Soluciones – práctico 6

Divide & Vencerás (Divide & Conquer)

Ejercicio 5 (I)
Una forma de modelar el fixture del torneo es mediante una matriz con una fila por cada jugador y una columna por día de torneo. En cada celda de la matriz, se colocará al jugador con el que se debe enfrentar un jugador dado en un día determinado.

A continuación se muestra un fixture de 7 días en el cual participan 8 jugadores:

Jugador/día	día 1	día 2	día 3	día 4	día 5	día 6	día 7
0							
1							
2							
3							
4							
5							
6							
7							

Dado que se quiere aplicar la estrategia Divide y Vencerás para resolver este ejercicio, se debe buscar una forma de dividir este problema en sub-problemas (de tamaño menor al problema original).

Una forma de dividir el problema, es partir la tabla del torneo en dos y organizar los cruzamientos de dos "sub-torneos" por separado.

Para poder realizar esta partición se asume que ***n*** es **potencia de 2**.

Continuando con el ejemplo anterior, se divide el torneo original en dos sub-torneos:

Jugador/día	día 1	día 2	día 3
0			
1			
2			
3			

Jugador/día	día 1	día 2	día 3
4			
5			
6			
7			

Se continúa de esta forma hasta llegar a un problema básico, el cual consta de organizar un torneo de sólo dos jugadores y, por lo tanto, un partido.

Continuando con el ejemplo:

Jugador/día	día 1
0	1
1	0

Jugador/día	día 1
2	3
3	2

Jugador/día	día 1
4	5
5	4

Jugador/día	día 1
6	7
7	6

Al momento de juntar/combinar los resultados de la división del problema, se realiza lo siguiente:

1) En las primeras fechas se le asigna a cada jugador los cruzamientos dados para su sub-torneo.
2) Para el siguiente día se enfrenta cada jugador de un sub-torneo con un jugador del otro sub-torneo.
3) Luego, para cada jugador, se le asignan los cruzamientos que se le habían asignado, en el paso **1)**, al jugador con el que compitieron en la fecha de **2)**.

En el ejemplo:

Jugador/día	día 1	día 2	día 3
0	1	2	3
1	0	3	2
2	3	0	1
3	2	1	0

Jugador/día	día 1	día 2	día 3
4	5	6	7
5	4	7	6
6	7	4	5
7	6	5	4

Finalmente el fixture del torneo queda armado de la siguiente manera:

Jugador/día	día 1	día 2	día 3	día 4	día 5	día 6	día 7
0	1	2	3	4	5	6	7
1	0	3	2	5	4	7	6
2	3	0	1	6	7	4	5
3	2	1	0	7	6	5	4
4	5	6	7	0	1	2	3
5	4	7	6	1	0	3	2
6	7	4	5	2	3	0	1
7	6	5	4	3	2	1	0

A continuación se muestran los procedimientos que resuelven el problema planteado aplicando Divide & Conquer:

```
#include <iostream.h>
#include <stdio.h>
#include <stdlib.h>

// construye una tabla de dimensiones [cant, cant-1]
void crearTabla(int cant, int** & tabla){
	tabla = new int* [cant];

	for (int i = 0; i < cant; i++)
		tabla[i] = new int[cant-1];

	for (int i = 0; i < cant; i++)
		for(int j = 0; j < (cant-1); j++)
			tabla[i][j] = 0;
}

// destruye la tabla
void destruirTabla(int cant, int** & tabla){
	for (int i = 0; i < cant; i++)
		delete [] tabla[i];

	delete [] tabla;
}

// imprime el contenido de la tabla
void imprimirTabla(int cant, int** tabla){
	for (int i = 0; i < cant; i++){
		for(int j = 0; j < (cant-1); j++){
			cout<<tabla[i][j]<<"  ";
		}
		cout<<"\n";
	}
}
```

```
// dentro de la matriz tabla se copian los elementos que conforman
// una matriz con origen en (origen, 0) y tamaño [cant, cant-1]
// en otra matriz de origen (destino, cant).
void completar(int origen, int destino, int cant, int ** & tabla){
        for(int i = 0; i < cant; i++)
                for(int j = 0; j < (cant-1); j++)
                        tabla[destino+i][cant+j] = tabla[origen+i][j];
}

// genera los cruzamientos del torneo y los coloca en la
// tabla a partir de la posición [ini][0].
void generarCruzamientos(int ini,int cant, int **  &tabla){
        if(cant == 2){   // caso base, solo dos competidores
                tabla[ini][0]   = ini+1;
                tabla[ini+1][0] = ini;
        }
        else{
                // generar cruzamientos de los dos sub-torneos
                generarCruzamientos(ini, cant/2, tabla);
                generarCruzamientos(ini+cant/2, cant/2, tabla);

               // asignar al primer jugador del sub-torneo 1 el
               // primer jugador del sub-torneo 2 (y viceversa)
               // y así sucesivamente
               for(int i = ini; i < (ini+cant/2); i++){
                  tabla[i][(cant/2)-1] = i+cant/2;
                  tabla[i+cant/2][(cant/2)-1] = i;
               }

                // completar los cruzamientos de cada jugador con
                // los del jugador asignado en la parte anterior
                completar(ini, ini+cant/2, cant/2, tabla);
                completar(ini+cant/2, ini, cant/2, tabla);
        }
}

// imprime los cruzamientos de un torneo de cant participantes
void torneo(int cant){
        int ** tabla;
        crearTabla(cant, tabla);
        generarCruzamientos(0, cant, tabla);
        imprimirTabla(cant, tabla);
        destruirTabla(cant, tabla);
}
```