Exercici 3 (2.5 punts) (RHEX). El joc de RHEX és una variació del joc de l'HEX inspirada en el Reversi. En aquest joc, hi han 2n fitxes, les fitxes tenen dos cares, una de color blanc i l'altre de color negre. El tauler es una xarxa $n \times n$ de cel.les hexagonals, com la de l'HEX. Hi han dos jugadors, un amb color negre i l'altre amb color blanc, que fan torns. A cada torn el jugador que li toca col·loca una fitxa mostrant el seu color a una posició encara buida. Un cop col·locada una fitxa al tauler, no es pot moure.

Quan es col.loca una fitxa al tauler es considera la *zona dominada*: totes les celles ocupades i connectades amb la nova cella amb un camí continuo de fitxes, normalment hi hauran fitxes de tos dos colors en aquests camins. Dues cel·les es consideren connectades si comparteixen una vora i les dues tenen una fitxa, independentment del color de les fitxes. Si a la zona dominada hi ha menys fitxes del color del jugador que té el torn que del color de l'altre jugador, es giren totes les fitxes a la zona dominada.

La partida acaba quan un dels dos jugadors guanya (aconsegueix tenir més de n fitxes del seu color) o quan s'hagin col·locat les $2n$ fitxes.

(a) (1.5 punts) Descriviu un algorisme eficient per al seguiment d'una partida. L'algorisme, després de cada jugada, ha de permetre mantenir el nombre total de fitxes de cada color al tauler i a més ha d'indicar si el jugador que acaba de jugar ha guanyat el joc de RHEX.

(b) (1 punt) Expliqueu com modificar el vostre algorisme per tal de que a més de mantenir la puntuació podeu actualitzar el tauler.

Solució.

El tablero tiene n^2 posiciones y en total solo puede haber $2n$ fichas. Por ello la estructura de datos que utilizaré solo contendrá información sobre las casillas en las que hay fichas.

Podemos asociar a cada estado del tablero, después de k, jugadas por un grafo $G(k)$ en el que los nodos son las posiciones de las k fichas y donde dos vértices están conectados si las casillas son contiguas.

(a) Observemos que la "zona dominada"es la componente conexa (cc) del grafo $G(k)$ que contiene la la ficha k-ésima.

La estructura eficiente para mantener las componentes conexas en un grafo que se va modifiando con el tiempo es Union Find (UF). Tendremos que modificar UF para que el nodo raíz de una componente guarde dos contadores n_u y b_u con el númeo de fichas negras, blancas, de la componente. Tendremos que añadir a la implementación de UF métodos que nos permitan actualizar los contadores.

Cuando introducimos una ficha de color c en la posición p:

- Makeset(p); actualizamos los contadores de p para que reflejen el color de la ficha.
- Hacemos un FIND para cada una de las posiciones contiguas y obtenemos la lista de hasta 6 identificadores diferentes de sets en la ED, podría darse el caso de que dos posiciones contiguas ya estuviesen en la misma cc.
- Calculamos n y b, sumando los contadores de las diferentes cc vecinas y sumamos 1 al color de la nueva ficha. Si se cumple la condición del problema intercambiamos los contadores. Actualizamos N y B.

- Hacemos UNION de todas las componentes conexas vecinas y del set correspondiente a la nueva ficha. Actualizamos los contadores de la cc para que coincidan con los valores calculados en el paso previo.
- Si N o B son $> n$ acabamos y si no continuamos con la siguiente ficha de color contrario.

El número total de operaciones UF por elemento es constante por lo que el coste del algoritmo es $O(n \log^* n)$.

Por definición el algoritmo mantiene correctamente las cc del tablero después de cada jugada y actualiza los contadores adecuadamente.

(b) Para modificar el tablero de forma eficiente observemos que solo pueden cambiar de color las fichas la cc que contiene la ficha que se acaba de colocar en el tablero. Necesitamos mantener información de los colores de las fichas y de las posiciones que forman cada componente conexa. Para ello añadiremos al nodo representante una lista de pares (posición,color) conteniendo todas las posiciones del tablero incluidas en la cc.

Esta información se puede mantener fácilmente. En un Makeset, añadimos una lista con un solo par. En un UNION, unimos las dos listas en tiempo constante. Con esta implementación podemos cambiar el color de las fichas en tiempo proporcional al número de fichas que cambian de color.