Tarea 1
Optimización de distancias
Entrega: viernes 14 de abril a las 21:00 hrs.

1. Introducción

Usted ha sido contratado para ayudar a una liga de fútbol masiva a organizar sus partidos. Debido a la enorme cantidad de equipos y a la extensión territorial en que están dispersos, la asociación decide que en la primera fase cada equipo juegue **exactamente** 4 partidos, intentando minimizar la distancia a recorrer por los equipos.

Para hacer esto le piden crear los partidos de manera que todos los equipos juegen 4 partidos y la distancia total viajada por todos los equipos sea la menor posible (esto no necesariamente será verdad para cada equipo en particular). Para esto le proporcionan las ubicaciones geográficas de todos los equipos participantes.

2. La Tarea

Para resolver esta tarea, se debe implementar un programa en lenguaje C que permita organizar los partidos de fútbol de manera eficiente. Se proporcionará como entrada un archivo que contenga la información de ubicación geográfica de cada uno de los equipos participantes.

En primer lugar, el programa debe leer el archivo de entrada y almacenar la ubicación de los equipos en un arreglo, esta información estará en el formato de `nombre,x,y` donde cada coordenada está asociada únicamente a un equipo, el cual se identifica por la posición de sus coordenadas en el archivo de entrada (ej: las coordenadas de la primera linea de texto corresponden al equipo 1, la segunda al equipo 2, etc.)

Hint: para calcular la distancia entre dos equipos usar distancia euclidiana:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

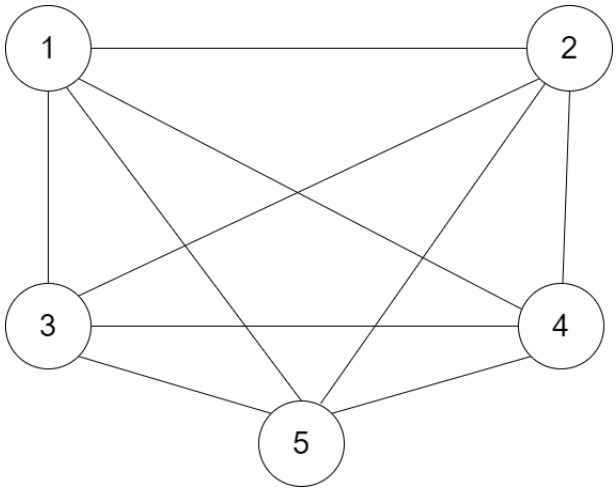

Figura 1: Ejemplo mínimo con solo 5 equipos

La relación entre estos equipos puede ser expresada a través de una matriz de adyacencia:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$$

Figura 2: Matriz de adyacencia de la Fig. 1

El índice de cada fila corresponde al número del equipo en la figura anterior, un 1 en las columnas significa que juega contra el equipo y un 0 que no. Por ejemplo, la primera fila corresponde al equipo ”1”quien juega contra 2, 3, 4, 5, como obviamente no puede jugar contra si mismo la primera columna tiene un 0.

Es importante tener en consideración que la solucion optima general no necesariamente es la optima para cada equipo, por ejemplo:

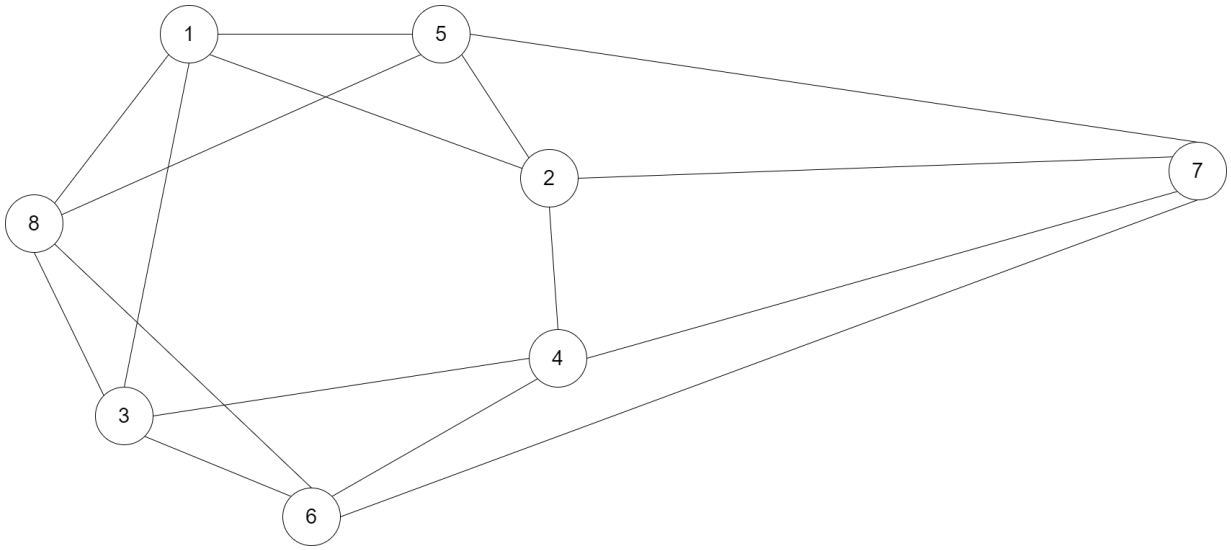

Figura 3: Ejemplo con óptimo general distinto al óptimo de cada equipo individualmente

Y además, no importa si los equipos quedan separados en divisiones zonales, como en la siguiente figura:

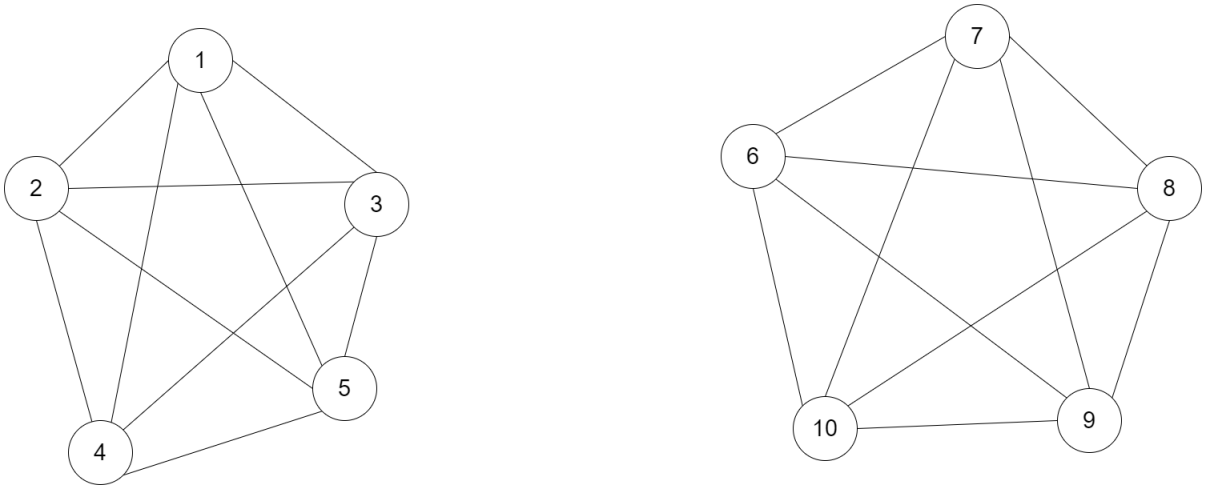

Figura 4: Ejemplo equipos separados por zonas

3. Evaluación

El puntaje se distribuirá de la siguiente manera:

- **[1 pt.]** Leer correctamente el archivo.
- **[1 pt.]** Llegar a una solución válida (todos juegan exactamente 4 partidos, aunque la distancia no sea óptima).
- **[1 pt.]** Imprimir salida en un archivo (Formato: `NombreEquipo,rival1,rival2,rival3,rival4`)
- **[1 pt.]** Por implementar algún tipo de optimización útil.
- **[1 pt.]** Por lograr una eficiencia razonable[1] en la distancia total, aunque no sea óptima.
- **[1 pt.]** Escalabilidad de la solución (o sea, ser capaz de optimizar con muchos equipos a la vez en un tiempo breve).
- Bonus: **[0.3 pt.]** Mejor solución

4. Consideraciones Generales

Su programa debe ser robusto frente a datos que no corresponden. Esto quiere decir que su programa no debe caerse en caso de que el usuario haga operaciones matemáticas no válidas (p.ej., dividir por cero si fuera el caso). Sin embargo, se puede asumir que el usuario será amigable, en el sentido de que cuando le soliciten un número este no ingresará letras, por ejemplo.

Recuerde que el archivo puede tener más de una línea de texto.

5. Evaluación y Entrega

Esta tarea puede ser resuleta en grupos de máximo 2 personas. El plazo para la entrega de la tarea vence impostergablemente el viernes 14 de abril a las 21:00 hrs..

Formato de entrega: Subir un único archivo con el código de su programa al módulo de tareas de la página del curso, con el nombre de archivo "APELLIDO1-APELLIDO2-Tarea1.c". El incumplimiento de este requisito de nombre de archivo será penalizado con un descuento de un punto en la nota. Los archivos compilados no serán tomados en cuenta, si se llega a subir solo un archivo compilado, este será ignorado, y será evaluado con nota 1.

En el momento de compilación, se deberá indicar las siguientes *flags* al compilador:

```
-Wall -Wextra -Wundef -Werror -Wuninitialized -Winit-self
```

Aquellas tareas que no compilen de la forma normal (gcc -o salida entrada) serán evaluadas con nota 1. Las que compilen, pero se caigan durante la ejecución, serán evaluadas con nota máxima 3.

[1] Se comparará con una solución simple hecha por el ayudante.

Su programa será evaluado con múltiples casos de prueba y deberá ser capaz de ejecutarlos todos de manera correcta. De fallar en algún caso de prueba serán descontados los puntos correspondientes a dicho caso.

6. Consideraciones de Trabajo

El trabajo en esta tarea puede hacerse en parejas. Cuide su tarea para que no sea copiada parcial o íntegramente por otros. Todas las tareas entregadas serán comparadas por un sistema automático de detección de plagios. Cualquier copia (de otras tareas o de internet) será penalizada, recibiendo el mismo castigo tanto quien copia como quien permite que le copien. También es considerada copia cualquier ayuda externa recibida directamente en la tarea, sin importar si proviene de un alumno del curso, de la universidad, o de otro lugar. El castigo será establecido por el Consejo de la Facultad, siendo como mínimo un 1,0 de promedio en el curso.