Práctica: Tournament Trees – Parte II
Sesión de reposición – Estructura de Datos

En esta práctica terminaremos el Min Win Tree que empezamos en la sesión 1

1. Termina los siguientes métodos de tu clase:
    a. Si fuiste a la sesión del miércoles, no terminamos el método void initialize(int[ ] a), nos faltó completar para los casos en los que el número de jugadores es par (no potencia de 2) y el caso el de número de jugadores impar. El código completo del método es el siguiente, por favor complétalo y en los comentarios correspondientes indica qué está sucediendo.

```
   public void initialize(int[] a){
        int n=a.length-1;
        //no puede haber árboles menores de 2 jugadores
        if (n < 2)
               throw new IllegalArgumentException("At least 2 players");
        //inicializo las variables de instancia
        players=a;
        tree=new int[n];
        //inicializo s, lowExt y offset
        //las cuales nos van a servir para determinar el nivel más bajo
        int i, s;
        for (s = 1; 2 * s <= n - 1; s += s);
        lowExt = 2 * (n - s);
        offset = 2 * s - 1;
        //Mando a play para formar el árbol de los nodos exteriores hacia
arriba
        for (i = 2; i <= lowExt; i += 2)
               play((offset + i) / 2, i - 1, i);
        // ¿Qué caso atiendo aquí?
        if (n % 2 == 1){//¿Qué hace aqui?
               play(n/2, tree[n - 1], lowExt + 1);
               i = lowExt + 3; //¿Para qué hago esto?
        }
        else{//¿Para qué hago esto y en qué caso se hace?
               i = lowExt + 2;
        }
        //¿Que hace aqui?
        for (; i <= n; i += 2)
               play((i - lowExt + n - 1) / 2, i - 1, i);
    }
```

    b. Agrega el método void replay(int i), donde i es el jugador que cambió. El método realiza la modificación del árbol, sólo en los nodos involucrados. Por favor documenta con comentarios, lo que realiza el método, identificando los casos que está atendiendo.

```
public void rePlay(int thePlayer)
{
   int n = players.length - 1;
   if (thePlayer <= 0 || thePlayer > n)
      throw new IndexOutOfBoundsException(thePlayer + "invalid player");

   int matchNode, leftChild, rightChild;
   if (thePlayer <= lowExt)
   {
      matchNode = (offset + thePlayer) / 2;
      leftChild = 2 * matchNode - offset;
      rightChild = leftChild + 1;
   }
   else
   {
      matchNode = (thePlayer - lowExt + n - 1) / 2;
      if (2 * matchNode == n - 1)
         leftChild = tree[2 * matchNode];
         rightChild = thePlayer;
      }
      else
      {
         leftChild = 2 * matchNode - n + 1 + lowExt;
         rightChild = leftChild + 1;
      }
   }

   tree[matchNode] = players[leftChild]<players[rightChild]? leftChild : rightChild;

  if (matchNode == n - 1 && n % 2 == 1)
    {
       matchNode /= 2;
       tree[matchNode] = players[tree[n - 1]]<
                         (players[lowExt + 1]) ?
                         tree[n - 1] : lowExt + 1;
    }
  matchNode /= 2;
   for (; matchNode >= 1; matchNode /= 2)
      tree[matchNode] = players[tree[2 * matchNode]]<
                        (players[tree[2 * matchNode + 1]]) ?
                        tree[2 * matchNode] : tree[2 * matchNode + 1];
}
```

   c. Crea el método void change(int theplayer, int value), el cual cambia el valor del jugador en la posición theplayer por el valor value, en el arreglo de jugadores.

2. Prueba tu clase.
3. Ahora utiliza tu Min Win Tree para ordenar los elementos de un arreglo. Puedes generar un método dentro de la misma clase, o utilizar tu estructura en un método externo. Consulta el proceso a seguir en tus diapositivas de clase.
   a. ¿Qué número utilizarás para reemplazar al elemento ganador, en cada replay?