Programació 1

Sessions de laboratori i enunciat de pràctiques

Autor: Jordi Esteve

(Curs 2015/16 - Q2)

Versió 1.0 - 2016.04.21

NORMATIVA DE LA PRÀCTICA

Aquesta és la normativa de la pràctica. És important que la llegiu i la seguiu. Incomplir la normativa pot suposar el suspens de l'assignatura.

1. La qualificació obtinguda a pràctiques fa mitjana ponderada amb la nota de teoria amb un pes del 35% sobre la nota total de l'assignatura.
2. La pràctica és obligatòria. La no presentació de la pràctica resultarà en un NO PRESENTAT com a nota final de l'assignatura.
3. La nota de la pràctica està formada per dues parts: programa i prova de validació individual (PVI). La nota de la PVI serà un número real entre 0 i 1. La fórmula per calcular la nota final de laboratori és la següent: NotaPrograma * NotaPVI.
4. La pràctica es realitzarà en equips de màxim dues persones. Serà motiu de no acceptació de la pràctica l'haver estat realitzada per més de dues persones.
5. Si dos o més treballs presentats són iguals, no serà acceptat **cap** dels treballs. El criteri del professor decidirà si dues pràctiques es poden considerar iguals o no. No s'admetran reclamacions sobre aquesta consideració.
6. Les pràctiques acceptades s'avaluaran mitjançant:

   - La correctesa i claredat del programa.
   - La correctesa de la documentació annexa i/o comentaris del programa.
   - L'execució del programa.

   Una pràctica que no funcioni pot ser igualment acceptada i avaluada pel professor.
7. El lliurament de la pràctica s'efectuarà a través del campus digital, en les condicions i format especificats a l'enunciat. L'incompliment de qualsevol d'aquestes condicions pot suposar la nota de NO PRESENTAT a la pràctica.
8. Un cop presentada la pràctica, tindrà lloc la Prova de Validació Individual. La PVI pot realitzar-se mitjançant una entrevista presencial o bé en el Control Final de l'assignatura. En cas que la PVI sigui una defensa presencial, el professor farà públic un mecanisme (una llista per apuntar-se o similar) tal de reservar data i hora per a cada membre de l'equip. La PVI també té en compte els exercicis preparatoris realitzats a la plataforma jutge.org.
9. Els estudiants del mateix equip poden obtenir notes diferents a la PVI, resultant en notes de laboratori diferents per a cada un.
10. La no compareixença a la PVI d'un estudiant farà que la seva nota de laboratori sigui NO PRESENTAT, encara que hagi fet el lliurament de la pràctica.

1 Introducció

Volem construir un programa que ens permeti jugar a l'Otelo entre 2 jugadors o entre un jugador i l'ordinador.

L'Otelo (també anomenat Othello o Reversi) és un joc d'estratègia entre dos jugadors i es juga sobre un tauler de 64 caselles. Es comparteixen 64 peces iguals, totes amb una cara blanca i una de negra. Cada jugador té assignat un d'aquests colors. Els jugadors van col·locant, per torns, les seves peces sobre el tauler i guanya el jugador que aconsegueix tenir més peces del seu color a l'acabar la partida.

Les files del tauler s'enumeren de 1 a 8 mentre que les columnes ho fan de la A a la H. Inicialment, dues peces negres se situen en les caselles E4 i D5, i dues peces blanques se situen a les caselles D4 i E5. A l'Otelo sempre comença el jugador que tingui les peces negres.

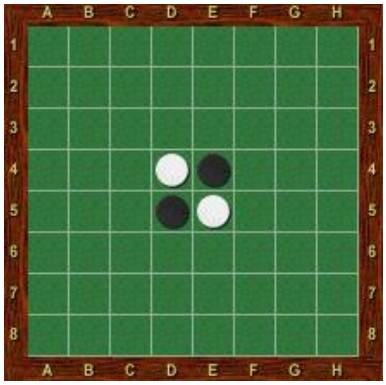


Figura 1: Tauler inicial de l'Otelo

L'objectiu general de la pràctica és escriure un programa en C++ que permeti jugar a l'Otelo usant vàries classes auxiliars. Per aconseguir-ho, cal seguir els passos i exercicis que es donen en la resta del document. Cal presentar la pràctica en tres lliuraments (cada lliurament suposa un terç de la nota de la pràctica):

1. Implementareu les classes auxiliars amb els seus mètodes corresponents i les testejareu individualment.

2. Implementareu un programa que permeti jugar a Otelo entre 2 jugadors humans o un jugador humà i l'ordinador que triarà les jugades a l'atzar.

3. Millorareu el programa anterior per afegir l'opció de jugar Otelo entre un jugador humà i l'ordinador que triarà les jugades segons un algorisme minimax per analitzar fins a una certa profunditat les possibles jugades dels dos contrincants usant un parell de funcions per avaluar les peces en el taulell.

2 Regles del joc Otelo

1. Hi ha un tauler de 64 caselles i 64 peces amb dues cares, una blanca i una de negra. Es comença amb quatre peces, dues de blanques i dues de negres, a les posicions centrals. Comença el jugador que hagi triat el color negre.

2. A cada torn, els jugadors han de col·locar una peça del seu color sobre una casella del tauler que estigui buida i que estigui al costat d'una casella amb una peça del contrincant. A més, s'ha de complir que, amb aquest moviment, una o més peces del contrincant quedin envoltades per la peça que acaba de col·locar i per una altra peça del seu color (ja col·locada en una tirada anterior). En aquest moment, les peces del contrincant que han quedat envoltades, han de girar-se i, per tant, esdevenen del mateix color que les del jugador que acaba de fer el seu moviment.

3. El jugador que hagi de fer un moviment, està obligat a girar peces del contrincant. Si no pot girar-ne cap, passa el torn al contrincant. Així, partint del tauler inicial 1, el jugador negre té quatre moviments possibles: 4C, 3D, 6E o 5F. Si optés per col·locar una peça a la posició 5F, el resultat seria:

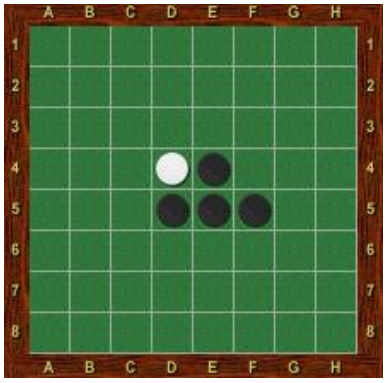
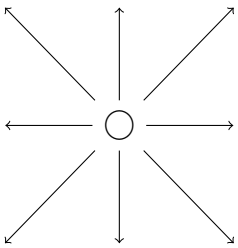


Figura 2: El jugador negre ha col·locat una peça a la posició 5F

Un sol moviment d'un jugador pot envoltar les peces del seu contrincant en una o més d'una de les 8 direccions possibles. En cadascuna d'aquestes direccions pot girar més d'una peça del seu contrincant.



Per exemple, suposem que en el tauler 3 el jugador negre vol posar una peça a la posició 6C, marcada en vermell:

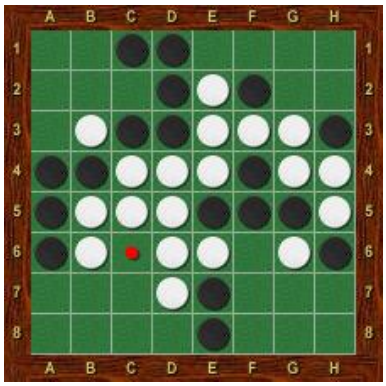


Figura 3: El jugador negre vol col·locar una peça a la posició 6C

El resultat seria el següent:

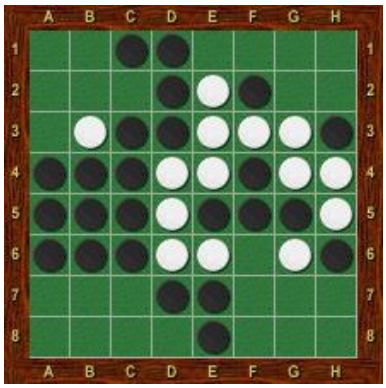


Figura 4: El jugador negre ha col·locat una peça a la posició 6C

Observeu que només es giren les peces envoltades entre la peça que s'acaba de col·locar (6C) i una altra peça del mateix color (o altres peces del mateix color) que ja estava al tauler, no hi ha reacció en cadena.

4. Guanya el jugador que tingui més peces del seu color en el tauler, a l'acabar la partida.

5. El joc s'acaba quan cap jugador pot moure, hagi quedat ple de peces el tauler o no.

3 Format del tauler de l'Otelo

Si bé el joc Otelo es juga amb taulells de 8x8 caselles, nosaltres permetrem jugar amb taulells de n x n caselles, sent $n \geq 2$.

Durant la partida caldrà mostrar el tauler de joc sempre de la mateixa manera. Per simplificar el programa tant les files com les columnes han d'estar enumerades de 1 a n. Les caselles o posicions on hi hagi peces blanques es mostren amb una 'B' i les negres amb una 'N'. Totes les posicions on el jugador actual pugui moure-s'hi es mostren amb un '?'. La resta de posicions es mostren amb un '-'. Per exemple en un taulell 8x8:

```
  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - - - - - - - -
3 - - - ? - - - -
4 - - ? B N - - -
5 - - - N B ? - -
6 - - - - ? - - -
7 - - - - - - - -
8 - - - - - - - -
```

Primer indicarem la fila i després la columna. Com es pot veure en aquest exemple, que correspon al tauler inicial de joc, hi ha dues peces negres a les posicions (4,5) i (5,4) que es mostren amb una 'N', i dues peces blanques a les posicions (4,4) i (5,5), que es mostren amb una 'B'. Les posicions (3,4), (4,3), (5,6) i (6,5), es mostren amb un '?' perquè són posicions on el jugador actual (el negre) podria col·locar-hi una de les seves peces. La resta de posicions, no contenen cap peça i tampoc no són vàlides per al jugador actual, per tant, es mostren amb un '-'.

Per fer el programa, usarem una estructura de dades `taulell` predefinida, amb les següents propietats:

- El taulell està representat per una taula de dues dimensions, amb n files i n columnes.
- Cada casella té unes coordenades, amb origen a (0,0).
- Cada casella pot estar lliure o ocupada per un fitxa blanca o negra.
- S'aniran col·locant peces blanques/negres i girant les que ja hi han segons les normes d'Otelo explicades anteriorment.

4 Primer lliurament: Classes bàsiques

Abans de començar a construir el programa que permeti jugar a Otelo definirem algunes classes bàsiques que ens resultaran útils més tard.

Les classes bàsiques a definir són:

- Classe `coord`: Ens permetrà tenir variables de tipus `coord` que guardin un parell de coordenades (x,y). Tindrem operacions per sumar coordenades i saber així les caselles veïnes en una certa direcció (per ex. `(3,5)+(0,-1)=(3,4)`).
- Classe `direccio`: Contindrà la llista de direccions possibles a mirar des d'una coordenada (Nord/Nord-Est/Est/Sud-Est/Sud/Sud-Oest/Oest/Nord-Oest). Per tant, tindrem variables de tipus `direccio` amb operacions com l'autoincrement per saber la següent direcció segons el sentit de les agulles del rellotge, o per obtenir les components del desplaçament en aquella direcció (per ex. `Est -> (0,+1)`).
- Classe `casella`: Contindrà la informació d'una casella del taulell, com per exemple l'ocupació (color) de la casella (lliure, blanca, negra) i si està visitada o no.
- Classe `tauell`: Servirà per guardar la taula n x n de caselles del joc d'Otelo. També ofereix mètodes per escriure el taulell a la pantalla, avaluar el taulell, veure moviments possibles, gira fitxes, ...

El primer lliurament consisteix en programar aquestes classes i uns programes que en provin el funcionament.

Per fer-ho, cal partir dels fitxers següents, que contenen un esquelet de les classes anteriors, que facilitarà la feina de programar-les.

Els fitxers són a `/home/public/pro1/practica/L1.inicial`:

`coord.h`	Definició dels atributs i mètodes de la classe `coord`.
`direccio.h`	Definició dels atributs i mètodes de la classe `direccio`.
`casella.h`	Definició dels atributs i mètodes de la classe `casella`.
`taulell.h`	Definició dels atributs i mètodes de la classe `taulell`.
`coord.cc`	Implementació dels mètodes de la classe `coord`.
`direccio.cc`	Implementació dels mètodes de la classe `direccio`.
`casella.cc`	Implementació dels mètodes de la classe `casella`.
`taulell.cc`	Implementació dels mètodes de la classe `tauell`.

En el mateix directori, hi ha els següents programes de prova, que també cal completar, i usar per verificar que hem programat correctament les classes anteriors.

`proves_coord.cc`	Programa de prova de la classe `coord`.
`proves_direccio.cc`	Programa de prova de la classe `direccio`.
`proves_casella.cc`	Programa de prova de la classe `casella`.
`proves_taulell.c`	Programa de prova de la classe `taulell`.

- Tots els fitxers tenen comentaris amb pistes sobre què cal fer.
- A les sessions de laboratori es donaran més detalls sobre l'estructura del programa a construir.
- Vegeu l'apèndix *A.-Lliuraments* al final d'aquest document per més detalls sobre com i quan fer els lliuraments.

5 Segon lliurament: Programa Otelo bàsic

El segon lliurament consisteix en programar una primera versió del programa per jugar a Otelo. Afegirem un argument al programa principal que permetrà jugar a Otelo:

- Entre 2 jugadors humans.
- Entre un jugador humà i l'ordinador que triarà les jugades a l'atzar.

Per fer-ho, cal partir del fitxer següent que conté un esquelet de programa principal:

`otelo.cc` Esquelet programa que permet jugar a Otelo.

El fitxer és a `/home/public/pro1/practica/L2.inicial`:

- Tots els fitxers tenen comentaris amb pistes sobre què cal fer.
- A les sessions de laboratori es donaran més detalls sobre l'estructura del programa a construir.
- Vegeu l'apèndix *A.-Lliuraments* per més detalls sobre com i quan fer els lliuraments.

5.1 Funcionament del programa

En aquesta secció es mostra el funcionament bàsic d'una possible partida del joc de l'Otelo.

1. Incialitzar el taulell de mida n x n i el torn a les fitxes NEGRES.
2. Comprovar que el jugador que té el torn pugui fer algun moviment. En cas que no pugui, haurà de passar el torn al seu contrincant.
3. Mostrar per pantalla el taulell indicant el jugador (color) que té el torn.
4. Demanar al jugador que té el torn un moviment, és a dir, una posició (fila,columna) on vulgui col·locar la seva peça. Si la fila o la columna no estan entre 1 i n s'ha acabat el joc i saltem al pas 9.
5. Cal comprovar que el moviment sigui vàlid:
   (a) Ha de correspondre a una posició buida.
   (b) Ha de permetre girar alguna peça del seu contrincant.
6. Un cop validat el moviment, cal localitzar totes les peces del contrincant que hagin quedat envoltades entre la nova peça i alguna altra del mateix color, i girar-les. Això caldrà fer-ho en les 8 direccions possibles.
7. Mostrar el resultat provisional de la partida i canviar el torn.
8. Repetir el procediment (des de 2) fins que s'acabi la partida o bé algun dels dos jugadors vulgui aturar-la introduint una fila o columna fora de l'interval [1,n].
9. Tant si s'acaba la partida com si s'ha aturat sense finalitzar-la, es mostra per pantalla el taulell i el resultat final de la partida.

5.2 Exemple de joc

Una possible partida amb un taulell 8x8 seria la següent:

```
========== Jugador N ==========
  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - - - - - - - -
3 - - - ? - - - -
4 - - ? B N - - -
5 - - - N B ? - -
6 - - - - ? - - -
7 - - - - - - - -
8 - - - - - - - -
Fila (1-8): 1
Columna (1-8): 1
Error: Posició ja ocupada o moviment no vàlid.
Fila (1-8): 6
Columna (1-8): 6
Error: Posició ja ocupada o moviment no vàlid.
Fila (1-8): 3
Columna (1-8): 4
Està guanyant el jugador N.
========== Jugador B ==========
  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - - - - - - - -
3 - - ? N ? - - -
4 - - - N N - - -
5 - - ? N B - - -
6 - - - - - - - -
7 - - - - - - - -
8 - - - - - - - -
Fila (1-8): 3
Columna (1-8): 3
Esteu empatant.
========== Jugador N ==========
  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - - - - - - - -
3 - ? B N - - - -
4 - - ? B N - - -
5 - - - N B ? - -
```

```
6 - - - - ? - - -
7 - - - - - - - -
8 - - - - - - - -
Fila (1-8): 4
Columna (1-8): 3
Està guanyant el jugador N.
========== Jugador B ==========
  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - - - - - - - -
3 - - B N ? - - -
4 - - N N N - - -
5 - - ? N B - - -
6 - - - - - - - -
7 - - - - - - - -
8 - - - - - - - -
Fila (1-8): 3
Columna (1-8): 5
Està guanyant el jugador B.
========== Jugador N ==========
  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - ? ? ? ? ? - -
3 - - B B B ? - -
4 - - N N B ? - -
5 - - - N B ? - -
6 - - - - - ? - -
7 - - - - - - - -
8 - - - - - - - -
Fila (1-8): 0
Columna (1-8): 0
========== Fi partida ==========
  1 2 3 4 5 6 7 8
1 - - - - - - - -
2 - - - - - - - -
3 - - B B B - - -
4 - - N N B - - -
5 - - - N B - - -
6 - - - - - - - -
7 - - - - - - - -
8 - - - - - - - -
Ha guanyat el jugador B.
```

Us poden ser útils les següents funcions, definides en el fitxer `util.h`:

-

```
//—- Neteja la pantalla
void neteja();

//—- Espera el numero de segons indicat abans de continuar.
//—- És útil per evitar que el programa s'executi tan ràpid
//—- que no doni temps de veure com es va dibuixant el camí.
void espera(float);
```

6 Tercer lliurament: Programa Otelo avançat

El tercer lliurament consisteix en millorar el programa anterior per afegir l'opció de jugar Otelo entre un jugador humà (negres) i l'ordinador (blanques) que seleccionarà les jugades de les blanques segons un algorisme minimax per analitzar fins a una certa profunditat les possibles jugades dels dos contrincants usant un mètode per avaluar les peces en el taulell.

S'afegeixen tres arguments al programa principal que permeten indicar:

- La profunditat màxima a la que ha d'arribar la cerca minimax. Per defecte 1.
- El mètode (funció heurística) a usar per avaluar el taullell quan encara no s'ha acabat la partida.
  - 0: Avalua les fitxes del taulell blanques-negres (figura 6). És el mètode per defecte.
  - 1: Avalua les fitxes del taulell ponderant segons posició (figura 7): Les caselles blanques de la primera/última fila/columna les multiplica per 5, les caselles blanques de les 4 cantonades per 10, les caselles negres de la primera/última fila/columna per -5 i les caselles negres de les 4 cantonades per -10.
- Com depura la funció minimax (veure els fitxers `.out` de la carpeta `L3.test`):
  - 0: No es depura (la sortida a cout serà la de una partida normal). És l'opció per defecte.
  - 1: Imprimirà el taullel abans de cada crida recursiva.
  - 2: Imprimirà el valor calculat per minimax després de cada crida recursiva.

6.1 Algorisme minimax per la presa de decisions

Minimax és un mètode de decisió en jocs bipersonals que minimitza la pèrdua màxima esperada. El funcionament de Minimax es pot resumir com triar el millor moviment per a tu mateix suposant que el teu contrincant escollirà el pitjor per a tu.

Passos de l'algorisme Minimax (veure figura 5, podeu trobar més informació a la pàgina http://ca.wikipedia.org/wiki/Minimax):

1. Generació de l'arbre del joc. Es generaran tots els nodes (successions de jugades en el taulell) fins a arribar a un estat terminal (o bé s'ha finalitzat la partida o bé s'ha arribat a la profunditat màxima).
2. Càlcul dels valors de la funció d'utilitat per a cada node terminal.
3. Calcular el valor dels nodes superiors a partir del valor dels inferiors. Alternativament s'escolliran els valors màxims i mínims representant els moviments del jugador i de l'oponent (d'aquí ve el nom de Minimax).
4. Escollir la jugada valorant els valors que han arribat al nivell superior (la jugada associada al valor màxim del nivell superior).

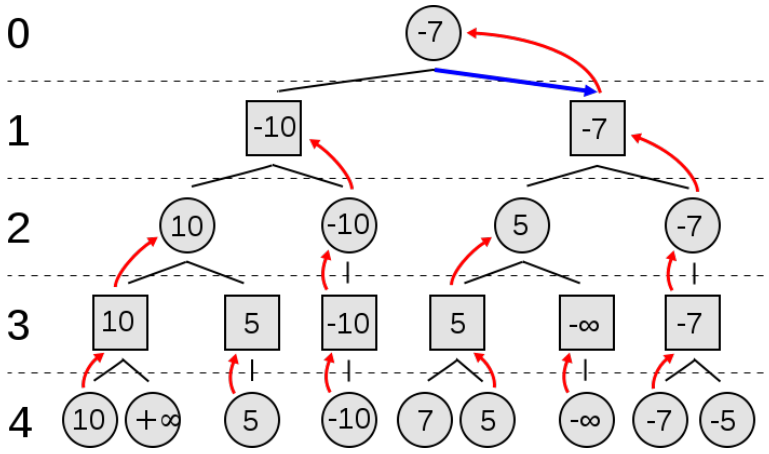


Figura 5: Exemple evolució algorisme Minimax. 1) Expansió de les jugades. 2) Càlcul valor nodes terminals. 3) Selecció alternativa de mínims i màxims. 4) La jugada final seleccionada és la de la fletxa blava.

6.2 Aplicació algorisme minimax a Otelo

En el nostre cas, amb el mètode minimax caldrà calcular bones jugades per a les blanques (és el jugador ordinador). Per tant l'algorisme minimax haurà de maximitzar els valors per a les blanques (s'ha de triar el millor per a les blanques) i minimitzar per a les negres (suposem que les negres escollirien el pitjor per a les blanques). En cas d'empat en la valoració, triarem la jugada amb la fila menor i per la mateixa fila, la jugada de columna menor.

No és necessari construir un arbre amb totes les jugades, a l'implementar l'algorisme minimax de forma recursiva i anar fent còpia del taulell abans de fer la següent jugada serà suficient per expandir i analitzar totes les jugades.

Com que és temporalment impossible analitzar totes les jugades entre blanques i negres en una partida de Otelo a no ser que estem a les acaballes de la partida, s'analitzaran les jugades fins a una certa profunditat (una partida d'Otelo en un taulell de 8x8 pot arribar a tenir 60 jugades i suposant que cadascuna d'elles té un promig de 5 alternatives de jugades a triar caldria analitzar $5^{60} = 867361737988403547205962240695953369140625$ jugades). Tingueu en compte que a Otelo no sempre hi ha una alternança entre blanques i negres, ja que a vegades un dels jugadors ha de passar.

Tenim casos finals quan:

- S'acaba la partida (ni blanques ni negres poden tirar). Suposarem que la puntuació serà de 1000 si guanyen les blanques, 0 si empaten i -1000 si guanyen les negres (per a les blanques és molt bo que guanyin les blanques i molt dolent que guanyin les negres).

- Hem arribat a la profunditat màxima. En aquest cas usarem una funció per avaluar les fitxes en el taulell (podem triar 2 mètodes possibles, un de simple que sumi número

de fitxes blanques menys número de fitxes negres (figura 6), i un altre de més efectiu que sumi les blanques menys les negres però ponderant per un factor segons a on estigui situada la fitxa ja que les caselles més externes són estratègiques en el joc de Otelo (figura 7)).

	1	2	3	4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1

	1	2	3	4
1	-1	-1	-1	-1
2	-1	-1	-1	-1
3	-1	-1	-1	-1
4	-1	-1	-1	-1

Figura 6: Pesos ponderació mètode 0. Esquerra per les blanques. Dreta per les negres.

	1	2	3	4
1	10	5	5	10
2	5	1	1	5
3	5	1	1	5
4	10	5	5	10

	1	2	3	4
1	-10	-5	-5	-10
2	-5	-1	-1	-5
3	-5	-1	-1	-5
4	-10	-5	-5	-10

Figura 7: Pesos ponderació mètode 1. Esquerra per les blanques. Dreta per les negres.

Per aquest últim cas caldrà ampliar els fitxers *taulell.h* i *taulell.cc* per implementar el mètode *avalua_posicio()* que avalua les fitxes del taulell (blanques-negres) ponderat segons posició: Les caselles blanques de la primera/última fila/columna les multiplica per 5, les caselles blanques de les 4 cantonades per 10, les caselles negres de la primera/última fila/columna per -5 i les caselles negres de les 4 cantonades per -10.

6.3 Exemple d'avaluació minimax amb Otelo

A continuació es mostra un exemple d'aplicació de l'algorisme d'avaluació minimax en un taulell d'Otelo de mida 4x4 amb una profunditat màxima de 2.

El jugador blanc ha de tirar i pot triar 3 possibilitats:

```
========== Jugador B ==========
  1 2 3 4
1 ? N ? -
2 - N N -
3 ? N B -
4 - - - -
```

Aquí veiem com s'analitza la seqüència de jugades en les 3 profunditats: La jugada de profunditat 2 que toca tirar les blanques, per cadascuna d'elles 3 jugades de profunditat 1 que toca tirar les negres, i per cadascuna d'aquestes les jugades de profunditat 0 que tornen a tirar les blanques (i com que hem arribat a la profunditat màxima, s'avaluen les fitxes del taulell, en aquest exemple amb el segon mètode de ponderació de les fitxes).

```
PROF: 2 COLOR: B
  1 2 3 4
1 - N - -
2 - N N -
3 - N B -
4 - - - -

PROF: 1 COLOR: N
  1 2 3 4
1 B N - -
2 - B N -
3 - N B -
4 - - - -

PROF: 0 COLOR: B
  1 2 3 4
1 B N - -
2 N N N -
3 - N B -
4 - - - -
VALOR: -2

PROF: 0 COLOR: B
  1 2 3 4
1 B N - -
```

```
2 - B N -
3 - N N N
4 - - - -
VALOR: -2

PROF: 0 COLOR: B
  1 2 3 4
1 B N - -
2 - B N -
3 - N N -
4 - - N -
VALOR: -2

PROF: 1 COLOR: N
  1 2 3 4
1 - N B -
2 - N B -
3 - N B -
4 - - - -

PROF: 0 COLOR: B
  1 2 3 4
1 - N N N
2 - N N -
3 - N B -
4 - - - -
VALOR: -22

PROF: 0 COLOR: B
  1 2 3 4
1 - N B -
2 - N N N
3 - N B -
4 - - - -
VALOR: -7

PROF: 0 COLOR: B
  1 2 3 4
1 - N B -
2 - N N -
3 - N N N
4 - - - -
VALOR: -9
```

```
PROF: 0 COLOR: B
  1 2 3 4
1 - N B -
2 - N B -
3 - N N -
4 - - - N
VALOR: -12

PROF: 1 COLOR: N
  1 2 3 4
1 - N - -
2 - N N -
3 B B B -
4 - - - -

PROF: 0 COLOR: B
  1 2 3 4
1 - N - -
2 - N N -
3 B N B -
4 N - - -
VALOR: -12

PROF: 0 COLOR: B
  1 2 3 4
1 - N - -
2 - N N -
3 B N B -
4 - N - -
VALOR: -7

PROF: 0 COLOR: B
  1 2 3 4
1 - N - -
2 - N N -
3 B B N -
4 - - N -
VALOR: -7
```

```
PROF: 0 COLOR: B
  1 2 3 4
1 - N - -
2 - N N -
3 B B N -
4 - - - N
VALOR: -12
```

Finalment s'agafen les mínimes valoracions de la profunditat 0 per calcular les de la profunditat 1, i s'agafen les màximes valoracions de la profunditat 1 per calcular la de la profunditat 2, en aquest cas el $valor = -2$ que és la jugada a la coordenada (1,1) que finalment serà la jugada seleccionada que faran les blanques.

```
PROF: 0 COLOR: B VMILLOR: -2
PROF: 0 COLOR: B VMILLOR: -2
PROF: 0 COLOR: B VMILLOR: -2
PROF: 1 COLOR: N VMILLOR: -2
PROF: 0 COLOR: B VMILLOR: -22
PROF: 0 COLOR: B VMILLOR: -7
PROF: 0 COLOR: B VMILLOR: -9
PROF: 0 COLOR: B VMILLOR: -12
PROF: 1 COLOR: N VMILLOR: -22
PROF: 0 COLOR: B VMILLOR: -12
PROF: 0 COLOR: B VMILLOR: -7
PROF: 0 COLOR: B VMILLOR: -7
PROF: 0 COLOR: B VMILLOR: -12
PROF: 1 COLOR: N VMILLOR: -12
PROF: 2 COLOR: B VMILLOR: -2
```

6.4 Consells finals

Tingueu en compte que:

- Tots els fitxers tenen comentaris amb pistes sobre què cal fer.
- A les sessions de laboratori es donaran més detalls sobre l'estructura del programa a construir.
- Vegeu l'apèndix *A.-Lliuraments* per més detalls sobre com i quan fer els lliuraments.

A Lliuraments

- Tots els lliuraments s'han de fer pel campus digital.
- Cada lliuraments consistirà en **un únic** fitxer comprimit, amb nom `dni1-dni2.zip` format pels DNIs dels membres del grup, ordenats de menor a major. El format pot ser .zip o qualsevol altre compressor disponible. Si el nom del fitxer no compleix aquest format, pot significar la qualificació de NO PRESENTAT per al lliurament.
- Aquest fitxer contindrà tots els fitxers del lliurament, sense cap altre subdirectori addicional.
- Cada lliurament ha de contenir *tots* els fitxers de codi de les llista corresponent (veure més avall). Els fitxers entre parèntesis provenen de lliuraments anteriors o de l'enunciat. Els que no estan entre parèntesis són els que cal programar o modificar respecte el lliurament anterior. Els que no apareixen a la llista, no han d'estar al paquet.

  Els fitxers que contindrà el paquet d'un lliurament són:

  - Un fitxer de **text pla** (.txt) amb el nom, cognoms i DNI dels membres de l'equip.
  - Els fitxers de codi necessaris per compilar la pràctica. El codi ha d'estar comentat descrivint quin és el resultat de cada funció (postcondició), i en quines condicions és aplicable (precondicio).
  - Programes de prova, fitxers de dades, i qualsevol altre material necessari per a provar la pràctica.

  Un paquet incomplet pot significar la qualificació de NO PRESENTAT per al lliurament. Un paquet es complert quan conté tot el necessari per compilar i executar la pràctica, apart dels fitxers addicionals demanats més amunt.

Les dates de cada lliurament, i el codi demanat són els següents:

- **Primer lliurament:**
  - *Data límit:* Diumenge 22 de maig a les 23:59.
  - *Codi a presentar:*

Definicions classes	*Implementacions classes*	*programes principals*
`coord.h`	`coord.cc`	`proves_coord.cc`
`direccio.h`	`direccio.cc`	`proves_direccio.cc`
`casella.h`	`casella.cc`	`proves_casella.cc`
`taulell.h`	`taulell.cc`	`proves_taulell.cc`

- **Segon lliurament:**
  - *Data límit:* Diumenge 5 de juny a les 23:59.
  - *Codi a presentar:*

Definicions classes	*Implementacions classes*	*programes principals*
(`coord.h`)	(`coord.cc`)	
(`direccio.h`)	(`direccio.cc`)	
(`casella.h`)	(`casella.cc`)	
(`taulell.h`)	(`taulell.cc`)	
(`util.h`)		
		`otelo.cc`

- **Tercer lliurament:**
  - *Data límit:* Diumenge 19 de juny a les 23:59.
  - *Codi a presentar:*

Definicions classes	*Implementacions classes*	*programes principals*
(`coord.h`)	(`coord.cc`)	
(`direccio.h`)	(`direccio.cc`)	
(`casella.h`)	(`casella.cc`)	
`taulell.h`	`taulell.cc`	
(`util.h`)		
		`otelo.cc`