PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 Programación Avanzada (I/2018)

Tarea 2

1. Entregas

- `Entregable - Obligatorio`
  - **Fecha/hora:** sábado 14 de abril del 2018, 23:59 horas.
  - **Lugar:** `https://goo.gl/forms/sGE0LxmHd7x4isTq2`
- `Tarea - Obligatoria`
  - **Fecha/hora:** domingo 22 de abril del 2018, 23:59 horas.
  - **Lugar:** GitHub – Carpeta: `Tareas/T02/`
- `README.md - Obligatorio`
  - **Fecha/hora:** lunes 23 de abril del 2018, 23:59 horas.
  - **Lugar:** GitHub – `Carpeta: Tareas/T02/`

2. Objetivos

- Crear y diseñar estructuras de datos personalizadas.
- Aplicar conceptos y nociones de estructuras de datos para el correcto modelamiento de un problema.
- Crear y usar algoritmos eficientes para recorrer las estructuras creadas.
- Aprender a ignorar archivos en Git mediante el uso del archivo `.gitignore`.
- Familiarizarse con el paquete de interfaz gráfica `PyQt5`.

Índice

3. Introducción

Luego de un pequeño error en el código de tu tarea, el universo virtual colapsa por lo que te ves resignado a volver al mundo real. Revisando tu aburrido correo, descubres un mensaje de un tal Pep "Ocho" Guardiola que dice lo siguiente:

> *"Hola tío, he sabido que estáis cursando el ramo de Programación avanzada y la verdad es que estoy bastante jodido con la organización de mi nuevo equipo de fútbol. ¿Podríais ayudarme a manejarlo?"*

Decides aceptar el desafío y desarrollar un programa capaz de gestionar las relaciones entre jugadores y el calendario de partidos de un equipo de fútbol, entre otras cosas.

4. *Programming Evolution Soccer* (PES)

4.1. Afinidad entre jugadores [40 %]

En tu programa los jugadores serán una parte vital para poder organizar tu equipo. Un jugador contará con los siguientes atributos: alias, nombre completo, nacionalidad, el club y liga a los que pertenecen por *default* (estos no cambian a través del programa) y el *overall* que es un atributo que va desde 0 a 99, donde 0 representa a un jugador muy malo y 99 a un excelente jugador de fútbol.

Por *cosas del fútbol*, se sabe que algunos jugadores juegan mejor entre ellos y otros no tanto. Gracias a una base de datos muy completa, cortesía de la FIFAE (claro, *Fédération Internationale de Football Association Électronique*), cuyos detalles se encuentran en la sección 4.4, es posible conocer información sobre cada jugador, como el club, su nacionalidad y la liga en donde juega. En base a estos tres factores, deberás calcular el nivel de **afinidad** entre dos jugadores. La afinidad es un número entre 0 y 1 donde:

- 1 punto de afinidad significa que son amigos cercanos
- 0,95 puntos de afinidad significa que son amigos lejanos
- 0,90 puntos de afinidad significa que son conocidos

La relación de afinidad entre cada par de jugadores se determina de acuerdo a las siguientes reglas:

- Si los jugadores comparten nacionalidad **y** club, entonces son amigos cercanos.
- Si los jugadores comparten nacionalidad **y** liga **y** no son amigos cercanos, entonces son amigos lejanos.
- Si los jugadores comparten nacionalidad **o** liga **o** club y no son amigos lejanos entonces son conocidos.
- Si los jugadores no comparten ni nacionalidad ni liga ni club, entonces decimos que no se conocen. El cálculo de la afinidad entre dos jugadores que no se conocen se detalla en la sección 4.1.1.
- Un jugador **no** tiene una relación de afinidad consigo mismo.

La figura 1 muestra un ejemplo de grafo construido en base a la relación de afinidad descrita (grafo de afinidad) con un universo de 20 jugadores. Es importante notar que el grafo de afinidad en el ejemplo es conexo[1] al igual que el que usarán en la tarea. Esto significa que todos los jugadores están conectados a todos los otros jugadores, ya sea directamente, o a través de otros jugadores. También es importante saber que el grafo de afinidad no debiera contener bucles[2]; es decir, ningún jugador está conectado consigo mismo.

[1] `https://es.wikipedia.org/wiki/Grafo_conexo`
[2] `https://es.wikipedia.org/wiki/Bucle_(teoria_de_grafos)`

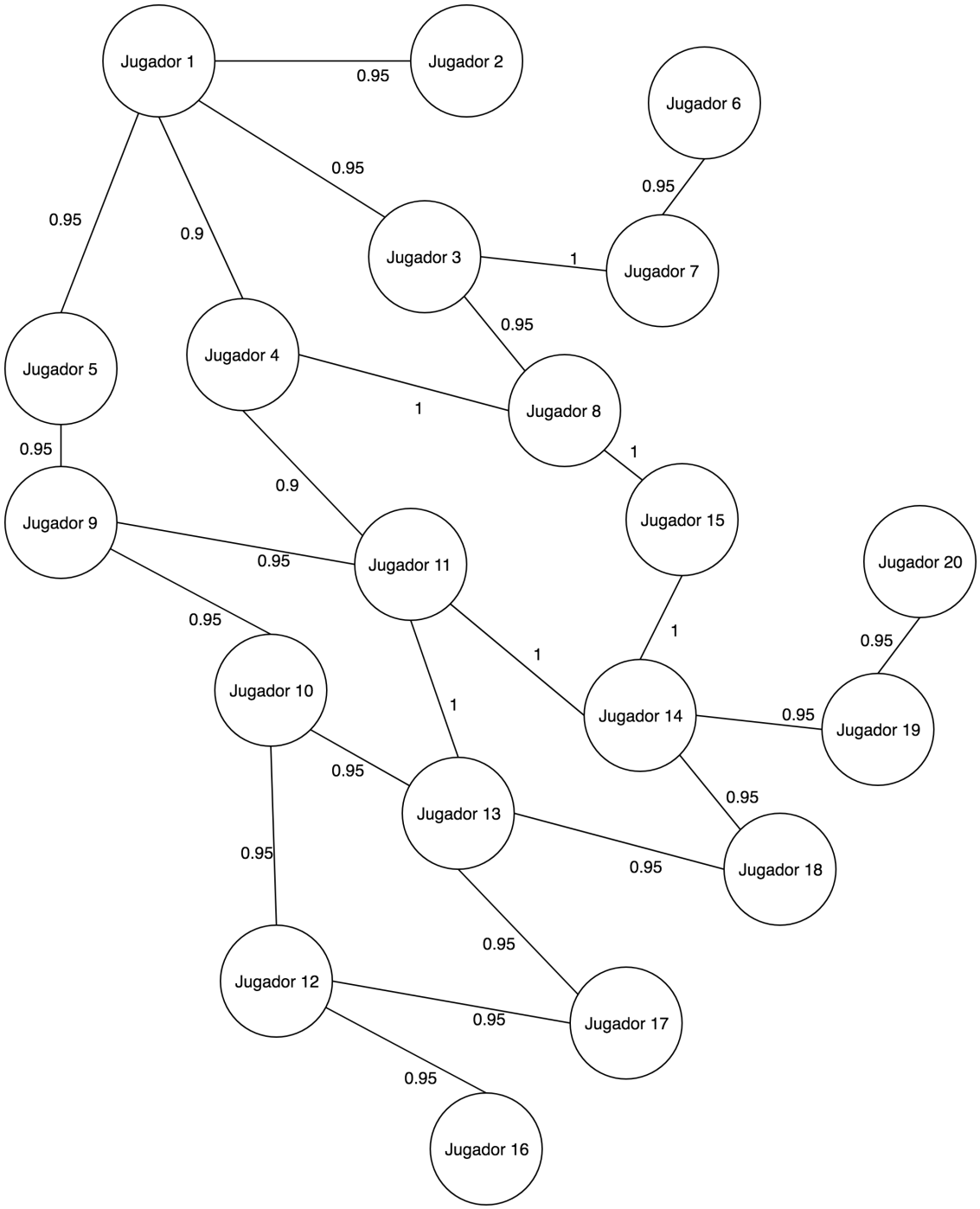


Figura 1: Ejemplo del grafo de un universo de 20 jugadores

Como se puede ver en la figura 1, los jugadores 4 y 11 son solamente conocidos ya que su afinidad es de 0,9; el jugador 13 y el jugador 17 son amigos lejanos, ya que su afinidad es de 0,95; y los jugadores 4 y 8 son amigos cercanos ya que su afinidad es de 1,0. Los jugadores 14 y 20 no tienen ninguna relación entre ellos. Finalmente, para cualquier par de jugadores es posible encontrar una secuencia de aristas que los conecta.

4.1.1. Afinidad entre jugadores no relacionados

Dos jugadores que no se conocen también poseen un nivel de afinidad. Para calcular la afinidad entre ellos, es necesario encontrar el camino **de mayor afinidad** entre ellos, donde la afinidad del camino es:

$$1 - \sum_{i:\text{arista}} (1 - \text{afinidad}_i)$$

Calcularemos la afinidad del camino en el ejemplo de la figura 2.

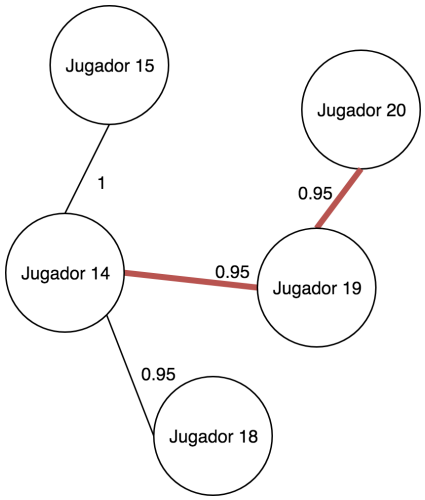


Figura 2: Ejemplo de un camino entre los jugadores 14 y 20

En la figura 2, podemos observar un camino en color rojo, que conecta al jugador 14 y 20 a través del jugador 19. Como mencionamos, la afinidad del camino es la suma de $(1 - \text{afinidad})$ de cada uno de los tramos del camino. Para este caso:

$$\text{Afinidad del camino} = 1 - [(1 - 0{,}95) + (1 - 0{,}95)] = 0{,}9$$

Para este ejemplo, el camino rojo es el único camino que conecta a los jugadores 14 y 20, por lo que también es el de mayor afinidad. Entonces podemos afirmar que la afinidad es 0,9.

En la figura 3, podemos ver un ejemplo en que hay cuatro caminos que permiten conectar al jugador 11 con el jugador 1. Los caminos están marcados en amarillo, rojo, azul y morado. Notemos que estos no son los únicos posibles caminos. De hecho, es posible pensar en muchos más caminos, como por ejemplo el camino [Jugador 11, Jugador 4, Jugador 8, Jugador 3, Jugador 1].

El ejemplo de la figura 3 nos ayuda a visualizar que **no necesariamente el camino con menor largo (número de nodos en el camino) es necesariamente el camino de mayor afinidad**. Para eso, analicemos las cuatro rutas descritas el la figura 3, cuyo análisis se puede ver en la tabla 1.

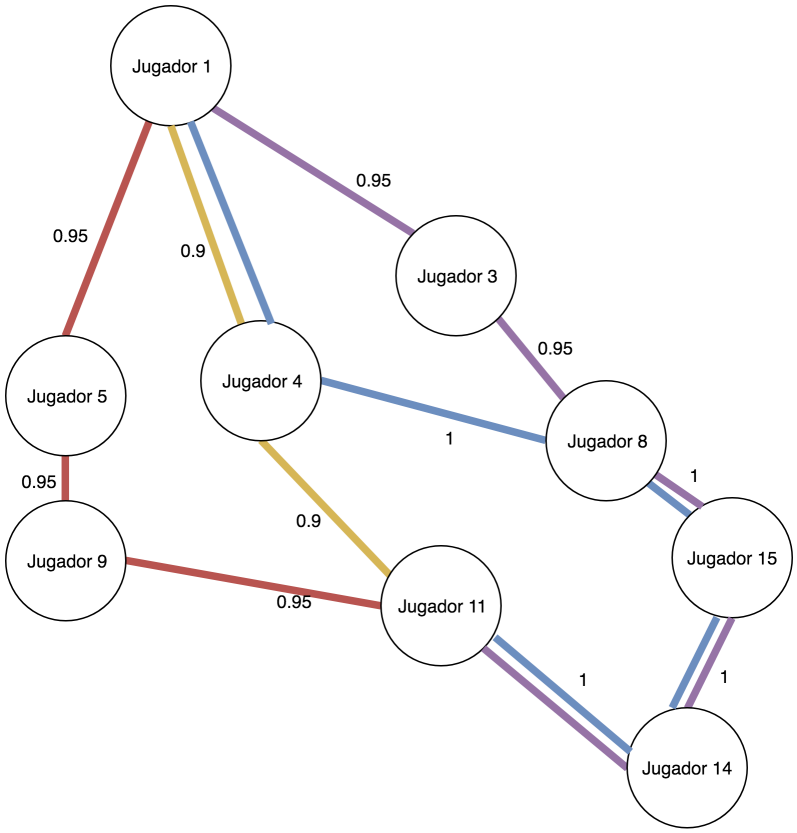


Figura 3: Ejemplo de algunos candidatos a caminos mas afines entre los jugadores 1 y 11

Nombre	Largo	Camino	Cálculo de la afinidad del camino	Afinidad
Amarillo	2	11, 4, 1	$1-[(1-0{,}9)+(1-0{,}9)]$	0,80
Rojo	3	11, 9, 5, 1	$1-[(1-0{,}95)+(1-0{,}95)+(1-0{,}95)]$	0,85
Azul	5	11, 14, 15, 8, 4, 1	$1-[(1-1)+(1-1)+(1-1)+(1-1)+(1-0{,}9)]$	0,90
Morada	5	11, 14, 15, 8, 3, 1	$1-[(1-1)+(1-1)+(1-1)+(1-0{,}95)+(1-0{,}95)]$	0,90

Tabla 1: Afinidad de distintos caminos entre los jugadores 11 y 1

En la tabla 1 podemos observar que para el camino amarillo, de menor largo, su afinidad es de 0,8; sin embargo el camino rojo que es de largo 3, mayor que el largo del camino amarillo, tiene una mayor afinidad. Tanto el camino azul, que tiene un largo de 5, y el camino morado, de largo 5, ambos mayores al largo del camino amarillo y camino rojo, son los caminos con mayor afinidad[3]. En este caso, la afinidad de dos caminos es igual y ambas son las mayores, por lo que la afinidad entre jugador 11 y jugador 1 es de 0,9.

En su tarea, ustedes deben implementar un algoritmo que recorra su grafo y permita encontrar la ruta de mayor afinidad entre dos jugadores.

[3]Nota que el camino azul y morado tienen la mayor afinidad, tanto para los caminos analizados en el ejemplo como para todos los otros. Caminos no considerados incluyen el [11, 4, 8, 3, 1] entre varios otros. En su tarea, deberás considerar todos los posibles caminos entre dos jugadores.

4.1.2. Consultas y estadísticas sobre el grafo de afinidad

Pep Ocho está interesado en algunas estadísticas tanto para jugadores como para la población de jugadores totales, por lo que se espera que en su tarea ustedes sean capaces de calcular las siguientes datos:

- **Mejor amigo**: Para cualquier jugador, ustedes deben ser capaces de encontrar el jugador con quien tiene mayor afinidad.
- **Peor amigo**: Para cualquier jugador, ustedes deben ser capaces de encontrar el jugador con quien tiene menor afinidad.
- **El más popular**: Encontrar el jugador o los jugadores que tiene o que tienen más amigos, es decir, el que tiene mas aristas.
- **El fichaje estrella**: A Pep Ocho le interesa realizar nuevas contrataciones; para esto quiere que realices un fichaje estrella. Se te entregan las siguientes instrucciones para que, dado un jugador, selecciones un fichaje estrella. Dado un jugador:
  1. Considera a todos los jugadores con grado 1 de relación, es decir todos los amigos cercanos o amigos lejanos o conocidos.
  2. Entre estos jugadores el fichaje estrella será el jugador que tenga más *chispeza*, donde la *chispeza* con un jugador se calcula como:

$$\text{Chispeza} = \text{Afinidad} \times \text{Overall}$$

4.2. Campeonato [21 %]

Los campeonatos son torneos de eliminación directa: se juega sólo un partido y el ganador avanza a la siguiente fase. Para los torneos se recomienda trabajar con árboles.

4.2.1. Inicio de campeonato

El usuario debe ser capaz de crear su propio equipo con 11 jugadores de su preferencia. Puede incluir cualquier jugador siempre que el grafo de afinidades entre los 11 jugadores sea conexo. Cabe mencionar que las posiciones y formaciones son indiferentes para la tarea. Ya con el equipo armado, se debe crear automáticamente el resto de los equipos de forma aleatoria con jugadores restantes. Se pide igualmente que el grafo de los otros equipos sea conexo.

4.2.2. Eliminatoria

Con los equipos ya formados, se puede dar inicio al campeonato de PES. El torneo se inicia con 16 equipos, los cuales juegan una ronda eliminatoria en que se juega sólo un partido, y el ganador pasa a la siguiente fase. En semifinales, cada ganador pasan a la final, y los perdedores juegan un partido para determinar el tercer lugar.

Todos los partidos deberán tener un *id*: los de octavos de final irán del 1 a 8; en cuartos del 9 al 12; en semifinales, 13 y 14; y tercer lugar y final serán 15 y 16 respectivamente. Es importante guardar un registro de las estadísticas que se dan en cada partido, las cuales son explicadas a continuación.

4.2.3. Partidos

¡Es hora de jugar! En cada partido los equipos parten con una medida que indica la esperanza de ganar del equipo. Esto depende de la afinidad que calculaste en las secciones anteriores, y la calidad de tus jugadores. Esto porque, si bien es importante tener química con tus compañeros, es igualmente importante tener

a jugadores con talento.

La esperanza de ganar se calcula de la siguiente forma:

$$\text{Esperanza de ganar} = \text{Afinidad total} \times \text{Calidad equipo}$$

Donde la calidad de un equipo está dada por:

$$\text{Calidad Equipo} = \frac{\Sigma \ \text{Overall titulares}}{1089}$$

Esto significa que la calidad es la suma del *overall* de todos los jugadores, normalizado por la calidad máxima que puede tener un equipo de 11 jugadores ($11 \times 99 = 1089$).

La afinidad total de un equipo se calcula de la siguiente manera:

1. Se calcula la afinidad promedio de cada jugador con aquellos jugadores que sean vecinos a él en el grafo de afinidad del equipo.
2. La afinidad del equipo será el promedio simple de todas las afinidades de los jugadores.

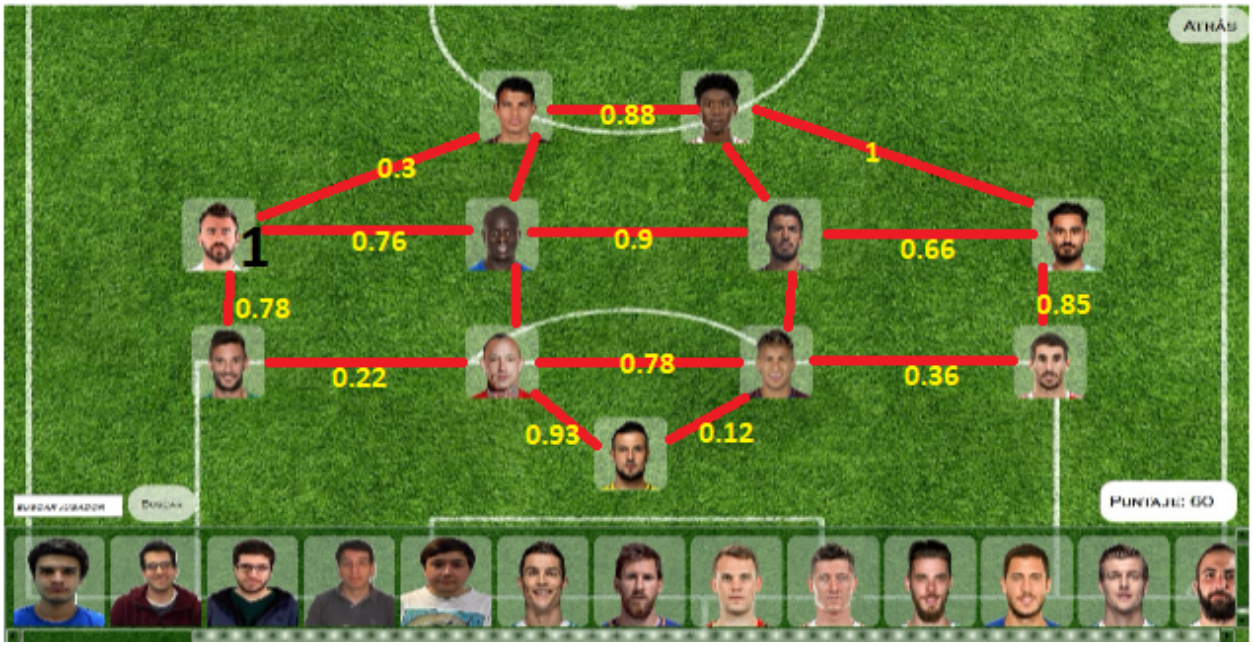


Figura 4: Ejemplo de formación y afinidad equipo

En la figura 4 la afinidad del jugador 1 será $\text{Afinidad}(1) = (0{,}3 + 0{,}76 + 0{,}78)/3 = 0{,}6133$. Es necesario calcular la afinidad puntual de cada jugador, y promediar sus valores para obtener la afinidad total del equipo.

4.2.4. Eventos en partidos

Un equipo puede estar muy bien construido y tener una alta esperanza de ganar; sin embargo, pueden suceder eventos que alteren esa esperanza inicial. Los eventos que sucedan deben almacenarse por cada partido, junto con los jugadores involucrados en el evento.

- **Faltas:** Si un jugador no se lleva bien con sus compañeros, éstos no lo ayudarán, y se verá forzado a cometer faltas. La probabilidad de cometer una falta comienza siendo 5 %, y aumenta 2 % por cada compañero de equipo con el que tenga afinidad bajo 0,8. Esta afinidad es la de jugadores vecinos y también es la mencionada en el punto 4.1.1, de jugadores no relacionados, si es que estos no están conectados directamente. Por ejemplo, si un jugador tiene 4 compañeros con afinidad bajo el umbral, su probabilidad de cometer falta será de 13 %. Las faltas de cada jugador se deben sumar a un total de equipo. Cada falta de equipo reduce en un 1 % la esperanza de ganar del equipo. Es decir, la esperanza se reduce según la siguiente fórmula:

$$\text{Esperanza Nueva} = \text{Esperanza Inicial} \times (1 - \tfrac{\text{Número de Faltas}}{100})$$

- **Jugadores con tarjeta roja y amarilla:** Los equipos pueden obtener tarjetas en sus partidos. Cada jugador solo puede obtener una sola tarjeta de cada tipo. Es decir, puede obtener 1 amarilla, o bien 1 roja, o bien 1 amarilla y 1 roja, o el último caso que es no obtener tarjetas. No existen dobles tarjetas amarillas. Las tarjetas deben ser guardadas luego de cada partido. Cada jugador tiene las siguientes probabilidades de obtener una tarjeta:

Tarjeta	**Probabilidad**
Amarilla	20 %
Roja	5 %

- **Goles:** Finalmente, para determinar al ganador del partido, es necesario calcular los goles que hará cada equipo. Los goles que hace cada equipo estarán dados por:

$$\text{Goles} = \left\lfloor \left(\frac{\text{Esperanza final de ganar}}{40} \right)^2 \right\rfloor$$

  El ganador es el que marque más goles. En caso de empate, los equipos se irán a penales, donde el equipo con mayor esperanza tiene un 80 % de ganar, y el equipo con menor esperanza, un 20 %.

4.2.5. Consultas

Una vez terminado el campeonato, quieres saber la mayor cantidad de estadísticas sobre éste, y por eso te pedimos que implementes las siguientes consultas:

- **Información general del usuario:** Se debe poder mostrar el estatus del equipo del usuario: hasta que ronda llegó, quién lo eliminó, cuántos goles hizo, cuántos le hicieron, faltas, tarjetas rojas y amarillas.
- **Información general de un equipo:** Dado el nombre de un equipo, se debe poder mostrar la misma información que para el equipo del usuario.
- **Ganadores:** Mostrar el primer, segundo y tercer lugar.
- **Información por fase:** Dado el número de una fase se deben mostrar los equipos que pasaron a la siguiente fase y los que se quedaron en la fase solicitada.
- **Información por partido:** Dado el *id* de un partido se debe mostrar el número de goles hechos por cada equipo, las faltas y tarjetas.

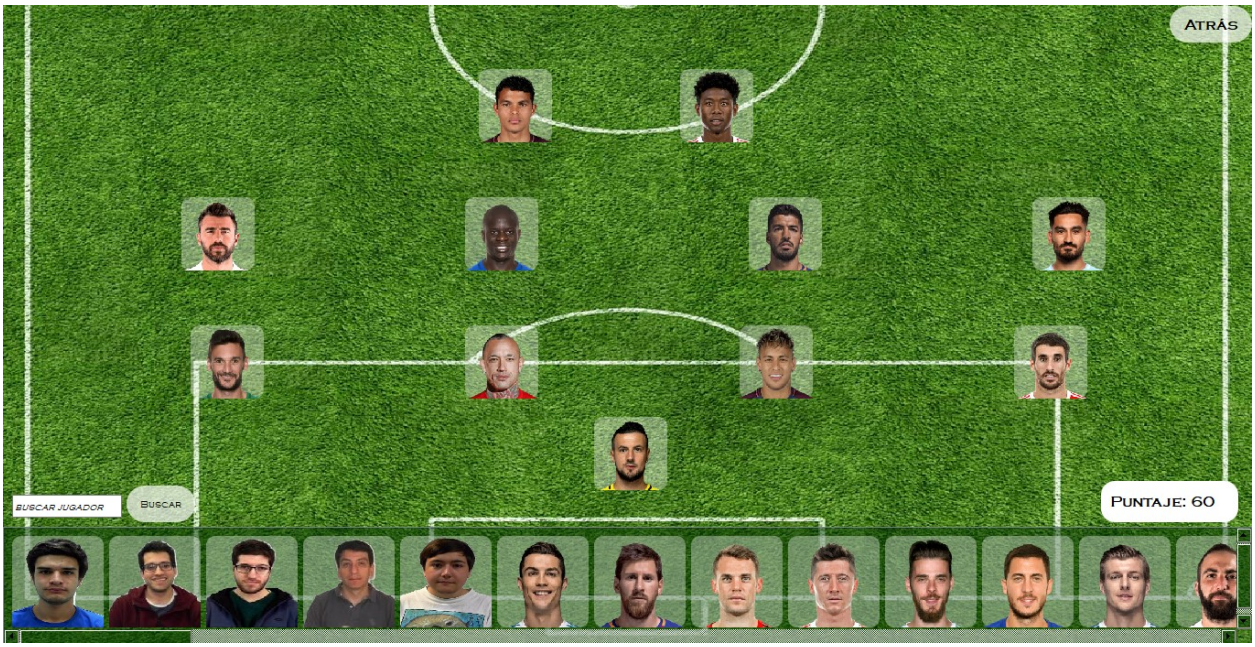
4.3. GUI [6 %]

Toda la interacción con el programa será a través de una interfaz gráfica, la cual te entregaremos y contará con distintos métodos para que puedas realizar tu tarea. Para desarrollar tu tarea, tendrás que sobrescribir los métodos del archivo `demo.py`, además de agregar los tuyos propios.

Para instanciar la GUI, debes tener una estructura de datos **ideada por ti** que te permita entregarle 200 jugadores, seleccionados al azar, con atributos iterables ordenados en el mismo orden que la base de datos (`id`, `nombre`, `nombre_completo`, ...) y al menos 16 equipos de la forma `nombre_equipo`, `esperanza_ganar`. En el archivo `demo.py` se muestra un ejemplo de cómo llamar a la GUI.
Ten en cuenta que no puedes usar listas ni debes escribir los jugadores y equipos uno a uno.

4.3.1. Edición de equipo

Para editar tu equipo, contarás con la siguiente interfaz gráfica. Las posiciones de los jugadores dentro del equipo son irrelevantes para el cálculo de la esperanza de ganar del equipo, y se usan en la interfaz sólo por temas visuales.



Al hacer clic en un jugador dentro de la cancha, se muestra la información del jugador y se da la opción de cambiarlo:

Si se hace clic en cambiar, luego se debe hacer clic en cualquier jugador (dentro o fuera de la cancha) para intercambiarlo con este. Si se hace clic en cualquier otro lugar, el cambio se cancela.
También puedes utilizar el buscador que se encuentra por arriba de los jugadores que están fuera del equipo para buscar a uno o más jugadores específicos.

Es importante destacar que los archivos `interfaz.py` y `clases.py` no deben ser modificados ni tampoco es necesario que los abras o los leas. Sólo debes trabajar con los métodos que se listan a continuación (además de los que inventes). El resto de los métodos y atributos de los archivos previamente mencionados no los puedes utilizar.

4.3.2. Interfaz de campeonatos

Para simular los campeonatos, contarás con la siguiente interfaz:

Se divide en 4 secciones:

- `1`: Se listan todos los equipos con los que se inicializó la interfaz, sumados al primer equipo (tu equipo).
- `2`: Esta es la tabla de resultados, donde debes agregar los resultados del campeonato al iniciarlo con 16 equipos seleccionados o al apretar el botón *al azar*.

- `3`: En esta zona se encuentran los botones necesarios para que el usuario pueda realizar las consultas que desee. Los botones se activan cuando se simula el campeonato.
- `4`: Esta es la tabla de respuestas. En ella debes agregar las respuestas a las consultas realizadas por el usuario.

4.3.3. Métodos de `demo.py`:

Los siguientes métodos no debes llamarlos, sino que se llaman automáticamente cuando ocurre algo específico.

- `cambio_jugador(j1, j2, en_cancha)`: Se llama cada vez que el usuario hace un cambio de jugadores. en_cancha es un booleano que es `True` si `j2` es un jugador en cancha (es decir, si se hizo un cambio de posición entre dos jugadores en cancha) y `False` si es un jugador que entró de afuera. `j1` siempre será un jugador que está en cancha. `j1` y `j2` son los nombres completos de los jugadores, en el orden que se les seleccionó.
- `simular_campeonato(equipos)`: Se llama cuando el usuario aprieta el botón de jugar o el botón de aleatorio en la interfaz de campeonatos. Los equipos recibidos son los equipos seleccionados en el cuadro de equipos (o 16 equipos al azar), en una lista de la forma `[(nombre_equipo_1, esperanza_ganar_1), (nombre_equipo_2, esperanza_ganar_2), ...]`. Es importante destacar que si bien tu código puede leer y utilizar los datos de esta lista, no debe manipular la lista de ninguna manera.

  Para cada uno de los siguientes métodos, al presionar el botón indicado, se debe mostrar en la tabla respuestas la respuesta a la consulta (indicadas en la sección 4.2.5). Toma en cuenta que algunas consultas reciben parámetros, los cuales son escritos por el usuario en la GUI, por lo que este parámetro puede no existir (por ejemplo, el usuario ingresa un nombre de equipo no existente).

- `consulta_usuario()`: Se llama cuando el usuario aprieta el botón de consulta usuario en la interfaz de campeonato.
- `consulta_equipo(nombre)`: Recibe el nombre del equipo, Se llama cuando el usuario consulta por un equipo.
- `consulta_ganadores()`: Se llama cuando el usuario consulta por los ganadores.
- `consulta_partido(id)`: Se llama cuando el usuario consulta por un partido. Recibe el *id* (en formato *string*) del partido que quiere ver el usuario.
- `consulta_fase(numero)`: se llama cuando el usuario consulta por una fase. Recibe el número de la fase, como *string*.

4.3.4. Métodos de la GUI:

Los siguientes métodos nunca se llaman automáticamente y debes llamarlos para producir cambios específicos en la GUI. Para llamarlos, debes usar `self.gui.*metodo a utilizar*`.

- `cambiar_esperanza(valor)`: Recibe un valor y cambia la esperanza de ganar del equipo en la GUI. Sólo se puede utilizar cuando el usuario está en la interfaz del equipo.
- `agregar_resultado(resultado)`: En la interfaz de campeonatos, agrega el resultado dado a la tabla de resultados. El atributo recibido debe ser un *string*.
- `agregar_respuesta(respuesta)`: En la interfaz de campeonatos, agrega la respuesta a la tabla de respuestas. El atributo recibido debe ser un *string*.

- `resetear_resultados()`: En la interfaz de campeonatos, deja en blanco la tabla de resultados.
- `resetear_respuestas()`: En la interfaz de campeonatos, deja en blanco la tabla de respuestas.

4.4. Base de datos

Para proporcionarte datos más reales y así puedas ayudar a Pep Ocho de manera más exacta, te hemos brindado los datos reales del juego FIFA 18. A continuación, se explican los archivos con los cuales deberás interactuar:

- `players_db.csv`: En este archivo están los datos de todos los jugadores. Se encuentra en la carpeta `Assets` de la GUI y tiene la estructura detallada en la tabla 2:

Nombre	Tipo de dato	Comentarios
ID	`int`	cada ID es único y van desde 0 en adelante
ALIAS	`string`	nombre abreviado
FULL_NAME	`string`	nombre completo
CLUB	`string`	club al que pertenece el jugador
LEAGUE	`string`	liga a la que pertenece el jugador
NATIONALITY	`string`	nacionalidad del jugador
OVERALL	`int`	habilidad del jugador, va desde 0 a 99

Tabla 2: `players_db.csv`

- Carpeta `photo`: En esta carpeta se encuentran las fotos de cada uno de los jugadores. Cada imagen tiene como nombre `num.png` donde `num` es la ID del jugador señalada en la base de datos. La carpeta en formato ZIP esta en este ***link***, para que las fotos puedan ser visualizadas es necesario descomprimirlas en la carpeta `Assets` de la GUI.

4.5. *Hackerman* [30 %]

Para demostrar tu capacidad como programador, decides demostrarle al mundo que eres capaz de gestionar las relaciones entre jugadores **SIN USAR LAS ESTRUCTURAS DADAS POR PYTHON** (listas, diccionarios, *sets*, etcétera)[4]. Heredar de estas estructuras también esta prohibido. Debes detallar en el **README.md** las especificaciones de cómo implementaste cada una de las estructuras utilizadas. Tus implementaciones deben replicar la funcionalidad básica de sus contrapartes; esto es, aquellas que son iterables (`__iter__`), deben serlo. Si son estructuras ordenadas deben poder ordenarse usando el método `sort`. Además deben responder bien a la función `len` (`__len__`), `append` y a la indexación (`__getitem__`, `__setitem__`). Para esto, deben hacer uso de los *dunder methods* correspondientes.

4.6. `.gitignore`

Para no saturar los repositorios de GitHub, **deberás** agregar en tu carpeta `Tareas/T02/` un archivo llamado `.gititgnore` de tal forma de **no** subir los archivos de la carpeta `gui` ni la carpeta `players_photo` ni el archivo `players_db.csv`.

5. Entregable [3 %]

Para esta tarea, se te solicitará responder un Google Form con tus datos de usuario de GitHub y número de alumno, donde especifiques dos algoritmos que utilizarás para poder realizar las consultas de afinidad,

[4] https://docs.python.org/3/tutorial/datastructures.html

y de utilizar alguna fuente externa, citarla (página web, libro, etc.). Además de esto, elige cuál intentarás primero y menciona una ventaja de ese sobre el otro, explicando como lo adaptarías al problema.

6. Restricciones y alcances

- Tu programa debe ser desarrollado en Python v3.6.
- Esta tarea es estrictamente individual, y está regida por el Código de Honor de la Escuela: Clickear para Leer.
- No utilizar `.gitignore` para evitar subir la carpeta `players-photo` tendrá un descuento de 5 décimas.
- No se corregirán partes de código que utilicen estructuras dadas por python, a menos que se indique lo contrario en una parte especifica del enunciado.
- Tu código debe seguir la guía de estilos descrita en el PEP8.
- Si no se encuentra especificado en el enunciado, asume que el uso de cualquier librería Python está prohibida. Pregunta en el foro si es que es posible utilizar alguna librería en particular.
- El ayudante puede castigar el puntaje[5] de tu tarea, si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación algoritmos.
- Debes adjuntar un archivo `README.md` donde comentes sus alcances y el funcionamiento del sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*. **Tendrás hasta 24 horas después de la fecha de entrega** de la tarea para subir el `README.md` a tu repositorio.
- Crea un módulo para cada conjunto de clases. Divídelas por las relaciones y los tipos que poseen en común. **Se descontará hasta un punto si se entrega la tarea en un solo módulo[6].**
- Cualquier aspecto no especificado queda a tu criterio, siempre que no pase por sobre otro.

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).

[5] Hasta −5 décimas.
[6] No agarres tu código de un solo módulo para dividirlo en dos; separa su código de forma lógica