Tecnológico de Monterrey Campus Guadalajara

Diseño y análisis de algoritmos

Segundo Proyecto Parcial

Enrique Antonio Enciso De León A00367589
Antonio Reyes Espinoza A01224955

Introducción

En el siguiente documento se especifica el desarrollo de un programa capaz de resolver el problema de planeación de un torneo de fútbol, mediante el algoritmo de Round Robin de todos contra todos. El programa simplifica el proceso de tener que iterar manualmente las funciones para poder obtener el resultado requerido, siguiendo detalladamente un algoritmo que crece exponencialmente en su cantidad de iteraciones.

Descripción

Los distintos algoritmos de planificación son utilizados por muchas ramas dentro de las ciencias computacionales y en el exterior ya que son muy eficientes y utilizados en varias ramas ingenieriles por su metodología simple.
El proyecto consiste en realizar una implementación del algoritmo Round Robin antes mencionados para ofrecer una simplicidad mayor en su uso. La programación de dicho proyecto debe incluir una forma de interactividad con el usuario para que pueda especifiar el tamaño del problema, las opciones a desplegar incluyen elegir entre obtener como resultado una jornada simple y la jornada entera y aparte el programa de juegos

Diseño de la solución

El diseño del programa se basó principalmente en 6 clases, una clase main, la cual corre la interfaz principal con el usuario y llama todos los métodos necesarios según sea el caso del método elegido y sus distintos parámetros de ejecución, en esta clase también se llevan a cabo varias validaciones simples del sistema para reducir la probabilidad de que haya un error en la ejecución del mismo.

La clase Juego, es donde se crea la competencia entre equipos y los lineamientos para tener control sobre ellos, recibe como parámetro dos objetos Equipo y los lineamientos

de puntaje, para poder asignar a los equipos su respectivo puntaje y llamar a su función jugarPartido.

La clase Equipo es una clase desarrollada para poder resolver el dilema de cómo organizar los diferentes equipos que se vayan necesitando según la entrada del programa. los parámetros que recibe son aquellos que se le pasan después con el archivos de texto.

La clase Lector y Lector2 es la encargada de abrir y procesar los datos obtenidos mediante un archivo de texto, estos datos son procesados con un orden establecido formalmente para que se pueda jugar explícitamente de una buena manera y no ocurran inconsistencias en los datos.

La clase Competencia es la clase responsable de llamar a las clases Equipo y Juego para en conjunto con sus métodos y parámetros para ejecutar hasta el final todos los partidos de la temporada y a su vez guardar toda la información relacionada con los partidos para el archivo de jornadas.

La clase roundInter es una clase desarrollada para poder resolver mediante el algoritmo de Round Robin la planeación de los distintos partidos a ser jugados, en un tiempo específico. Todos ellos están representados con una matriz de números enteros positivos y se debe especificar las dimensiones de la misma para poder iniciar la simulación.

Se seleccionó el lenguaje de programación Python para implementar la estructura de clases anterior debido a su simplicidad y elegancia, además por la pericia de los programadores de salir del desarrollo con lenguajes en los que estaban implementando varios proyectos durante su momento.

Al inicio se planteó una solución con algoritmo recursivo que sonaba más simple y lógico en su planteamiento pero a la hora de realizar la implementación se llegó a la conclusión que aunque la lógica del pseudocódigo parecía convincente a fin de cuentas en la implementación resultaba poco útil debido a que no consideraba un caso del tercer cuadrante de la matriz de rondas.

Documentación del programa

Round Robin

```
def encuentraRonda(self,fila,columna,dimension):
        posibles = range(dimension)
        # remover posibles en la fila
        for i in range(columna):
            posibles.remove(self.tabla[fila][i])
        # remover posibles en las filas contruidas
        for i in range(fila):
            if self.tabla[i][columna] in posibles:
                posibles.remove(self.tabla[i][columna])

        for i in posibles:
            self.tabla[fila][columna]=i
            self.tabla[columna][fila]=i
            if (fila == dimension-2) and (columna == dimension-1):
                # Todo fue encontrado
                return 1
            if columna < dimension - 1:
                if self.encuentraRonda(fila,columna+1,dimension):
                    return 1
            elif fila < dimension - 2:
                if self.encuentraRonda(fila+1,fila+2,dimension):
                    return 1
            # deshacer el intento
            self.tabla[fila][columna]=0
            self.tabla[columna][fila]=0
```

Competencia

```
def jugarJornada(self):
                    self.arreglo = []
                    self.juegos =[]
                    for i in range(1,self.lec.nEquipos):
                            for j in range(0,self.lec.nEquipos):
                                    x = self.round.matrix[j][i]
                                    temp = Juego(self.equipos[j], self.equipos[x],
self.lec.pGanar, self.lec.pEmpatar, self.lec.pPerder)
                                    self.juegos.append([self.equipos[j].nombre,
self.equipos[x].nombre,temp.gol1,temp.gol2])

            for x in xrange(0,self.lec.nEquipos):
                    a = self.equipos[x]
```

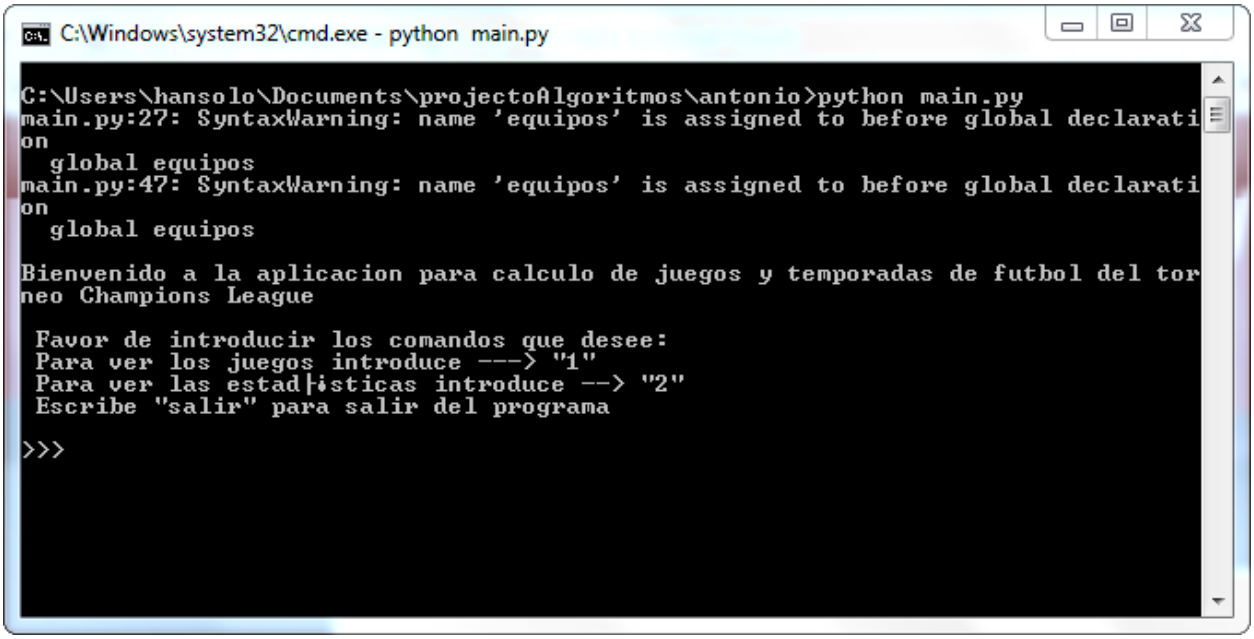
self.arreglo.append([a.nombre, a.pJugados, a.pGanados, a.pPerdidos, a.golFavor, a.golContra, a.difGoles, a.puntos])

Consideraciones de uso

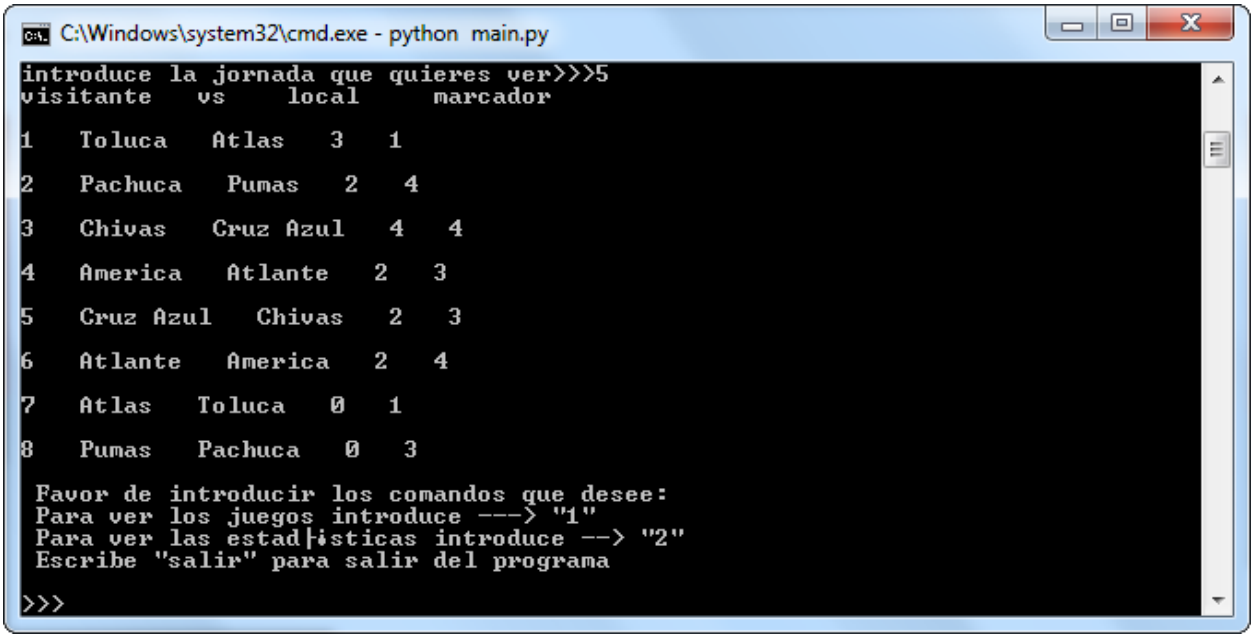
Las siguientes son las consideraciones necesarias que deben ser aplicadas para que el programa funcione de la manera más óptima.

1. El programa funciona como un archivo ejecutable “exe” en el sistema operativo Windows xp o superior.
2. El archivo de entrada para el programa debe ser un archivo de texto, extensión .txt, el cual no se puede encontrar vacío, con caracteres que no sean caracteres no especiales, o con cualquier otra circunstancia fuera de números separados por coma.
    a. Cada renglón del archivo representa según el orden de los requerimientos cada información necesaria para especificar la funcionalidad del programa
    b. El último renglón de los equipos necesita de ser ingresado con un formateo especial, el cual es nombre del equipo, separado por una coma y el peso del equipo que debe ir de 0 a 10
3. Aunque algunas validaciones están realizadas para la parte de interacción con el usuario se debe tomar en cuenta solo seleccionar los valores que se le indiquen y hacer las entradas lo más simple posible.
4. Las expresiones mostradas por la consola carecen de acentos y se espera que los valores de ingreso de los usuarios de igual manera no incluyen acentos ni caracteres especiales.

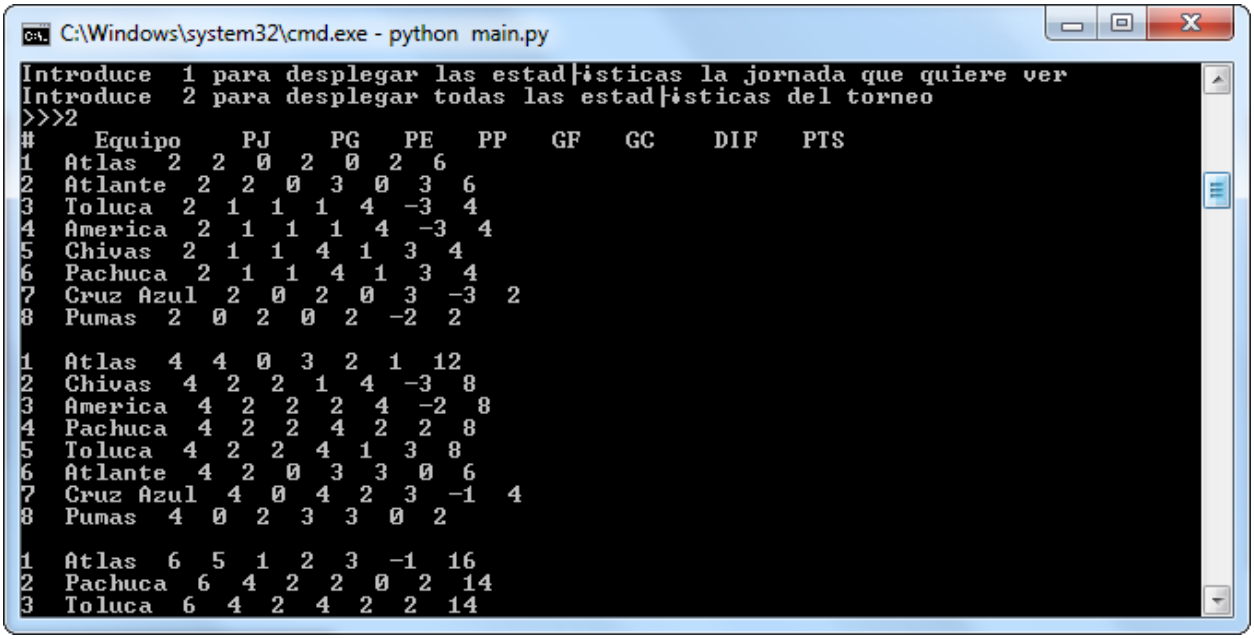
Casos de uso



Inico de pantalla



Despliegue de juegos por jornada



Despliege de estadísticas por jornadas

Conclusiones

Una vez terminados el diseño y la implementación del programa para desarrollar el algortimo Round Robin:
Definitivamente, con el poder de la computación podemos implementar los diversos algoritmos que darán solución de un problema, los algoritmos tienen una importancia muy grande y con el desarrollo tecnológico podemos obtener de manera rápida y eficiente los resultados de esos algoritmos.
En este caso,desarrollamos el diseño de una temporada de juegos de futbol. Con el poder computacional de las computadoras actuales, podemos obtener esos resultados de manera rápida solamente haciendo y diseñando el algoritmo e implementando el algoritmo.