Proyecto Final

1st Daniela Orrego Alfonso
2259579-2427
Universidad Del Valle
Tuluá, Colombia
daniela.orrego@correounivalle.edu.co

2nd Juan Camilo Velez Ospina
2259635-2724
Universidad Del Valle
Tuluá, Colombia
juan.velez.ospina@correounivalle.edu.co

***Index Terms*—Cronograma, Equipos, Distancia, Tiempo**

I. Introducción

El proyecto muestra la optimización de una programación de partidos para equipos deportivos, considerando restricciones de distancia entre ciudades.

II. Facilidad de Uso

A. por clases

Clase Algoritmo: El código está bien estructurado y sigue las convenciones de nomenclatura de Java, lo que facilita la lectura. Las secciones del código están separadas lógicamente, por ejemplo, la lectura de archivos, la creación de matrices, los cálculos y la impresión de resultados. Se proporcionan comentarios en varios lugares, lo que ayuda a entender la funcionalidad de las secciones del código. La interfaz gráfica proporciona una forma fácil de que el usuario seleccione un archivo. Se manejan las excepciones (IOException) y se imprime un mensaje claro en caso de error. Se imprime el tiempo total de ejecución del algoritmo, lo que es útil para evaluar la eficiencia.

Clase PTorneo: El código está bien estructurado y utiliza nombres de variables descriptivos. La lógica de generación del horario es clara Se proporcionan comentarios que describen las secciones clave del código y las funciones específicas. Se manejan las excepciones (IOException), lo que es adecuado para la operación de lectura y escritura. Se imprime el horario resultante por consola, lo que puede ser útil para la verificación.

III. Objetivos

Clase PTorneo: El objetivo principal es generar un horario de partidos para el torneo. Utiliza una lógica específica para asignar local/visitante a cada equipo en cada ronda del torneo. Configura vectores y matrices necesarios para la generación del horario. Asigna valores a una matriz que representa el horario de los partidos, considerando restricciones específicas del problema. Imprime el horario resultante por consola.

Clase Algoritmo: El código comienza permitiendo al usuario seleccionar un archivo .txt que contiene datos relevantes para el problema. El archivo debe contener información sobre el número máximo de juegos consecutivos en casa o fuera y la matriz de distancias entre ciudades. Lee y procesa los datos del archivo seleccionado, incluidos los máximos consecutivos de juegos en casa/fuera y la matriz de distancias entre ciudades. Calcula el peso de estrellas para cada ciudad basándose en la matriz de distancias. Encuentra el índice del nodo con el peso mínimo de estrella. Utiliza un algoritmo de intercambio para encontrar la ruta más corta teniendo en cuenta el nodo con el peso mínimo de estrella como nodo inicial/final. Utiliza la clase PTorneo para generar un horario y calcula las distancias recorridas por cada equipo. Imprime la mejor ruta encontrada y la distancia total recorrida por todos los equipos. Muestra el tiempo total de cálculo.

A. Analisis temporal y espacial

Clase Algoritmo: Análisis Temporal: La lectura del archivo es lineal con respecto a la cantidad de datos en el archivo, lo que se representa como O(N), donde N es el número de datos en el archivo. La generación del horario (PTorneo) utiliza bucles anidados, lo que da como resultado un tiempo de ejecución de aproximadamente $O(N^2)$, donde N es el número de equipos. Otros cálculos en el código, como el cálculo de distancias, también pueden depender de la cantidad de equipos y ciudades, lo que puede ser $O(N^2)$ en el peor caso. Análisis Espacial: Se utilizan matrices y arreglos para almacenar información sobre equipos, distancias y horarios. La complejidad espacial es principalmente $O(N^2)$ debido a las matrices bidimensionales.

Clase PTorneo: Análisis Temporal: El tiempo de ejecución para generar el horario (PTorneo) es principalmente $O(N^2)$, ya que implica bucles anidados sobre el número de equipos. Análisis Espacial: Se utilizan matrices y vectores para almacenar información sobre el horario y las asignaciones de local/visitante. La complejidad espacial es principalmente $O(N^2)$.

B. Analisis General

Temporal: La complejidad temporal general está dominada por las operaciones de generación del horario y otros cálculos, siendo en su mayoría cuadrática ($O(N^2)$). El rendimiento del algoritmo puede depender de factores como el tamaño del

Identify applicable funding agency here. If none, delete this.

conjunto de datos y la eficiencia de las implementaciones específicas, incluida la lógica de intercambio.

Espacial: La complejidad espacial está dominada por la utilización de matrices y arreglos bidimensionales, siendo principalmente cuadrática ($O(N^2)$). El uso de memoria está relacionado principalmente con el tamaño del conjunto de datos, especialmente el número de equipos y ciudades. Es importante tener en cuenta que estas son estimaciones generales basadas en el análisis del código proporcionado y podrían variar según la implementación exacta de las funciones faltantes y los detalles específicos del problema y los datos de entrada.

C. Analisis de cercania al Optimo

- Heurística de Intercambio (PTorneo): El algoritmo implementa un enfoque de intercambio iterativo para encontrar una solución que mejore continuamente la calidad de la programación de partidos. La cercanía al óptimo dependerá de la eficacia de este enfoque de intercambio y su capacidad para explorar y mejorar soluciones.
- Complejidad Temporal y Espacial: La complejidad temporal y espacial del algoritmo puede afectar su capacidad para manejar conjuntos de datos más grandes y complejos. Una complejidad cuadrática ($O(N^2)$) en términos de tiempo y espacio podría limitar la eficiencia en problemas grandes.
- Influencia de Parámetros: La calidad de las soluciones podría depender de parámetros específicos del problema, como el número máximo de juegos consecutivos en casa o fuera. Ajustar estos parámetros podría influir en la cercanía al óptimo de las soluciones generadas.
- Exploración del Espacio de Soluciones: La calidad de las soluciones también dependerá de la capacidad del algoritmo para explorar de manera efectiva el espacio de soluciones y evitar quedar atrapado en mínimos locales.
- Lógica de Peso de Estrellas y Distancias: La lógica utilizada para calcular los pesos de estrellas y las distancias podría influir en la calidad de las soluciones. Una ponderación adecuada entre estas variables es crucial para obtener soluciones cercanas al óptimo.

D. Detalles principales de la Implementación

Clase Algoritmo: Utiliza JFileChooser para permitir al usuario seleccionar un archivo .txt. Lee el archivo para obtener información sobre el número máximo de juegos consecutivos en casa/fuera y la matriz de distancias entre ciudades. Calcula el peso de estrellas para cada ciudad con base en la matriz de distancias. Identifica el índice del nodo con el peso mínimo de estrella. Implementa un algoritmo de intercambio iterativo para encontrar una ruta más corta, utilizando el peso mínimo de estrella como nodo inicial/final. Crea y maneja matrices para representar distancias entre ciudades y otros cálculos necesarios. Realiza cálculos específicos para obtener resultados relevantes al problema. Imprime la mejor ruta encontrada, la distancia total recorrida por los equipos y el tiempo total de cálculo.

Clase PTorneo: Implementa un método para generar un horario basado en la lógica específica del torneo. Asigna local/visitante a cada equipo en cada ronda. Inicializa vectores y matrices necesarios para la generación del horario. Asigna valores a la matriz para representar el horario de partidos de acuerdo con la lógica del torneo. Imprime el horario resultante por consola.

E. Analisis de resultados

m1.txt	Tiempo
	52ms

m2.txt	Tiempo
	63ms

m3.txt	Tiempo
	48ms

m4.txt	Tiempo
	107ms

m5.txt	Tiempo
	47ms

m6.txt	Tiempo
	47ms

m7.txt	Tiempo
	45ms

m8.txt	Tiempo
	39ms

m9.txt	Tiempo
	58ms

m10.txt	Tiempo
	73ms

m11.txt	Tiempo
	57ms

F. Conclusiones

- Complejidad Temporal y Espacial: Ambos algoritmos tienen una complejidad temporal y espacial cuadrática ($O(N^2)$) en algunos puntos debido a bucles anidados y matrices bidimensionales.
- Lógica de Intercambio Iterativo: El algoritmo de intercambio iterativo implementado en la clase Algoritmo es clave para mejorar la solución inicial y buscar soluciones más óptimas.
- Interfaz Gráfica y Manejo de Archivos: La inclusión de una interfaz gráfica (JFileChooser) facilita la interacción

del usuario al seleccionar archivos. El manejo de excepciones para IOException demuestra una consideración para posibles errores durante la lectura de archivos.

G. Aspectos a Mejorar

- Optimización de Complejidad: Explorar oportunidades para optimizar la complejidad temporal y espacial, especialmente en las partes críticas del código, podría mejorar la eficiencia del algoritmo.
- Implementación de PTorneo: La referencia a una clase PTorneo está presente, pero su implementación no se proporciona. La falta de detalles sobre cómo se realiza la generación del horario puede limitar la comprensión completa del algoritmo.
- Pruebas Empíricas: Realizar pruebas empíricas con conjuntos de datos variados podría proporcionar una evaluación más precisa del rendimiento y la cercanía al óptimo de los algoritmos.

REFERENCES

[1] https://www.aprenderaprogramar.com/index.php?option=com$_contentview = articleid = 646 : documentar - proyectos - java - con - javadoc - comentarios - simbolos - tags - deprecated - param - etc - cu00680bcatid = 68Itemid = 188