UNIVERSIDAD DE ALMERÍA

Proyecto 1 Torneo de Tenis

Estructura de datos y algoritmos II

equipo-ualeda2-2024-simon-antonio

Venzal Sánchez, Simón María
Guerrero Crespo-López, Antonio

Índice

Proyecto 1 Torneo de Tenis

1.- Objetivo.

1.1 Introducción teórica.

La práctica se enmarca en el contexto de EDAland, un país ficticio donde las estructuras de datos y los algoritmos juegan un papel crucial en la resolución de problemas complejos, la tarea específica que nos ocupa es la organización de un torneo de tenis, conocido como EDAland-Garros, un evento de gran envergadura que requiere un sistema meticuloso para organizar los emparejamientos de los participantes. Este sistema debe asegurar que cada jugador se enfrente a cada uno de los demás competidores exactamente una vez, con la condición adicional de jugar un partido al día y tener, como máximo, un día de descanso durante todo el torneo.

En este contexto, se plantea el reto de desarrollar un algoritmo que no solo aborde eficientemente el problema sino que también sea capaz de adaptarse a diferentes números de participantes, incluyendo casos donde el número de jugadores es una potencia de dos, es par pero no una potencia de dos, o es impar. La solución a este problema complejo se busca a través del método algorítmico "Divide y Vencerás" (DyV).

1.1.1 ¿Qué es Divide y Vencerás?

Divide y Vencerás es una técnica algorítmica que aborda un problema dividiéndolo en subproblemas más pequeños, similares al original pero de menor tamaño. Estos se resuelven de manera independiente, y sus soluciones se unen para resolver el problema principal. Este método es particularmente efectivo para simplificar problemas complejos. A continuación mostramos un ejemplo de pseudocódigo genérico de un algoritmo de divide y vencerás.

Pseudocódigo genérico de un algoritmo DyV

```
método divideYVencerás(x) retorna y
  si x es suficientemente sencillo entonces
    // caso directo
    retorna algoritmoDirecto(x)
  fsi
  // caso recursivo
  descompone x en subproblemas x_1, x_2, .., x_s
  desde i := 1 hasta s hacer
    // llamadas recursivas
    y_i := divideYVencerás(x_i)
  fhacer
  // combina las soluciones
  y := combinación de las soluciones parciales (y_i)
  retorna y
fmétodo
```

Imagen 1: Pseudocódigo algoritmo genérico DyV.

1.1.2 Formas de Aplicar Divide y Vencerás.

Divide y Vencerás puede aplicarse de varias formas, según el caso a tratar y el objetivo:

- Descomposición Binaria: Dividir el problema en dos subproblemas de igual tamaño. Este enfoque es típico en algoritmos de ordenamiento como Quicksort y Mergesort.

- Descomposición en Múltiples Partes: Dividir el problema en más de dos subproblemas. Un ejemplo clásico es el Algoritmo de Strassen para la multiplicación de matrices.

- Reducción a un Caso Menor: Transformar el problema en un caso menor del mismo problema, como en la búsqueda binaria.

1.2 División del trabajo.

Para esta práctica, Simón ha hecho de líder, aunque al ser solo dos personas en nuestro grupo estamos trabajando juntos constantemente, de esta forma hemos ido avanzando a la vez por lo que no vemos conveniente dividir las horas de trabajo o el contenido de cada integrante del grupo ya que hemos hecho todo juntos en reuniones tanto presenciales como telemáticas.

2.- Antecedentes.

2.1 Motivación.

La principal motivación detrás de esta práctica surge de la necesidad de organizar eficientemente un torneo de tenis en EDALand, un país ficticio donde la utilización de estructuras de datos y algoritmos avanzados para resolver problemas complejos es común. El reto consistía en desarrollar un sistema que pudiera gestionar los emparejamientos de un número variable de jugadores de manera que cada uno compitiera contra todos los demás exactamente una vez, optimizando el uso del tiempo y recursos disponibles.

2.2 Alternativas planteadas.

Se consideraron varias estrategias algorítmicas para abordar el problema, incluyendo:

- Algoritmos Greedy: Se descartaron debido a su potencial para no encontrar la solución óptima global.

- Programación Dinámica: Considerada innecesariamente compleja para la naturaleza del problema.

- Backtracking: Aunque viable, se consideró demasiado lento para torneos con muchos participantes.

2.3 Solución final.

La solución adoptada fue el método "Divide y Vencerás", implementado a través de tres variantes específicas para manejar torneos con un número de jugadores que es potencia de dos, par e impar pero no potencia de dos y por último para N potencia de dos sin simplificación. Esta estrategia permitió una descomposición eficiente del problema en subproblemas más manejables, optimizando la organización del torneo.

3.- Trabajo a desarrollar.

3.1 Estudio de la implementación.

Decidimos desarrollar el código en Java, dado que es un lenguaje con el que poseemos una gran familiaridad y por continuar con lo realizado en la asignatura de EDA2 de este mismo grado. Han sido implementados tres métodos recursivos torneoRecursivoCaso1, torneoRecursivoCaso2, y torneoRecursivoCaso3, cada uno de ellos correspondiente a una de las variantes del problema.

El método divide y vencerás aplicado en esta práctica consiste en descomponer el problema de organizar un torneo de tenis en EDALand en subproblemas más pequeños y manejables. Cada subproblema trata sobre organizar emparejamientos para un número reducido de jugadores, y luego, los resultados de estos subproblemas se combinan para formar la solución al problema original. Este enfoque permite simplificar el proceso de organización del torneo, asegurando que cada jugador se enfrente a todos los demás una vez, respetando los requisitos de juegos diarios y descansos.

3.1.1 Caso 1.

Para N potencia de dos se dividen los jugadores en dos grupos iguales y se organizan los partidos primero dentro de cada grupo y luego entre grupos, asegurando que cada jugador juegue contra todos los demás.

Pseudocódigo Caso 1

```
Función torneoRecursivoCaso1(tabla, n)
  // Caso Base: Solo hay dos jugadores
  Si n == 2
    // Se enfrentan directamente entre sí
    tabla[1][1] = 2
    tabla[2][1] = 1
  // Caso Recursivo: Más de dos jugadores
  Sino
    // Aplica el algoritmo al primer grupo de jugadores (primera mitad)
    Llamar a torneoRecursivoCaso1 para la primera mitad de jugadores

    // Rellena el resto de la tabla basado en los enfrentamientos del primer grupo
    Para cada jugador en la segunda mitad
```

```
    // Ajusta enfrentamientos basados en la primera mitad
  Fin Para

    // Organiza enfrentamientos cruzados entre la primera y segunda mitad en los días restantes
  Fin Si
Fin Función
```

Tabla 1: Pseudocódigo caso 1.

A continuación se muestra una imágen de cómo se rellena una matriz con 8 jugadores teniendo en cuenta como actúa nuestro algoritmo haciendo uso de divide y vencerás para llegar a un caso base y ya desde ahí rellenar la tabla.

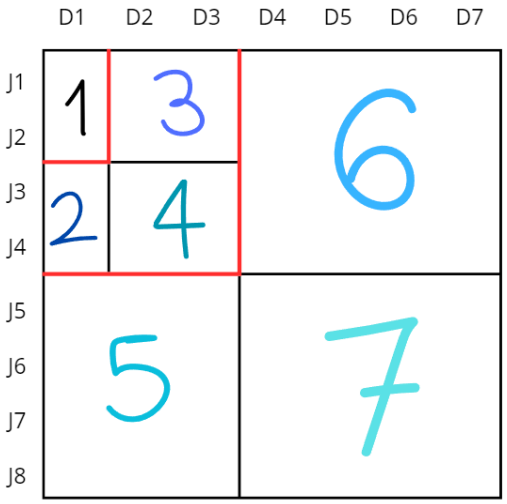

Imagen 2: División de la matriz para el caso 1.

A continuación se muestra gráficamente cómo se rellenan los diferentes cuadrantes desde el caso base

3.1.1.1 Cuadrante inferior izquierdo.

Para el cuadrante inferior izquierdo se suma el valor del cuadrante superior izquierdo con el valor de n/2, como podemos ver el la siguiente imágen recortada de la matriz resultante de aplicar el caso1 en una matriz de 8 jugadores, todos los valores cumplen la fórmula, la flecha azul por ejemplo, 2 (valor cuadrante sup. izq.) + n/2 (n=8) = 2+4 = 6. Lo mismo ocurre con todas las otras flechas y los otros valores de la matriz.

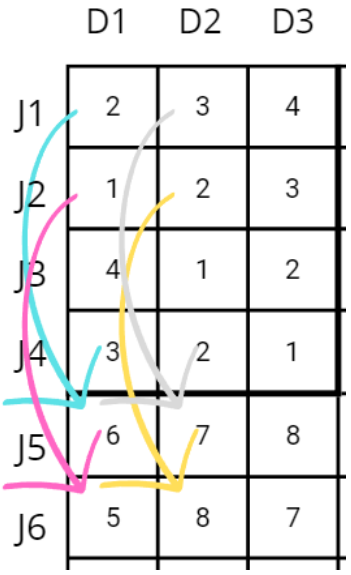

Imagen 3: Operaciones de DyV en el cuadrante inferior izquierdo para el caso 1.

3.1.1.2 Cuadrante superior derecho.

En el primer día de este cuadrante (día n/2), cada jugador del primer grupo se enfrenta directamente a su correspondiente en el segundo grupo. Es decir, el jugador 1 se enfrenta al jugador 5, el jugador 2 al 6, y así sucesivamente hasta el jugador 4 que se enfrenta al 8. Este esquema asegura que, desde el comienzo, todos los participantes tienen la oportunidad de competir con un oponente del otro grupo.

A partir del día siguiente, procedemos a rotar los valores de los jugadores del segundo grupo ((n/2)+1 a n).

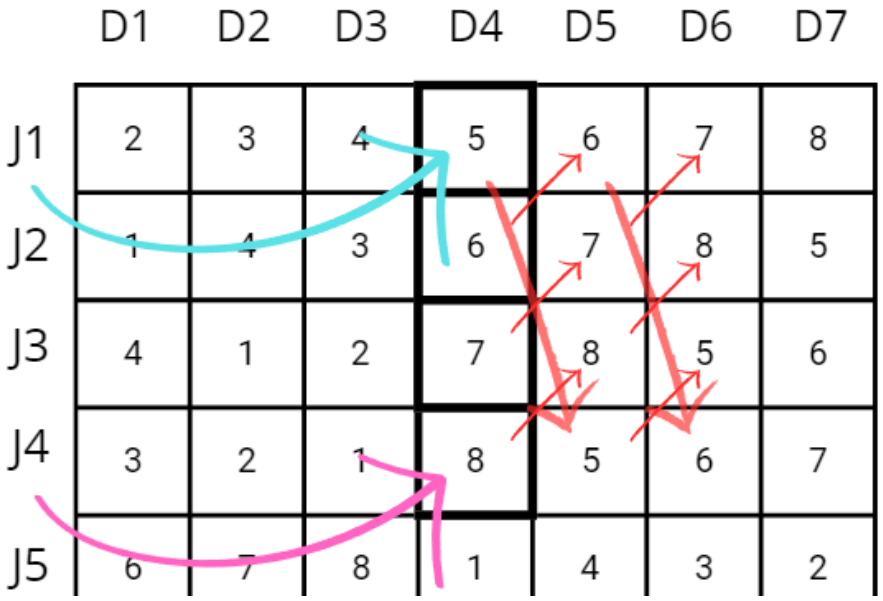

Imagen 4: Operaciones de DyV para el caso 1.

3.1.1.3 Cuadrante inferior derecho.

Para rellenar el cuadrante inferior derecho se ha usado la siguiente fórmula para rellenar sin tener que hacer rotaciones, empezamos rellenando en j5 desde d4 a d7 usando la fórmula que se muestra en la imagen, como se observa en la posición donde J5 y D4 se juntan, la fórmula nos dice que como el jugador es mayor que el día, hacemos jug-dia que es igual a 1, en la siguiente el jugador no es mayor que el día, así que hacemos jug + (n/2) - dia → 5 + (4) - 5 = 4. De esta forma vamos rellenando todos los días desde n/2 hasta n-1 para cada jugador desde (n/2) +1 hasta n.

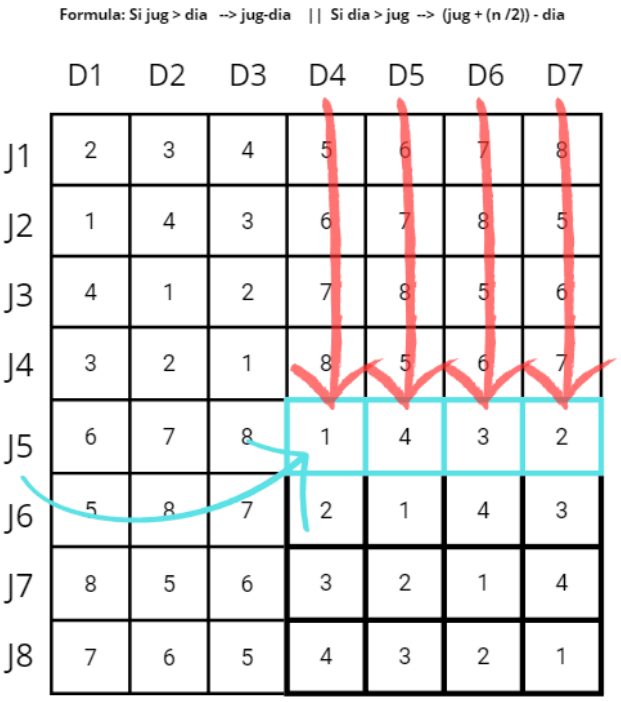

Imagen 5: Operaciones de DyV para el caso 1.

3.1.2 Caso 2.

Para cualquier N se aplica una estrategia similar al anterior, adaptándose para tratar con un número par de jugadores que no es una potencia de dos, para los impares se añade un jugador ficticio para hacer el número de jugadores par y se aplica el método como si fuera un caso par, eliminando posteriormente al jugador ficticio de la solución. A lo largo de la explicación se muestran imágenes para ayudar a entender cómo se están formando las tablas, aunque en las imágenes no aparece, hemos respetado tanto la columna 0, como la fila 0 para rellenar ambas de ceros y así acercarnos más al ejemplo de la práctica.

Pseudocódigo Caso 2

Función torneoRecursivoCaso2(tabla, n)
  // Caso Base: Solo dos jugadores
  Si n == 2
  // Se enfrentan directamente entre sí
  tabla[1][1] = 2

```
    tabla[2][1] = 1
  // Caso de n impar: Añadir un jugador ficticio
  Sino Si n es impar
    // Aplicar el algoritmo como si fuera par, añadiendo un jugador ficticio
    Llamar a torneoRecursivoCaso2 con n+1
    // Eliminar al jugador ficticio después de organizar los enfrentamientos
  // Caso Recursivo: n par y más de dos jugadores
  Sino
    // Aplicar el algoritmo a cada mitad y ajustar enfrentamientos entre mitades
  Fin Si
Fin Función
```

Tabla 2: Pseudocódigo caso 2.

En el caso dos la forma en la que se rellenará la matriz seguirá la siguiente distribución, coincidiendo con la forma en la que se rellenará en el caso 1.

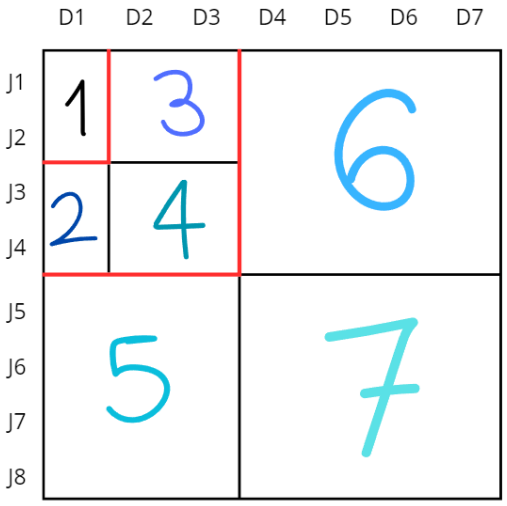

Imagen 6: División de la matriz para el caso 1.

3.1.2.1 Caso 2 (Par).

3.1.2.1.1 Caso m es par.

Para el Cuadrante Inferior Izquierdo y el Cuadrante Superior Izquierdo para el Caso 2 con (m) par y (m) impar.

3.1.2.1.1.1 Cuadrante Superior Izquierdo.

Este cuadrante se encarga de organizar los enfrentamientos entre los jugadores de la primera mitad del torneo (1 a (m)) durante los primeros (m-1) días. Se sigue el método recursivo para determinar los enfrentamientos, asegurando que cada jugador se enfrente a todos los demás jugadores de su grupo al menos una vez. La organización de los juegos en este cuadrante se basa en la división del torneo en subtorneos más pequeños, aplicando la misma lógica de organización hasta alcanzar el caso base de dos jugadores.

Para los jugadores de 1 a (m), el torneo se organiza de manera que cada jugador tiene un juego programado cada día, sin repetir oponentes, hasta que todos hayan jugado contra cada uno de los otros jugadores de la primera mitad.
La asignación de los oponentes y la programación de los juegos se realiza mediante la llamada recursiva al algoritmo, adaptando el tamaño del problema a la mitad de los jugadores en cada iteración.

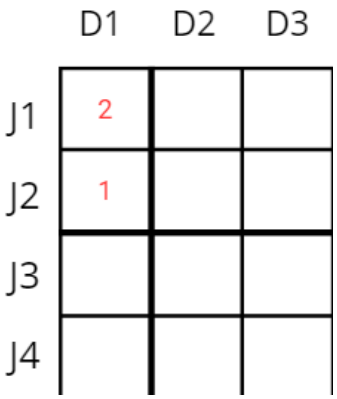

Imagen 7: Operaciones de DyV para el caso 2.

3.1.2.1.1.2 Cuadrante Inferior Izquierdo.

El cuadrante inferior izquierdo se ocupa de los enfrentamientos entre los jugadores de la segunda mitad ((m+1) a (n)) y los de la primera mitad (1 a (m)), pero con un enfoque que se inicia desde el primer día hasta el día (m-1). La asignación de los enfrentamientos sigue una regla simple que garantiza que todos los jugadores de la segunda mitad se enfrenten a todos los de la primera mitad sin repeticiones.

Se suma el valor (m) (la mitad del número total de jugadores) a los valores del cuadrante superior izquierdo para cada jugador en la segunda mitad ((m+1) a (n)). Esto significa que para cada día de juego, los jugadores de la segunda mitad se enfrentarán a un oponente distinto de la primera mitad.

La suma de (m) a los valores del cuadrante superior izquierdo efectivamente "desplaza" a los jugadores de la segunda mitad para enfrentarse a los de la primera mitad, utilizando el patrón establecido en el cuadrante superior izquierdo como base para los enfrentamientos.

3.1.2.1.1.3 Cuadrante Superior Derecho.

En este cuadrante, se organizan los encuentros entre los jugadores de la primera mitad del torneo (1 a (m)) y los de la segunda mitad (m+1) a (n)), comenzando en el día posterior a (m) y siguiendo hasta el final del torneo. La lógica de asignación de juegos es la siguiente:

Para cada jugador de la primera mitad (1 a (m)), se enfrenta a un oponente de la segunda mitad ((m+1) a (n)) en cada día, que va de (m) a (n-1).

Si jug + dia <= n, el oponente del jugador jug en el día día se calcula con jug + dia.

Si jug + dia > n, se ha completado una rotación completa de enfrentamientos, y para evitar repeticiones, el oponente se recalcula con jug + dia - m.

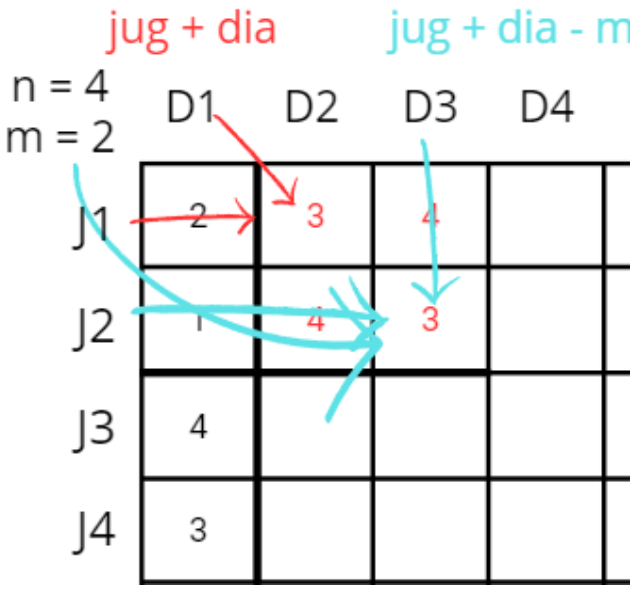

Imagen 8: Operaciones de DyV para el caso 2.

3.1.2.1.1.4 Cuadrante Inferior Derecho.

En el cuadrante inferior derecho, los jugadores de la segunda mitad ((m+1) a (n)) se enfrentan a los de la primera mitad (1 a (m)) en un esquema que promueve un encuentro único entre cada par de jugadores a lo largo del torneo. Se sigue una lógica específica para asignar los enfrentamientos.

Para cada jugador jug de la segunda mitad ((m+1) a (n)), y para cada día , también desde (m) a (n-1), se determina el oponente de la primera mitad.

Si jug - dia >= 1, el jugador jug se enfrenta a un oponente calculado restando jug - dia.

Si jug - dia < 1, para asegurar que todos los jugadores se enfrenten sin repetir oponentes, el oponente se determina con (jug + m) - dia.

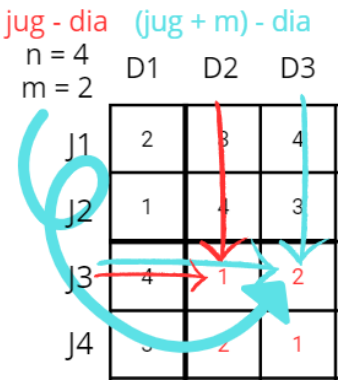

Imagen 9: Operaciones de DyV para el caso 2.

3.1.2.1.2 Caso m es impar.

3.1.2.1.2.1 Cuadrante inferior izquierdo.

Se sigue la misma regla de sumar m a los valores del cuadrante superior izquierdo, pero se deben considerar los valores 0 (descanso del jugador ficticio) para asegurar que se mantenga la estructura del torneo. Es decir, al igual que las tres primeras filas (jugadores) se han formado usando una fila extra llena de ceros para hacer los descansos de los enfrentamientos, esto hace básicamente lo mismo que en cuadrante superior izquierdo pero en el inferior izquierdo.

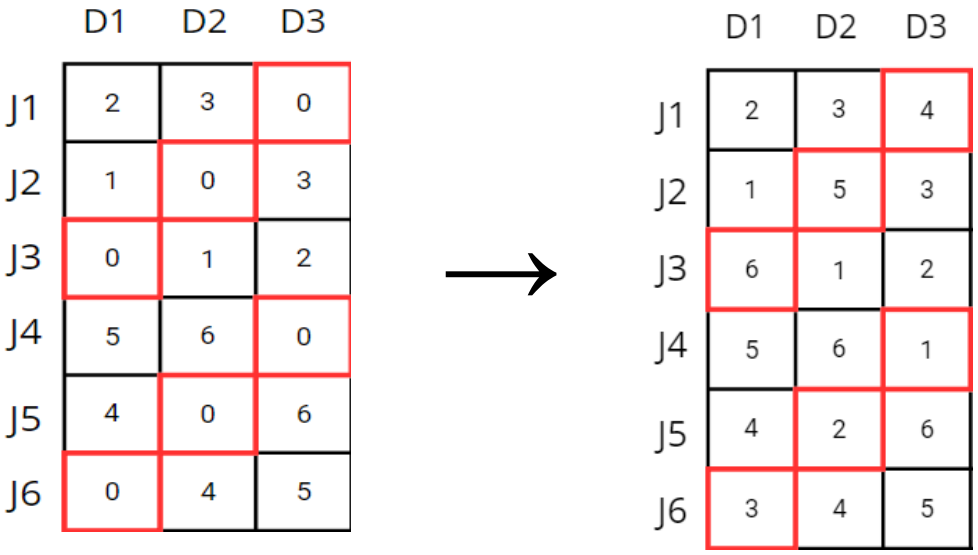

Imagen 10: Operaciones de DyV para el caso 2.

3.1.2.1.2.2 Cuadrante Superior Derecho.

Este cuadrante coordina los encuentros entre los jugadores de la primera mitad (1 a (m)) y los de la segunda mitad ((m+1) a (n)), en días que siguen al día (m). La iteración hace lo siguiente.

- Si (jug + dia) <= n, el jugador jug aún no ha jugado contra todos en la segunda mitad, entonces el oponente se asigna sumando jug + dia.

- Si (jug + dia) > n, se ha completado una ronda de juegos, y para continuar sin repetir oponentes, se ajusta la asignación haciendo jug + dia - m.

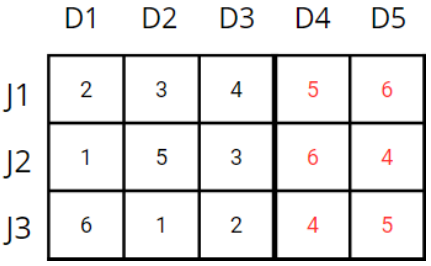

Imagen 11: Operaciones de DyV para el caso 2.

3.1.2.1.2.3 Cuadrante Inferior Derecho.

Este cuadrante organiza los juegos de la segunda mitad contra la primera, usando una lógica que evita repeticiones y asegura que todos jueguen entre sí.

- Si jug > dia, indica que el jugador de la segunda mitad ya enfrentó a los de la primera mitad señalados por día, entonces se calcula un nuevo oponente haciendo jug - dia.

- Si jug <= dia, para mantener la rotación sin repeticiones, hacemos (jug + m) - dia.

	D1	D2	D3	D4	D5
J1	2	3	4	5	6
J2	1	5	3	6	4
J3	6	1	2	4	5
J4	5	6	1	3	2
J5	4	2	6	1	3
J6	3	4	5	2	1

Imagen 12: Operaciones de DyV para el caso 2.

3.1.2.2 Caso 2 (Impar).

Para la resolución de una n impar en el caso2, simplemente la matriz se rellena de forma normal siguiendo la forma de crear la matriz con una n par, es decir, si n es impar se crea una matriz como si se resolviera el caso de n+1. Después de esto, el jugador nuevo (jugador n+1) rellena todos sus enfrentamientos con ceros, además todos los enfrentamientos del resto de jugadores con ese jugador nuevo también se ponen a cero. Por último, se elimina esta última fila con sólo ceros y el resultado nos da la matriz con n=3.

3.1.3 Caso 3.

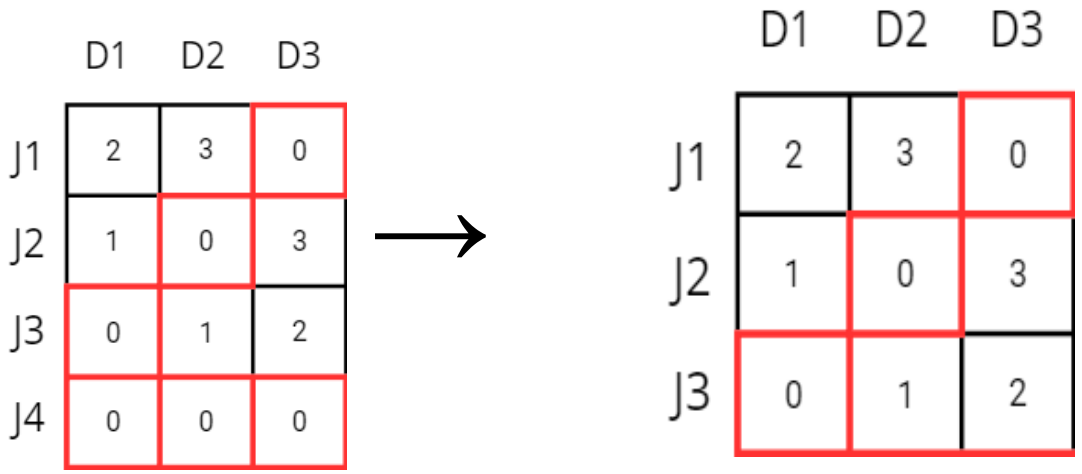

Imagen 13: Operaciones de DyV para el caso 2.

Para N potencia de dos sin simplificación en este caso, el número de jugadores corresponde exactamente a una potencia de dos, lo que permite una organización del torneo altamente estructurada y eficiente.

En lugar de incorporar ajustes o jugadores ficticios, esta implementación aprovecha la simetría y las propiedades matemáticas de las potencias de dos para planificar los emparejamientos.

Pseudocódigo Caso 3

```
Función torneoRecursivoCaso3(tabla, i, j)
        // Caso Base: Solo dos jugadores restantes para enfrentarse
            Si j == i + 1
                    // Programar el enfrentamiento directo entre ellos
                     tabla[0][i - 1] = j
                     tabla[0][j - 1] = i
                Sino // Caso Recursivo: Más de dos jugadores
  // Encuentra el punto medio para dividir a los jugadores en dos grupos
                    k = i + ((j - i + 1) / 2)
  // Aplica el algoritmo recursivamente a cada grupo
                        Llamar a torneoRecursivoCaso3 para cada mitad
```

```
    // Después de organizar enfrentamientos internos en cada grupo,
    // organiza los enfrentamientos entre grupos para los días restantes
                    Fin Si
Fin Función
```

Tabla 3: Pseudocódigo caso 3.

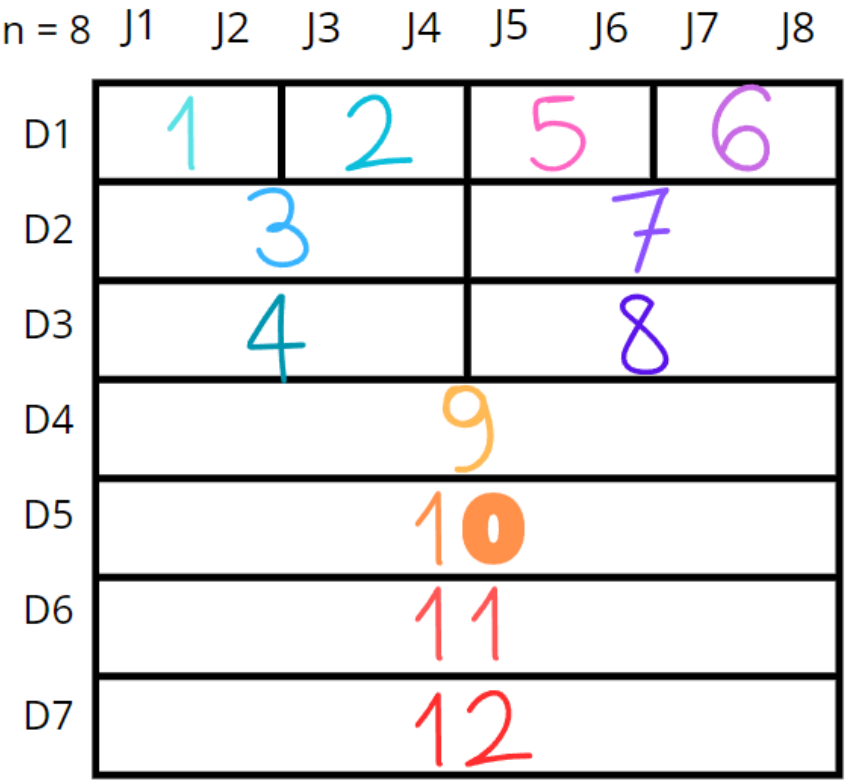

Imagen 14: Operaciones de DyV para el caso 2.

En el Caso 3, organiza un torneo para un número de jugadores (n) que es una potencia de dos, empleando un enfoque recursivo para determinar los enfrentamientos de cada día. A diferencia de los otros casos, este enfoque se centra en maximizar la utilización de los días de juego, permitiendo que el torneo se complete en (n-1) días.

3.1.3.1 Caso Base.

Cuando solo quedan dos jugadores por enfrentarse (j = i + 1)), se asigna directamente el enfrentamiento entre ellos. Esto se registra en la tabla de emparejamientos indicando que el jugador (i) juega contra el jugador (j) y viceversa. Este es el punto de terminación de la recursividad, donde se define claramente el resultado de un enfrentamiento directo.

3.1.3.1 Caso Recursivo.

Para un grupo de (n) jugadores, el proceso se divide en fases.

División en Subgrupos:
Se calcula el punto medio (k) para dividir a los jugadores en dos grupos, de (i) a (k-1) y de (k) a (j), permitiendo que el torneo se organiza en subgrupos más pequeños.

Recursión y Organización de Enfrentamientos:
La función se llama a sí misma recursivamente para cada uno de los dos subgrupos, organizando los enfrentamientos dentro de esos grupos.
Una vez que se han determinado los enfrentamientos internos de los subgrupos, se organiza la segunda fase de enfrentamientos cruzados entre los grupos para los días restantes. Aquí es donde se utiliza la variable (dia) para programar estos enfrentamientos adicionales.

Programación de Enfrentamientos Cruzados:
Para cada día adicional ((dia = (n / 2) + m)), se asignan enfrentamientos cruzados entre los dos subgrupos, utilizando un patrón que asegura que todos los jugadores se enfrenten entre sí sin repetir oponentes.

Los enfrentamientos se determinan basándose en la posición relativa de los jugadores dentro de sus respectivos subgrupos y el día del torneo.3.2 Estudio teórico.
El análisis teórico se centra en la eficiencia de los algoritmos utilizados, examinando su complejidad temporal y espacial. Se exploraron las características del método Divide y Vencerás, incluyendo cómo la división del problema en subproblemas menores contribuye a una solución más eficiente.

3.2 Estudio teórico.

3.2.1 Análisis general de los tiempos de ejecución en DyV.

Se realiza la división del problema hasta llegar al caso base, donde la recurrencia es del tipo:

$$t(n) = \begin{cases} h(n) & Caso\ base \\ a \cdot t\left(\frac{n}{b}\right) + g(n) & Casos\ recursivos \end{cases}$$

Fórmula 1: Esquema recursivo general en DyV.

- t (n) → Tiempo de ejecución.
- h (n) → Coste de resolver el problema base (casos triviales).
- g (n) → Coste de dividir el problema en subproblemas y de combinar las soluciones parciales de éstos.
- a → Número de subproblemas que se generan.
- b → Factor de división.

Las funciones h(n) y g(n) no son recursivas y suelen ser polinómicas.

$$t(n) = \begin{cases} c_1 \cdot n^k & 0 \leq n < b \ (Caso\ base) \\ a \cdot t\left(\frac{n}{b}\right) + c_2 \cdot n^k & n \geq b \ (Casos\ recursivos) \end{cases}$$

Fórmula 2: Fórmula de recursividad general en DyV.

Con lo que la fórmula maestra sería:

$$t(n) \in \begin{cases} \Theta\left(n^k\right) & a < b^k \\ \Theta\left(n^k \cdot log_b\, n\right) & a = b^k \\ \Theta\left(n^{log_b\, a}\right) & a > b^k \end{cases}$$

Fórmula 3: Fórmula maestra general en DyV.

Para la implementación que se ha realizado de modo general tenemos:

- **Caso base:** O(1) = n^0
  n^0 → k1 = 0

- **Caso recursivo:** 2t (n/2) + n^2
  2t → a = 2
  n/2 → b = 2
  n^2 → k2 = 2

Donde K → máx (k1, k2) = 2; y por lo tanto a < b^k que es 2 < 2^2; es decir el coste algorítmico será de O(n^k) que para este caso en concreto será O(n^2).

El proceso de análisis algorítmico ha sido realizado linea a linea de dentro a fuera, con lo que se obtiene el coste total de T(n), al que tras aplicar el teorema maestro debemos obtener un orden O(n^2), con lo que la función crece cuadráticamente en función del tamaño de entrada de n.

3.2.2 Caso 1.

1. La declaración de la función carece de coste luego es O(0).
   public static void torneoRecursivoCaso1(int[][] tabla, int n) {

2. El caso base se ejecuta cuando n es igual a 2, en el que las operaciones dentro de este bloque se ejecutan en tiempo constante, por lo que el coste es O(1).
   if (n == 2) {
   tabla[1][1] = 2;
   tabla[2][1] = 1;

```
}
```

3. La llamada recursiva se llama a sí misma una vez, con lo que tiene un coste de **T(n/2)**.
   torneoRecursivoCaso1(tabla, n / 2);

4. Este bucle anidado recorre la mitad de los elementos de tabla, donde el bucle externo se ejecuta n/2 veces y el bucle interno se ejecuta (n/2) - 1 veces y por lo tanto, el coste de este bloque es O((n/2) * (n/2)) = **O(n^2/4)**.

```
for (int jug = (n / 2) + 1; jug <= n; jug++) {
        for (int dia = 1; dia <= (n / 2) - 1; dia++) {
                tabla[jug][dia] = tabla[jug - (n / 2)][dia] + (n / 2);
        }
}
```

5. Para el cuadrante superior derecho y el inferior derecho tenemos una estructura similar al anterior pero con un If-Else, en el que el coste final de ambos es **O(n^2 /4)**.

```
// Cuadrante superior derecho
        for (int jug = 1; jug <= (n / 2); jug++) {
                for (int dia = (n / 2); dia <= n - 1; dia++) {
                        if (jug + dia <= n) {
                                tabla[jug][dia] = jug + dia;
                        } else {
                                tabla[jug][dia] = jug + dia - (n / 2);
                        }
                }
        }

// Cuadrante inferior derecho
        for (int jug = (n / 2) + 1; jug <= n; jug++) {
                for (int dia = (n / 2); dia <= n - 1; dia++) {
                        if (jug > dia) {
                                tabla[jug][dia] = jug - dia;
                        } else {
                                tabla[jug][dia] = (jug + (n / 2)) - dia;
                        }

                }
        }
```

6. Coste total de la función.
   El coste total es **T(n) = T(n/2) + 3 * O (n^2 /4**).
   Haciendo uso del teorema maestro descrito anteriormente para ecuaciones de recurrencia, podemos simplificar esto a **O(n^2)**.

3.2.3 Caso 2.

1. En caso base en todos los casos será de coste **O(1)**.

```
if (n == 2) {
        tabla[1][1] = 2;
        tabla[2][1] = 1;
}
```

2. Esta función se llama a sí misma una vez, con lo que el coste es **T(n+1)**.
```
else if (n % 2 != 0) {
        torneoRecursivoCaso2(tabla, n + 1);
```

3. El bucle externo se ejecuta n veces y el bucle interno se ejecuta n veces, luego el coste de este bloque es O(n * n) = **O(n^2)**.
```
for (int jug = 1; jug <= n; jug++) {
        for (int dia = 1; dia <= n; dia++) {
                if (tabla[jug][dia] == n + 1)
                        tabla[jug][dia] = 0;
        }
}
```

4. Esta es otra llamada recursiva a la misma función, pero con n / 2, como esta función se llama a sí misma una vez se tiene un coste de **T(n/2)**.
```
else {
        int m = n / 2;
        torneoRecursivoCaso2(tabla, m);
```

5. Coste total de la función.
   Los siguientes bloques de código tienen una estructura similar a los anteriores, por lo que también tienen un coste de **O(n²)** cada uno.

   Por lo tanto, el coste total de es **T(n) = T(n+1) + T(n/2) + 4 * O(n²)**.

   Finalmente dado que la función se llama a sí misma dos veces con diferentes tamaños de entrada y tiene un bucle anidado que se ejecuta en tiempo cuadrático, el coste algorítmico de esta función es crece cuadráticamente con el tamaño de la entrada n, luego el coste es **O(n^2)**.

3.2.4 Caso 3.

1. En caso base las operaciones dentro de este bloque se ejecutan en tiempo constante, por lo que el coste es **O(1).**
```
if (j == i + 1) {
        tabla[0][i - 1] = j;
        tabla[0][j - 1] = i;
}
```

2. Esta línea tiene un coste constante de **O(1)**.
```
int k = i + ((j - i + 1) / 2);
```

3. Estas son dos llamadas recursivas a la misma función, pero con diferentes rangos de i a j. Como esta función se llama a sí misma dos veces el coste es de **2T(n/2)**.
   torneoRecursivoCaso3(tabla, i, k - 1);
   torneoRecursivoCaso3(tabla, k, j);

4. Este es bucle anidado se ejecuta n/2 veces y el bucle interno se ejecuta n/2 veces, por lo tanto, el coste de este bloque es O((n/2) * (n/2)) = O(n^2 /4).

```
for (int m = 0; m < n / 2; m++) {
        int dia = (n / 2) + m;
        tabla[dia - 1][i - 1] = k + m;
        tabla[dia - 1][k + m - 1] = i;
        for (int l = 1; l < n / 2; l++) {
                int oponente = k + ((l + m) % (n / 2));
                tabla[dia - 1][i + l - 1] = oponente;
                tabla[dia - 1][oponente - 1] = i + l;
        }
}
```

5. Coste total de la función.
   El coste total es **T(n) = 2T(n/2) + O(n^2 /4).**.
   Finalmente de igual modo que en los anteriores con el teorema maestro obtenemos un coste de **O(n^2)**.

3.3 Estudio experimental.

Se llevaron a cabo pruebas experimentales para validar el rendimiento y la correlación de los algoritmos implementados. Se recopilaron datos sobre el tiempo de ejecución y la memoria utilizada, analizando cómo estos factores varían con el número de participantes en el torneo.

El orden de ejecución será del cálculo temporal, el espacial y por último el cuadro de cruces para diferentes entradas.

Destacar que entradas no válidas serán descartadas por el programa y finalizará la ejecución del mismo.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: -512
Opción no válida. Por favor, ingrese 1, 2 o 3.
```

Imagen 15: Salida por consola para entrada de números negativos.

De igual modo si se introducen caracteres no numéricos también se lanza un error y se detiene la ejecución, esto se aplica a todas las entradas que se solicitan por consola.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: tenis
Error en la entrada. Por favor, ingrese un número válido.
```

Imagen 16: Salida por consola para entrada de caracteres alfabéticos.

3.3.1 Caso 1.

3.3.1.1 Cálculo temporal para una entrada máxima de N = 16384.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 1
Introduzca el número de jugadores: 16384

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 1
N       Tiempo(ns)
2       1020
4       2690
8       2200
16      16400
32      51560
64      283200
128     144940
256     444020
512     1250800
1024    2920530
2048    3176310
4096    12976030
8192    50027790
16384   190957460
```

Imagen 17: Salida por consola para entrada de N = 16384 en cálculo temporal.

3.3.1.2 Cálculo espacial para una entrada de N = 16384.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 2
Introduzca el número de jugadores: 16384

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 1
N       Espacio(MB)
2       2.288818359375E-5 MB
4       7.62939453125E-5 MB
8       2.74658203125E-4 MB
16      0.00103759765625 MB
32      0.0040283203125 MB
64      0.015869140625 MB
128     0.06298828125 MB
256     0.2509765625 MB
512     1.001953125 MB
1024    4.00390625 MB
2048    16.0078125 MB
4096    64.015625 MB
8192    256.03125 MB
16384   1024.0625 MB
```

Imagen 18: Salida por consola para entrada de N = 16384 en cálculo espacial.

3.3.1.3 Cuadro de cruces para N = 4.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 4

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 1

El cuadro de cruces es:

  |d1 d2 d3
------------
j1| 2  3  4
j2| 1  4  3
j3| 4  1  2
j4| 3  2  1
```

Imagen 19: Salida por consola del cuadro de cruces para entrada de N = 4.

3.3.1.4 Cuadro de cruces para N = 8.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 8

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 1

El cuadro de cruces es:

  |d1 d2 d3 d4 d5 d6 d7
-----------------------
j1| 2  3  4  5  6  7  8
j2| 1  4  3  6  7  8  5
j3| 4  1  2  7  8  5  6
j4| 3  2  1  8  5  6  7
j5| 6  7  8  1  4  3  2
j6| 5  8  7  2  1  4  3
j7| 8  5  6  3  2  1  4
j8| 7  6  5  4  3  2  1
```

Imagen 20: Salida por consola del cuadro de cruces para entrada de N = 8.

3.3.1.5 Cuadro de cruces para N = 16.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 16

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 1

El cuadro de cruces es:
```

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10	d11	d12	d13	d14	d15
j1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
j2	1	4	3	6	7	8	5	10	11	12	13	14	15	16	9
j3	4	1	2	7	8	5	6	11	12	13	14	15	16	9	10
j4	3	2	1	8	5	6	7	12	13	14	15	16	9	10	11
j5	6	7	8	1	4	3	2	13	14	15	16	9	10	11	12
j6	5	8	7	2	1	4	3	14	15	16	9	10	11	12	13
j7	8	5	6	3	2	1	4	15	16	9	10	11	12	13	14
j8	7	6	5	4	3	2	1	16	9	10	11	12	13	14	15
j9	10	11	12	13	14	15	16	1	8	7	6	5	4	3	2
j10	9	12	11	14	15	16	13	2	1	8	7	6	5	4	3
j11	12	9	10	15	16	13	14	3	2	1	8	7	6	5	4
j12	11	10	9	16	13	14	15	4	3	2	1	8	7	6	5
j13	14	15	16	9	12	11	10	5	4	3	2	1	8	7	6
j14	13	16	15	10	9	12	11	6	5	4	3	2	1	8	7
j15	16	13	14	11	10	9	12	7	6	5	4	3	2	1	8
j16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Imagen 21: Salida por consola del cuadro de cruces para entrada de N = 16.

3.3.2 Caso 2.

3.3.2.1 Caso 2 (Par).

En este caso la entrada es N = 16386 para la toma de datos ya que al ser un número par que no es potencia de 2 similar en tamaño a los datos introducidos anteriormente el programa accede a este caso.

3.3.2.1.1 Cálculo temporal para una entrada de N = 16386.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 1
Introduzca el número de jugadores: 16386

El número de jugadores es par.
Potencias de 2.
N       Tiempo(ns)
2       590
4       800
8       1900
16      7030
32      45030
64      125370
128     161060
256     374400
512     1348730
1024    1925490
2048    3482910
4096    14083660
8192    52235910
16384   211864510
```

Imagen 22: Salida por consola para entrada de N = 16386 en cálculo temporal con potencias de 2 (Mejor caso).

3.3.2.1.2 Cálculo espacial para una entrada máxima de N = 16386.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 2
Introduzca el número de jugadores: 16386

El número de jugadores es par.
2       2.288818359375E-5 MB
4       7.62939453125E-5 MB
8       2.74658203125E-4 MB
16      0.00103759765625 MB
32      0.0040283203125 MB
64      0.015869140625 MB
128     0.06298828125 MB
256     0.2509765625 MB
512     1.001953125 MB
1024    4.00390625 MB
2048    16.0078125 MB
4096    64.015625 MB
8192    256.03125 MB
16384   1024.0625 MB
```

Imagen 23: Salida por consola para entrada de N = 16386 en cálculo espacial.

3.3.2.1.3 Cuadro de cruces para N = 6.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 6

El número de jugadores es par.

El cuadro de cruces es:

  |d1 d2 d3 d4 d5
------------------
j1| 2  3  4  5  6
j2| 1  5  3  6  4
j3| 6  1  2  4  5
j4| 5  6  1  3  2
j5| 4  2  6  1  3
j6| 3  4  5  2  1
```

Imagen 24: Salida por consola del cuadro de cruces para entrada de N = 6.

3.3.2.1.4 Cuadro de cruces para N = 10.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 10

El número de jugadores es par.

El cuadro de cruces es:

   | d1  d2  d3  d4  d5  d6  d7  d8  d9
-------------------------------------
j1 |  2   3   4   5   6   7   8   9  10
j2 |  1   5   3   7   4   8   9  10   6
j3 |  8   1   2   4   5   9  10   6   7
j4 |  5   9   1   3   2  10   6   7   8
j5 |  4   2  10   1   3   6   7   8   9
j6 |  7   8   9  10   1   5   4   3   2
j7 |  6  10   8   2   9   1   5   4   3
j8 |  3   6   7   9  10   2   1   5   4
j9 | 10   4   6   8   7   3   2   1   5
j10|  9   7   5   6   8   4   3   2   1
```

Imagen 25: Salida por consola del cuadro de cruces para entrada de N = 10.

3.3.2.1.5 Cuadro de cruces para N = 14.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 14

El número de jugadores es par.

El cuadro de cruces es:

   | d1  d2  d3  d4  d5  d6  d7  d8  d9 d10 d11 d12 d13
-------------------------------------------------------
j1 |  2   3   4   5   6   7   8   9  10  11  12  13  14
j2 |  1   4   3   6   7   9   5  10  11  12  13  14   8
j3 |  4   1   2   7  10   5   6  11  12  13  14   8   9
j4 |  3   2   1  11   5   6   7  12  13  14   8   9  10
j5 |  6   7  12   1   4   3   2  13  14   8   9  10  11
j6 |  5  13   7   2   1   4   3  14   8   9  10  11  12
j7 | 14   5   6   3   2   1   4   8   9  10  11  12  13
j8 |  9  10  11  12  13  14   1   7   6   5   4   3   2
j9 |  8  11  10  13  14   2  12   1   7   6   5   4   3
j10| 11   8   9  14   3  12  13   2   1   7   6   5   4
j11| 10   9   8   4  12  13  14   3   2   1   7   6   5
j12| 13  14   5   8  11  10   9   4   3   2   1   7   6
j13| 12   6  14   9   8  11  10   5   4   3   2   1   7
j14|  7  12  13  10   9   8  11   6   5   4   3   2   1
```

Imagen 26: Salida por consola del cuadro de cruces para entrada de N = 14.

3.3.3.2 Caso 2 (Impar).

Para el caso impar al igual que con el par se selecciona una entrada similar grande pero impar, cabe destacar en este caso que se va a trabajar con números primos ya que es el peor caso en el que se puede ejecutar el programa, es por ello que se toman datos diferentes a potencias de 2.

3.3.3.2.1 Cálculo temporal para una entrada de N = 16385.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 1
Introduzca el número de jugadores: 16385

El número de jugadores es impar.

Números primos.
N       Tiempo(ns)
3       3000
7       3530
17      24610
37      106130
79      455550
163     624410
331     1144820
673     4393940
1361    3796600
2729    11017860
5471    30794740
10949   153990330
```

Imagen 27: Salida por consola para entrada de N = 16385 en cálculo temporal con números primos (Peor caso).

3.3.3.2.2 Cálculo espacial para una entrada de N = 16385.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 2
Introduzca el número de jugadores: 16385

El número de jugadores es impar.
3       7.62939453125E-5 MB
7       2.74658203125E-4 MB
17      0.00130462646484375 MB
37      0.00565338134765625 MB
79      0.02471923828125 MB
163     0.1032257080078125 MB
331     0.4217376708984375 MB
673     1.7354965209960938 MB
1361    7.081626892089844 MB
2729    28.44097137451172 MB
5471    114.2435302734375 MB
10949   457.43350982666016 MB
```

Imagen 28: Salida por consola para entrada de N = 16385 en cálculo espacial.

3.3.3.2.3 Cuadro de cruces para N = 5.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 5

El número de jugadores es impar.

El cuadro de cruces es:

  |d1  d2  d3  d4  d5
----------------------
j1|2   3   4   5   0
j2|1   5   3   0   4
j3|0   1   2   4   5
j4|5   0   1   3   2
j5|4   2   0   1   3
```

Imagen 29: Salida por consola del cuadro de cruces para entrada de N = 5.

3.3.3.2.4 Cuadro de cruces para N = 9.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 9

El número de jugadores es impar.

El cuadro de cruces es:

  |d1  d2  d3  d4  d5  d6  d7  d8  d9
-------------------------------------
j1|2   3   4   5   6   7   8   9   0
j2|1   5   3   7   4   8   9   0   6
j3|8   1   2   4   5   9   0   6   7
j4|5   9   1   3   2   0   6   7   8
j5|4   2   0   1   3   6   7   8   9
j6|7   8   9   0   1   5   4   3   2
j7|6   0   8   2   9   1   5   4   3
j8|3   6   7   9   0   2   1   5   4
j9|0   4   6   8   7   3   2   1   5
```

Imagen 30: Salida por consola del cuadro de cruces para entrada de N = 9.

3.3.3.2.5 Cuadro de cruces para N = 13.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 13

El número de jugadores es impar.

El cuadro de cruces es:

   |d1   d2   d3   d4   d5   d6   d7   d8   d9   d10  d11  d12  d13
-----------------------------------------------------------------------
j1 |2    3    4    5    6    7    8    9    10   11   12   13   0
j2 |1    4    3    6    7    9    5    10   11   12   13   0    8
j3 |4    1    2    7    10   5    6    11   12   13   0    8    9
j4 |3    2    1    11   5    6    7    12   13   0    8    9    10
j5 |6    7    12   1    4    3    2    13   0    8    9    10   11
j6 |5    13   7    2    1    4    3    0    8    9    10   11   12
j7 |0    5    6    3    2    1    4    8    9    10   11   12   13
j8 |9    10   11   12   13   0    1    7    6    5    4    3    2
j9 |8    11   10   13   0    2    12   1    7    6    5    4    3
j10|11   8    9    0    3    12   13   2    1    7    6    5    4
j11|10   9    8    4    12   13   0    3    2    1    7    6    5
j12|13   0    5    8    11   10   9    4    3    2    1    7    6
j13|12   6    0    9    8    11   10   5    4    3    2    1    7
```

Imagen 31: Salida por consola del cuadro de cruces para entrada de N = 13.

3.3.4 Caso 3.

3.3.4.1 Cálculo temporal para una entrada de N = 16384.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 1
Introduzca el número de jugadores: 16384

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 2
N       Tiempo(ns)
2       710
4       630
8       1710
16      9890
32      68210
64      176260
128     70370
256     259920
512     796310
1024    2655160
2048    7769550
4096    30992220
8192    124165170
16384   491393230
```

Imagen 32: Salida por consola para entrada de N = 16384 en cálculo temporal.

3.3.4.2 Cálculo espacial para una entrada máxima de N = 16384.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 2
Introduzca el número de jugadores: 16384

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 2
2       7.62939453125E-6 MB
4       4.57763671875E-5 MB
8       2.13623046875E-4 MB
16      9.1552734375E-4 MB
32      0.0037841796875 MB
64      0.015380859375 MB
128     0.06201171875 MB
256     0.2490234375 MB
512     0.998046875 MB
1024    3.99609375 MB
2048    15.9921875 MB
4096    63.984375 MB
8192    255.96875 MB
16384   1023.9375 MB
```

Imagen 33: Salida por consola para entrada de N = 16384 en cálculo espacial.

3.3.4.3 Cuadro de cruces para N = 4.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 4

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 2

El cuadro de cruces es:

  |j1  j2  j3  j4
------------------
d1|2   1   4   3
d2|3   4   1   2
d3|4   3   2   1
```

Imagen 34: Salida por consola del cuadro de cruces para entrada de N = 4.

3.3.4.4 Cuadro de cruces para N = 8.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 8

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 2

El cuadro de cruces es:

  |j1  j2  j3  j4  j5  j6  j7  j8
-----------------------------------
d1|2   1   4   3   6   5   8   7
d2|3   4   1   2   7   8   5   6
d3|4   3   2   1   8   7   6   5
d4|5   6   7   8   1   2   3   4
d5|6   7   8   5   4   1   2   3
d6|7   8   5   6   3   4   1   2
d7|8   5   6   7   2   3   4   1
```

Imagen 35: Salida por consola del cuadro de cruces para entrada de N = 8.

3.3.4.5 Cuadro de cruces para N = 16.

```
TORNEO DE TENIS

SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Cálculo tiempo de respuesta.
OPCIÓN 2: Estudio memoria principal.
OPCIÓN 3: Imprimir cuadro de cruces.

OPCIÓN SELECCIONADA: 3
Introduzca el número de jugadores: 16

El número de jugadores es potencia de 2.
SELECCIONE UNA OPCIÓN:
OPCIÓN 1: Caso 1.
OPCIÓN 2: Caso 3.

OPCIÓN SELECCIONADA: 2

El cuadro de cruces es:
```

	j1	j2	j3	j4	j5	j6	j7	j8	j9	j10	j11	j12	j13	j14	j15	j16
d1	2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15
d2	3	4	1	2	7	8	5	6	11	12	9	10	15	16	13	14
d3	4	3	2	1	8	7	6	5	12	11	10	9	16	15	14	13
d4	5	6	7	8	1	2	3	4	13	14	15	16	9	10	11	12
d5	6	7	8	5	4	1	2	3	14	15	16	13	12	9	10	11
d6	7	8	5	6	3	4	1	2	15	16	13	14	11	12	9	10
d7	8	5	6	7	2	3	4	1	16	13	14	15	10	11	12	9
d8	9	10	11	12	13	14	15	16	1	2	3	4	5	6	7	8
d9	10	11	12	13	14	15	16	9	8	1	2	3	4	5	6	7
d10	11	12	13	14	15	16	9	10	7	8	1	2	3	4	5	6
d11	12	13	14	15	16	9	10	11	6	7	8	1	2	3	4	5
d12	13	14	15	16	9	10	11	12	5	6	7	8	1	2	3	4
d13	14	15	16	9	10	11	12	13	4	5	6	7	8	1	2	3
d14	15	16	9	10	11	12	13	14	3	4	5	6	7	8	1	2
d15	16	9	10	11	12	13	14	15	2	3	4	5	6	7	8	1

Imagen 36: Salida por consola del cuadro de cruces para entrada de N = 16.

3.3.5 Conclusiones.

Se puede observar que se obtienen salidas limpias y satisfactorias en todo momento con cabeceras para la correcta identificación de lo que está haciendo el programa.

Respecto a las salidas de los test vemos que hay una correlación completa respecto a lo esperado, con ligeras diferencias que atienden al equipo con el que se toman los datos, también en el uso de memoria ya que eclipse necesitará para su uso una pequeña parte, pero que es despreciable respecto a lo calculado previamente.

4.- Anexo 1.

4.1 Diseño del código.

En este diagrama ilustrativo del procedimiento que sigue en línea general el programa se aprecia cómo se comprueban los datos introducidos para no generar problemas en la ejecución, también se determina según la opción introducida que se va a realizar, con ello automáticamente se selecciona el caso, por ejemplo si se selecciona imprimir el cuadro de cruces y se da una entrada N = 7, se llama al caso 2 se realizan los cálculos necesarios y se muestra por pantalla con la cabecera correspondiente. Para n = Potencia de 2, el programa solicita si se desea aplicar el caso 1 o el caso 3.

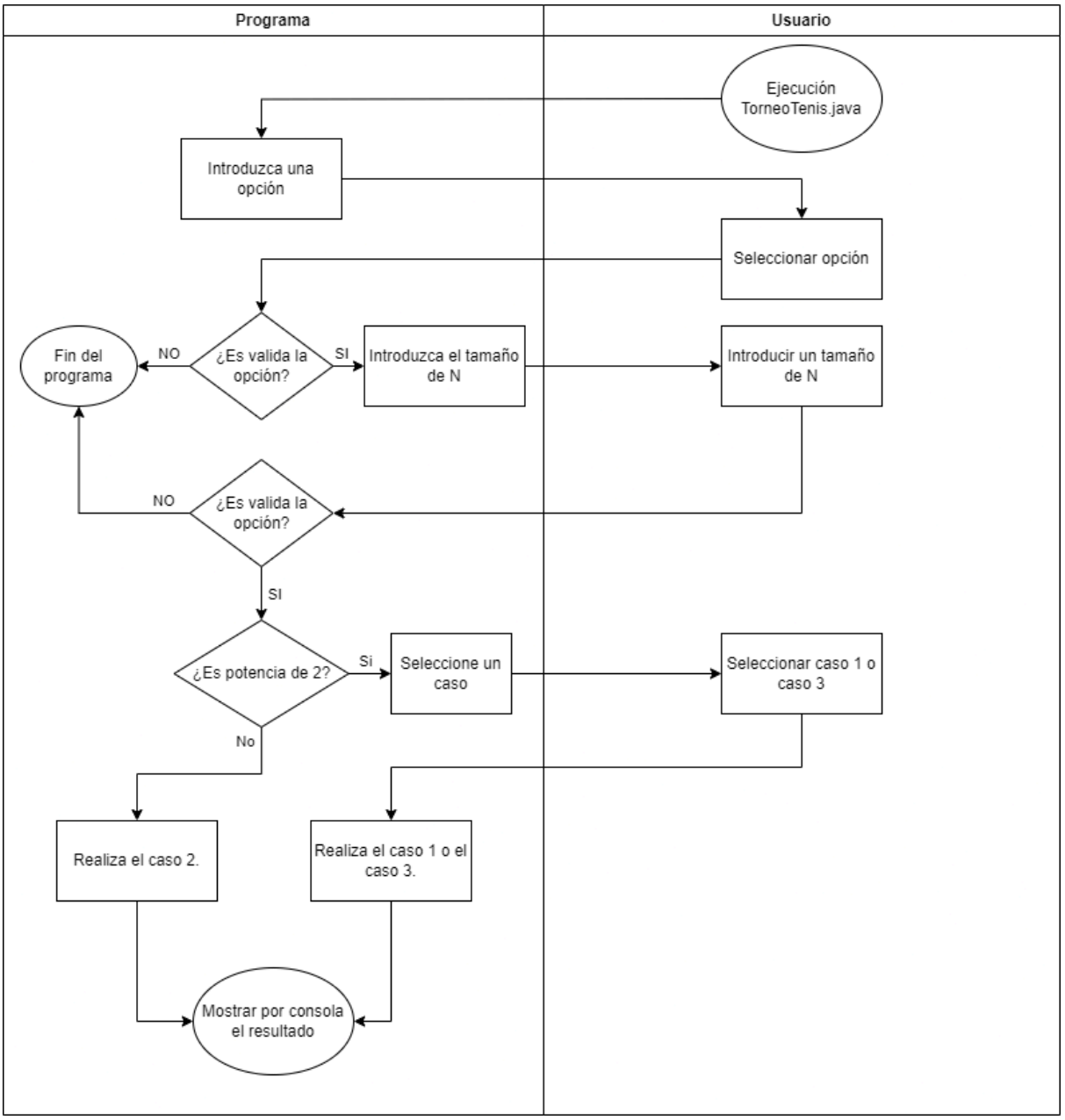

Diagrama 1: Actuación general del sistema con el usuario.

4.2 Diagramas de clase.

4.2.1 TorneoTenis.

Se ha realizado todo el programa en una única clase, ya que se decidió en equipo que al ser la primera vez que trabajabamos juntos mantener esta clase en una única clase era una opción satisfactoria para el problema que enfrentamos.

Diagrama 2: Diagrama de clases del TorneoTenis.

4.2.1.1 ImprimirTablaCaso1.

Este método imprime por pantalla el cuadro de cruces para N = Potencia de 2 del caso 1 y cuando N = Número par del caso 2.

4.2.1.2 ImprimirTablaCaso2.

En este método se imprime de manera exclusiva el cuadro de cruces del caso 2 cuando N = Número impar.

4.2.1.3 ImprimirTablaCaso3.

La tabla de cruces del caso 3 es diferente al caso uno por lo que este método opera de otra manera para imprimir el resultado de manera satisfactoria.

4.2.1.4 TorneoRecursivoCaso1.

Este método realiza los cálculos para N = Potencia de 2 del caso 1.

4.2.1.5 TorneoRecursivoCaso2.

Este método realiza los cálculos para N = Par e Impar, como se ha explicado en el estudio de implementación.

4.2.1.6 TorneoRecursivoCaso3.

Este método realiza los cálculos para N = Potencia de 2 del Caso 3.

4.2.1.7 EsPrimo.

Calcula si un número es primo o no.

4.2.1.8 ArrayList<Integer> generarPrimos.

Este método guarda en un arrayList números primos, haciendo uso del método EsPrimo, de modo que partiendo de 2, multiplica por 2 el número inicial, y comprobando si es primo guardará en el array el siguiente número primo, volviendolo a duplicar hasta el límite establecido que es la entrada de N.

4.2.1.9 UsoMemoria.

Este método se usa para calcular la complejidad espacial de la matriz que se genera, calculando el tamaño en bytes de una matriz bidimensional de enteros, donde *Integer.SIZE / 8.0* convierte el tamaño en bits de un entero en bytes y devuelve el tamaño en bits de un entero en la plataforma actual generalmente 32 bits. Al dividir por 8, se convierte de bits a bytes.

4.2.1.10 Main.

Se realizan todas las llamadas necesarias a los métodos para calcular las diferentes opciones así como métodos necesarios en la confección de las matrices, también posee los algoritmos de la toma de datos para el test temporal, este método toma una muestra del trabajo en equipo realizado, ya que la diversidad en la programación de cada integrante hace que se usen metodologías diferentes a la hora de implementar las funciones usadas.

4.2.2 TorneoTenisTest.

Este test ha sido proporcionado por parte del profesorado para comprobar el resultado de la matriz generada en el caso 1.

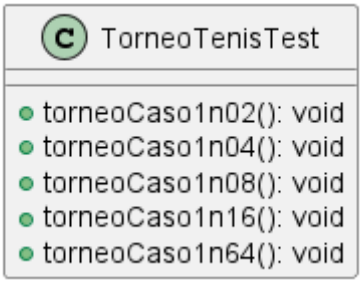

Diagrama 3: Diagrama de clases del TorneoTenisTest.

La ejecución del test muestra que todo se generó de manera correcta, con lo que este caso se da por válido en su implementación.

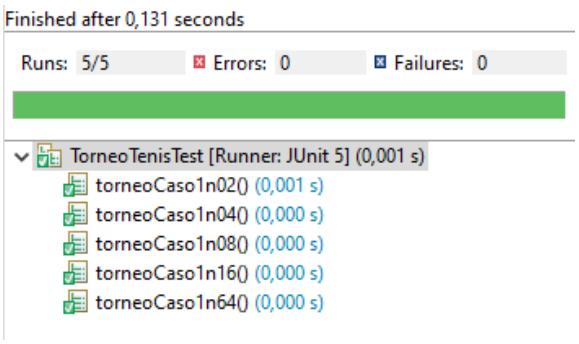

Imagen 37: Resultado de la prueba en eclipse de TorneoTenisTest.

4.2.3 TorneoTenisTestCaso3.

Este test se ha generado para comprobar el caso 3, ya que al operar de manera diferente a los demás casos se requería comprobar sus matrices resultantes.

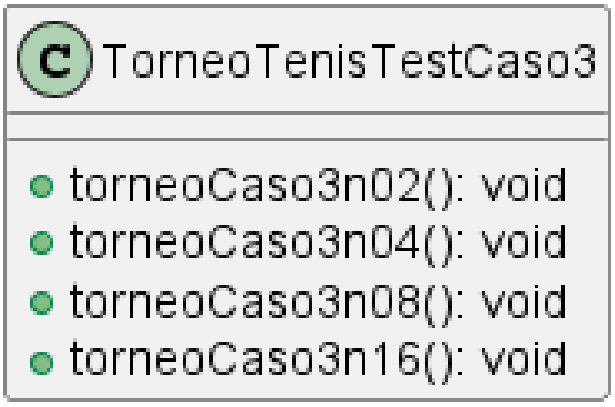

Diagrama 2: Diagrama de clases del TorneoTenisTestCaso3.

La ejecución de este test también es satisfactoria con lo que ya se determina que su implementación es correcta.

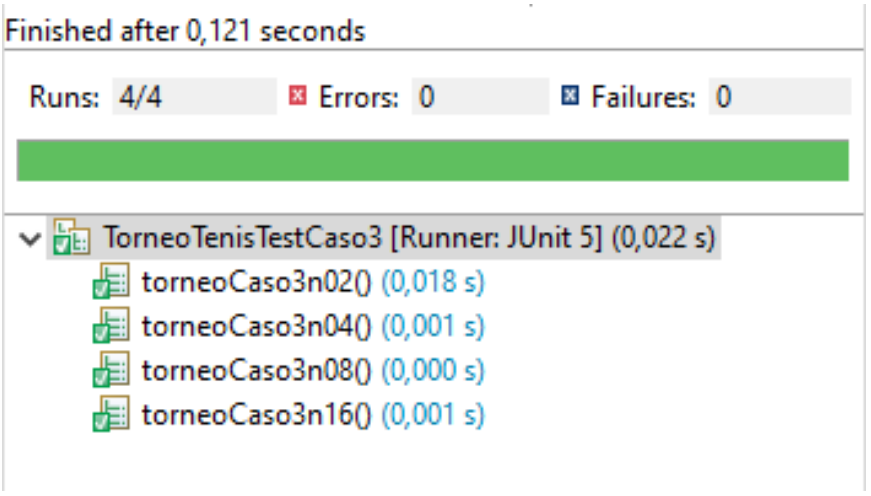

Imagen 38: Resultado de la prueba en eclipse de TorneoTenisTestCaso3.

4.3 Listado de archivos fuente.

En esta imagen se aprecia como para el funcionamiento del programa se usa exclusivamente una clase y hay dos Juegos de prueba o test para comprobar el funcionamiento del programa.

No se ha requerido para el caso 2 un test ya que en ningún momento este caso generó problemas en su salida.

Destacar que se ha seguido según lo especificado en el guión la nomenclatura indicada para la creación de paquetes y clases, los demás documentos son generados por eclipse para el funcionamiento del sistema, por ultimo destacar que en la carpeta docs se encuentran la memoria, un diagrama utilizado para la explicación del funcionamiento del programa y un excel que contiene los resultados tomados en el estudio experimental.

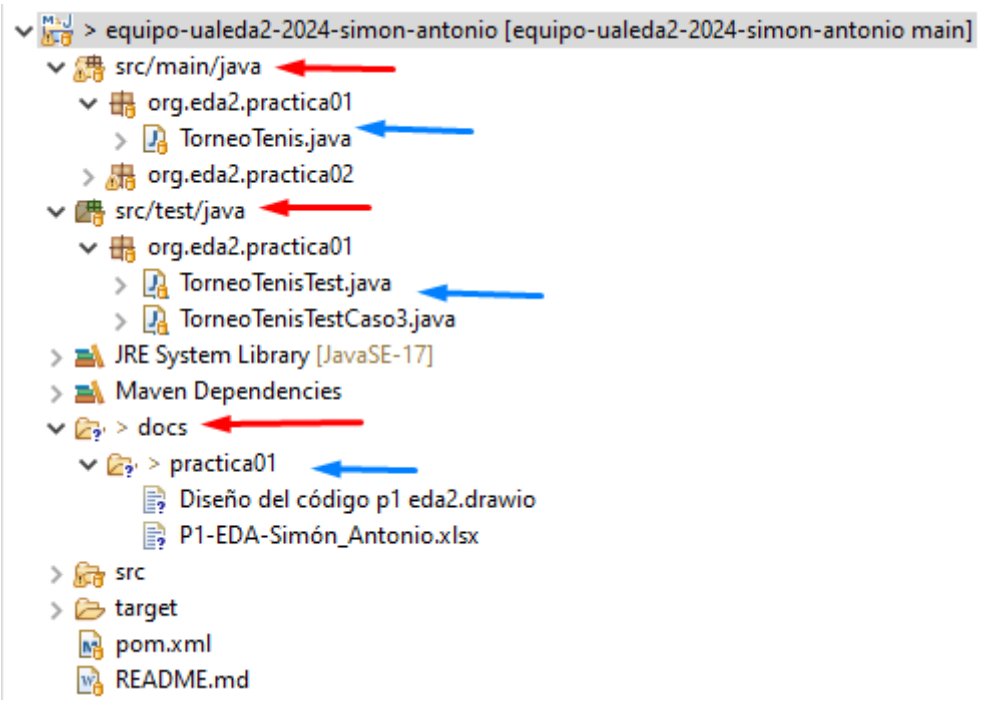

Imagen 39: Listado de archivos fuente para la práctica 1 Torneo de tenis.

5.- Anexo 2.

5.1 Cálculos realizados.

Se presentan gráficos que muestran los resultados obtenidos, junto con la correlación de los datos y la línea de tendencia.

Para la **toma de datos temporales** de manera general, lo que se realiza es un bucle en el que se va aumentando la entrada de la matriz, se toma el tiempo de inicio y a continuación en un bucle se toman en 10 ocasiones los datos temporales, el tiempo de respuesta es el inicial menos el final dividido por 10, de modo que se toman los datos en el menor tiempo posible sin posibles interferencias del sistema.

Pseudocódigo cálculo de tiempo de procesamiento

Imprimir "N\tTiempo(ns)"
Para nEntrada = 2 hasta n (incrementando nEntrada multiplicado por 2 en cada iteración):
  Crear una matriz tablaTest de tamaño (nEntrada + 1) x nEntrada
  inicio = Obtener tiempo actual en nanosegundos

```
    Para i = 0 hasta 10:
        Llamar a la función torneoRecursivoCaso1 con tablaTest y nEntrada como
parámetros

    fin = Obtener tiempo actual en nanosegundos
    tiempoPromedio = (fin - inicio) / 10  // Calculando el tiempo promedio de las 10
ejecuciones
    Imprimir nEntrada y tiempoPromedio separados por un tabulador
```

Tabla 4: Pseudocódigo procedimiento general cálculo del tiempo de procesamiento.

Para la **toma de datos espacial** se llama al método *UsoMemoiria* , que dependiendo del tamaño de la matriz generará diferentes resultados, en el que un int 32 bits = 4 bytes y 2^14 * 2^14 * 2^2 = 2^30 bytes = 1GB que es el resultado que obtenemos en MB que es 1024, con lo que 2^15*2^15*2^2 = 4GB que sería el resultado de una entrada de N = 32768.

En la **representación gráfica de los resultados** se usa un gráfico de dispersión donde se sitúa una línea de tendencia de tipo polinómica de grado 2 y el eje vertical se encuentra a una escala de 1*10^6 ya que el tiempo es tomado en nanosegundos.

5.1.1 Caso 1.

En el caso 1 se observa que la correlación es perfecta ya que los resultados siguen a la perfección la línea de tendencia del algoritmo de orden N^2.

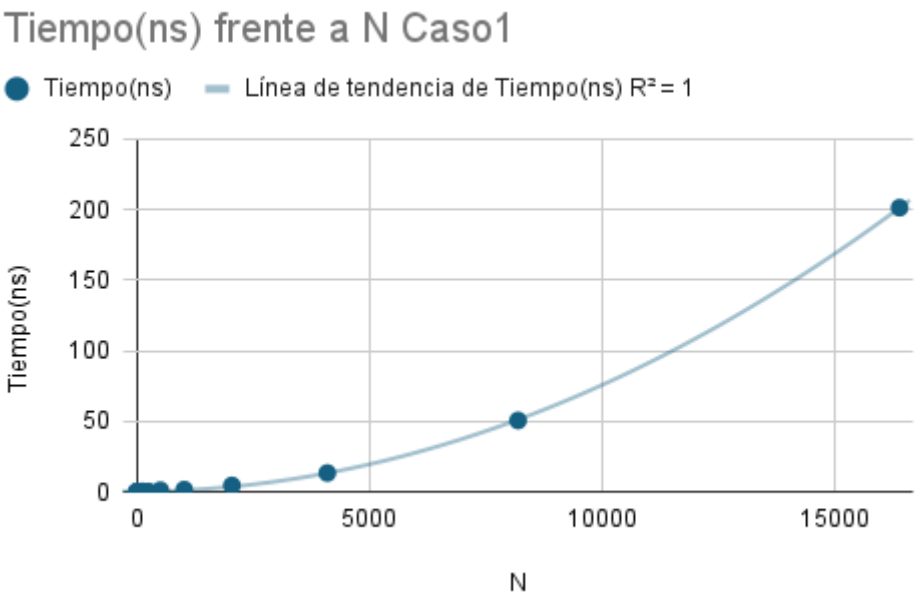

Imagen 40: Tiempo en ns frente a la entrada n para el caso 1 con línea de tendencia cuadrática.

5.1.2 Caso 2.

De igual modo que en el caso 1 la correlación es perfecta para números pares e impares, destacar que los pares usan el mejor caso que son las potencias de 2, mientras que los impares usan el peor caso que son los números primos, igualmente se dan los resultados por satisfactorios.

5.1.2.1 Caso 2 (Par).

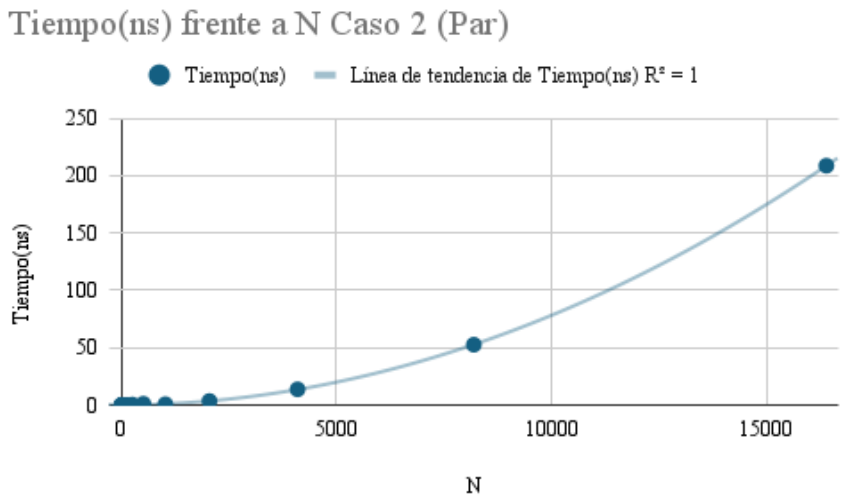

Imagen 41: Tiempo en ns frente a la entrada n para el caso 2 par con línea de tendencia cuadrática.

5.1.2.2 Caso 2 (Impar).

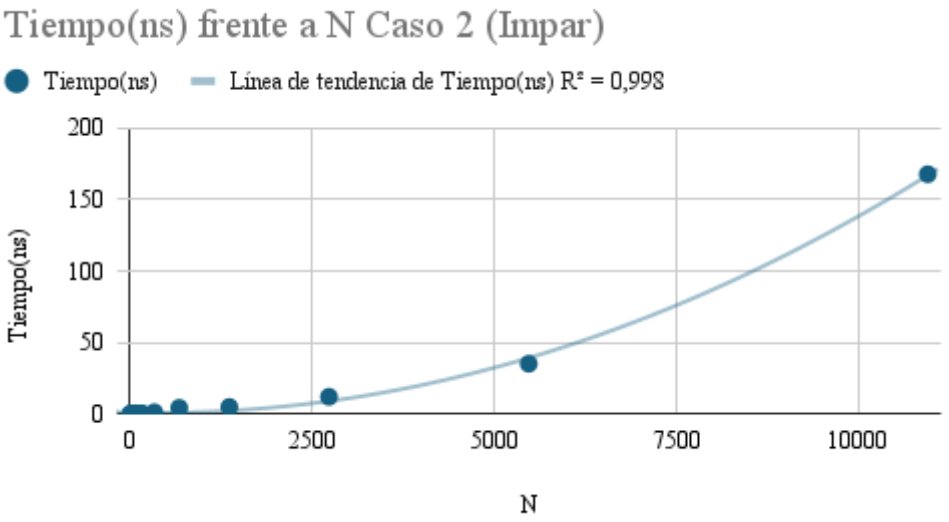

Imagen 42: Tiempo en ns frente a la entrada n para el caso 2 impar con línea de tendencia cuadrática.

5.1.3 Caso 3.

Finalmente el caso 3 sigue con la misma fidelidad la línea de tendencia, con lo que queda demostrado que la toma de datos ha sido adecuada y que hay una correlación directa entre los cálculos esperados teóricamente y los experimentales.

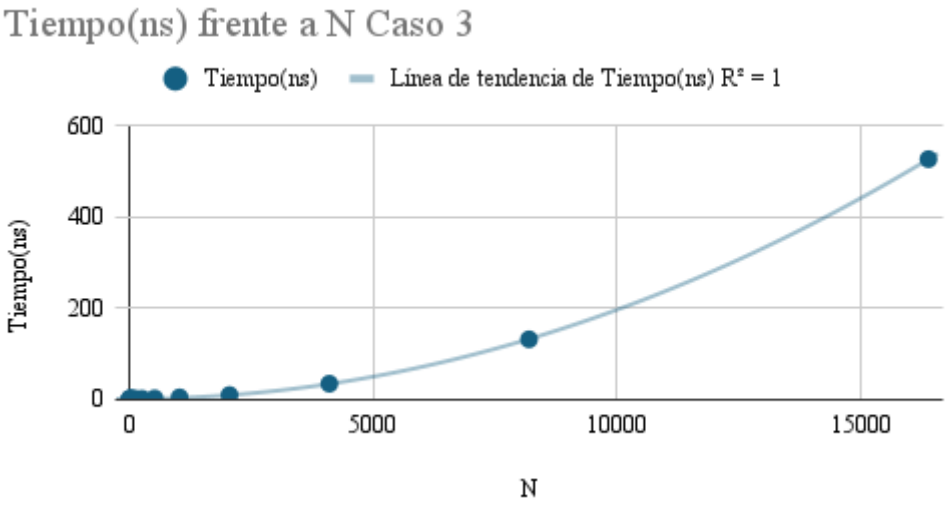

Imagen 43: Tiempo en ns frente a la entrada n para el caso 3 con línea de tendencia cuadrática.

5.1 Conclusiones finales.

En el desarrollo de esta práctica se aprecia que el método de divide y vencerás es muy practico y util ya que dividir el problema en partes más simples hace que el coste algorítmico no llegue a crecer de manera exponencial, y se mantenga en un coste de O (n^2), esta correlación entre el estudio teórico y el experimental da muestra de un buen análisis en la implementación del problema.

Como puntos a mejorar la modularidad en el programa podría ser mayor, de modo que existiera por ejemplo un único método para imprimir las tablas en función de la entrada fuera de la clase principal por si se requiere en un futuro para otro problema la impresión de arrays multidimensionales, también los métodos para calcular el cuadro de cruces se podría mejorar ligeramente ya que al final el coste no disminuiría mucho.

6.- Bibliografía.

- UNE 157001:2014 Criterios generales para la elaboración formal. . . (s. f.). https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma?c=N0052985
- Scribbr. (s. f.). Guía rápida de cómo citar en APA según su 7a edición. https://www.scribbr.es/category/normas-apa/
- Universidad de Cantabria. Programación II. Divide y Vencerás 06_DyV.fm (unican.es)
- Apuntes de clase Estructura de datos y algoritmos 2, Universidad de Almería.
- Apuntes de clase Lógica y algoritmica, Universidad de Almería.