tiene implementar la poda alfa-beta respecto a no utilizarla?

Al implementar la poda Alpha-Beta en nuestro algoritmo MiniMax, se consigue reducir el número total de nodos que recorremos y por lo tanto el programa es más rápido.

Notas:

- El código está comentado y explica que hace cada apartado.
- En el caso que el javadoc no aparezca puede que haga falta generarlo, está comentado en el código por lo tanto si se ha de generar se hace de esta forma:


c4_the_game - Apache NetBeans IDE 12.0
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects Services Files
Start Page Juga2.java Alumnos.java
c4_the_game
New
Build
Clean and Build
Clean
Generate Javadoc
Run
Debug
Profile
Test Alt+F6
Run Selenium Tests
Execute Java Shell
Set Configuration
Set as Main Project
Open Required Projects
Close
Rename...
Move...
Copy...
Delete Delete
Find... Ctrl+F
Inspect and Transform...
kage edu.epsevg.prop.lab.c4;
ort java.util.*;
ort java.math.*;
ort java.util.logging.Level;
ort java.util.logging.Logger;
Jugador para la actividad 2 de PROP
lic class Alumnos
implements Jugador, IAuto
rivate String nom; //nombre de nuestro jugador
rivate int profundidad; //profundidad maxima a la que llegaremos en el arbol de nodos
rivate int maxint = Integer.MAX_VALUE; //valor maximo que puede asignarse a un entero
rivate int minint = Integer.MIN_VALUE; //valor minimo que puede asignarse a un entero
* numero de nodos visitados en cada movimiento
ublic int numNodosExp = 0;
* color de nuestro jugador
prop.lab.c4.Alumnos


edu.epsevg.prop.lab.c4 (Connect
File | C:/Users/Usuario/Documents/NetBeansProjects/c4_the_game/dist/javadoc/edu/epsevg/prop/lab/c4/package-summary.html
PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
SEARCH:
Package edu.epsevg.prop.lab.c4
Class Summary
Class
Description
Aleatori
Jugador aleatori "Alea jacta est"
Alumnos
Jugador para la actividad 2 de PROP
Juga2
Manual
Jugador Manual (sobre la UI)
NewJFrame
PACKAGE CLASS USE TREE DEPRECATED INDEX HELP