UNIVERSIDAD METROPOLITANA

Departamento de Gestión de Proyectos y Sistemas.
Estructuras de datos. BPTSP06.
Trimestre 2425-3.

Proyecto 1: Sopa de letras.

Contexto preliminar

En el siguiente proyecto, usted tendrá que realizar una **investigación documental** que le permita obtener información sobre el contexto del problema. En tal sentido, se sugiere que comience por realizar la siguiente lectura relativa a grafos, pero tome en cuenta que es solo un recurso inicial que debe ser complementado con la búsqueda autónoma de información por parte de los integrantes del equipo de trabajo:

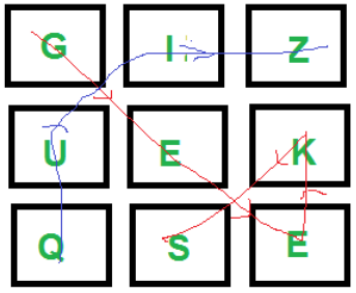
https://drive.google.com/file/d/1Q65Rh-Tx6qwJODUlsmWVfw_-bzqttwil/view?usp=sharing

Problema

Su equipo de trabajo ha sido contratado para desarrollar un programa que sea capaz de resolver una sopa de letras. Es decir, el programa deberá poder verificar que una lista de palabras se encuentra en un tablero de letras de 4x4. Las reglas para la formación de una palabra válida son muy sencillas:

I. La palabra encontrada debe estar compuesta por una secuencia de elementos adyacentes en el tablero. Dos elementos del tablero son adyacentes si están uno al lado del otro en el tablero, tanto horizontal como vertical o diagonalmente.
II. Para la formación de una palabra válida, solo se puede utilizar una vez cada elemento (letra) del tablero. Es decir, no se puede volver a utilizar un elemento del tablero para la formación de la misma palabra.
III. Una palabra válida debe tener al menos 3 letras.
IV. Una palabra válida debe existir en un diccionario.
V. No todas las palabras del diccionario se encontrarán en el tablero.
VI. Las palabras no llevan acento.

La siguiente figura muestra un ejemplo de palabras válidas (GEEKS, QUIZ) en dicho tablero.



Requerimientos funcionales

A. **Cargar archivo:** El usuario puede seleccionar a través de **JFileChooser** un archivo **txt** (archivo de texto plano) para ser cargado en el sistema, el cual contará con la información necesaria para la creación tanto del tablero como del diccionario. El programa deberá mostrar el tablero y el diccionario.

B. **Buscar palabras**: el usuario dará la orden al programa para que busque las palabras contenidas en el diccionario.

   a. El usuario podrá seleccionar entre dos métodos de búsqueda: DFS (Depth First Search) o BFS (Breadth First Search).

   b. El programa deberá indicar cuáles palabras se encuentran en el tablero.

   c. El programa, además, deberá mostrar el tiempo que tarda en encontrar todas las palabras en milisegundos.

C. **Buscar una palabra en específico:** el usuario podrá indicar una palabra en específico a ser buscada. Puede ser una palabra que no está contenida en el diccionario inicial. Si la encuentra, entonces esta debe ser agregada al diccionario.

D. **Mostrar árbol de recorrido BFS:** El sistema deberá mostrar una representación visual del árbol de recorrido BFS del grafo al buscar solo una palabra (solo cuando se utilice la función que responde al requerimiento C). El árbol deberá tener como raíz la primera letra de la palabra buscada. **Nota:** Puede hacer uso de librerías para dicha representación.

E. **Guardar Diccionario**: El usuario podrá solicitar al programa que actualice el archivo con los datos del diccionario cargado en memoria.

Requerimientos técnicos

1. La solución debe ser implementada (específicamente el denominado tablero) con base en un grafo no dirigido, que a su vez puede ser implementado mediante una **matriz de adyacencia** o una l**ista de adyacencia**.

2. Puede utilizar cualquier otra estructura auxiliar de ser necesario para mejorar los tiempos de respuesta del programa. Sin embargo, **NO podrá utilizar librerías para la implementación del tipo de dato abstracto**, solo podrá utilizar librerías para lo relativo a la representación gráfica del grafo (árbol de recorrido referido en el requerimiento E). Se sugiere utilizar GraphStream (https://graphstream-project.org/).

3. La aplicación debe ofrecer una interfaz gráfica al usuario. No se permite la entrada ni la salida de información por consola.

4. El programa debe poder cargar un archivo de texto para la lectura de datos. Para ello, es requerido el uso del componente **JFileChooser**.

5. Debe documentar el proyecto con **Javadoc**.

6. Junto al programa, cada equipo deberá presentar un **Diagrama de clases** *(arquitectura detallada)* que explique la solución obtenida.

Archivo de texto

El programa debe poder leer un archivo de texto en donde las palabras del diccionario se encuentran separadas por saltos de línea y encerradas entre las etiquetas “dic” y “/dic” y las 16 letras que conforman el tablero 4x4 se encuentran encerradas entre las etiquetas “tab” y “/tab”.

Puede tomar el siguiente ejemplo como referencia en lo relativo a la estructura del archivo a utilizar. Tome en cuenta que el archivo de prueba tendrá la misma estructura pero no los mismos datos:

dic
CHAO
OTRA
PATA
ORO
RATA
/dic
tab
C,H,A,O,X,A,T,M,P,R,R,A,A,O,R,O
/tab

Consideraciones

- Los equipos pueden tener como **máximo 3 personas.**
- Deberá entregar su carpeta de Proyecto NetBeans en un archivo ZIP en el aula del curso. Además, deberá cargar la dirección del repositorio.
- La carpeta deberá incluir un archivo llamado “readme”, el cual debe incluir una breve descripción del proyecto (una o dos oraciones que permitan entender el contenido sin consultar el enunciado), los nombres de los integrantes del grupo y la dirección web del repositorio de GitHub donde se desarrolló el proyecto.
- Los proyectos que no tengan interfaz gráfica, serán calificados con **0 (cero)**.

- Los proyectos que no tengan documentación interna alguna, serán calificados con **0 (cero)**.
- Los proyectos que implementen el tipo de dato abstracto usando librerías, serán calificados con **0 (cero)**.
- Los proyectos que sean iguales o parecidos, incluyendo aquellos que sean iguales o parecidos a los que se puedan encontrar en internet o generar con herramientas derivadas de ChatGPT, serán calificados con **0 (cero)**.
- Los programas que “no compilen” o “no corran”, serán calificados con **0 (cero)**
    - Si una entrega no incluye los archivos de proyecto de NetBeans, será considerado como que ***no compila***. El equipo debe estar seguro de que el archivo consignado contiene la totalidad del proyecto.
    - Si una entrega usa librerías en JAR y no incluye los archivos .jar en el zip de la entrega, será considerado como que ***no compila***.
    - Si un programa muestra una interfaz gráfica, pero la cierra antes de poder realizar ninguna acción útil, será considerado como que ***no corre.***
    - Si un programa muestra una interfaz gráfica de usuario sin funcionalidad alguna, será considerado como que ***no corre***.
- El enlace a GitHub en el *readme* ***debe*** conducir a un repositorio público válido. Los registros de Github que tengan todos los commits de un solo integrante o en un solo día, serán considerados como ***no válidos***, pues no reflejan la participación activa y significativa de todos los integrantes del equipo.
- Los proyectos **podrán** ser sometidos a defensa, es decir, el facilitador y/o preparadores convocarán al equipo para una revisión **solo** si lo considera necesario.
- Se podrá considerar funcionalidad adicional para puntos adicionales siempre y cuando se identifique claramente, mantenga los estándares de calidad del resto del programa y sea razonable para el problema planteado.

Criterios de Evaluación (Rúbrica)

- **Resolución de problemas de ingeniería**. Capacidad para comprender, definir y resolver problemas de análisis de ingeniería en el campo de estudio pertinente, con el uso de conocimientos básicos y avanzados de métodos analíticos modernos.
    - Comprensión y definición de problemas de ingeniería:
        - Identificación de Componentes Clave: Identifica y explica todos los componentes clave del problema (tablero, diccionario, reglas de adyacencia, uso único de letras) con absoluta claridad y precisión. **(10%)**

- Modelado como Grafo: Modela el tablero como un grafo de forma impecable y rigurosa, detallando con precisión la relación entre nodos (letras) y aristas (adyacencia, incluyendo horizontal, vertical, diagonal). **(10%)**

- Traducción de Reglas a Restricciones del Grafo: Explica de forma cristalina y detallada cómo todas las reglas del juego (uso único, longitud mínima, existencia en diccionario) se traducen en restricciones para la búsqueda en el grafo, mostrando una comprensión profunda de sus implicaciones algorítmicas. **(10%)**

- *Aplicación de Métodos Analíticos Modernos:*

  - Implementación de DFS y BFS: Implementa los algoritmos DFS y BFS de manera correcta, eficiente y robusta para la búsqueda de palabras en el grafo del tablero, respetando todas las reglas del juego (adyacencia, uso único de letras, longitud mínima). **(10%)**

  - Precisión en la Medición de Tiempo de Ejecución: Muestra el tiempo que tarda en encontrar todas las palabras en milisegundos de manera precisa, confiable y consistentemente correcta, indicando una buena implementación de la medición. **(5%)**

- *Resolución de Problemas con Conocimientos Básicos y Avanzados:*
  - Funcionalidad y Cumplimiento de Requerimientos: El programa es completamente funcional, robusto y eficiente, cumpliendo con todos y cada uno de los requerimientos funcionales solicitados (Cargar archivo, Buscar palabras con DFS/BFS, Buscar palabra específica, Mostrar árbol de recorrido BFS y Guardar diccionario). **(10%)**
  - Implementación de Estructuras de Datos (Grafo y Diccionario): La implementación del grafo subyacente para el tablero es modélica, robusta y eficiente. El manejo del diccionario (carga, búsqueda, adición, guardado) es impecable y optimizado. **(5%)**
  - Manejo de Restricción "Uso Único de Letras": La lógica para asegurar el "no usar la misma letra dos veces en la misma palabra" está implementada correctamente, de forma eficiente y sin fallos, utilizando mecanismos apropiados (ej., set de visitados en el camino actual). **(2,5%)**
  - Visualización del Árbol de Recorrido BFS: La visualización del árbol de recorrido BFS es clara, precisa, útil para el análisis y cumple perfectamente con el requisito, mostrando una comprensión profunda del proceso de búsqueda. **(5%)**
  - Calidad y Estructura del Código: El código fuente es de alta calidad, bien estructurado, exhaustivamente documentado (comentarios, nombres de

variables claros) y demuestra mejores prácticas de programación (modularidad, legibilidad). **(2,5%)**

- *Fiabilidad*: Capacidad para mantener un nivel especificado de prestaciones cuando se usa bajo ciertas condiciones:
    - *Madurez*: El programa no presenta fallas originadas por errores de programación, análisis o diseño. **(10%)**
    - *Tolerancia a fallos*: El programa responde adecuadamente al manejo inadecuado del usuario; es decir, mantiene su correcto funcionamiento aun cuando el usuario introduzca datos erróneos o manipule inadecuadamente las interfaces de usuario. **(10%)**
- *Usabilidad*: Capacidad del proyecto para ser entendido, aprendido, usado y al mismo tiempo, ser atractivo para el usuario, cuando se usa bajo condiciones específicas:
    - *Comprensibilidad*: El programa ofrece una interfaz de fácil comprensión, facilitando su aprendizaje y correcta utilización. El programa emite mensajes de alerta cuando se introducen valores erróneos. Existen elementos informativos que indican al usuario como operar el programa. **(5%)**
    - *Capacidad de ser atractivo*: El diseño de la interfaz de usuario, esto es: disposición de controles, esquema de colores, utilización de cajas de diálogo y demás elementos gráficos; es atractivo para el usuario. **(5%)**

Rúbrica Detallada

Resolución de problemas de ingeniería. Capacidad para comprender, definir y resolver problemas de análisis de ingeniería en el campo de estudio pertinente, con el uso de conocimientos básicos y avanzados de métodos analíticos modernos.

Criterio 1: Comprensión y Definición de Problemas de Ingeniería				
Sub-Criterio	Excelente (10%)	Satisfactorio (8%)	Básico (5%)	Insatisfactorio (0%)
Identificación de Componentes Clave	Identifica y explica todos los componentes clave del problema (tablero, diccionario, reglas de adyacencia, uso único de letras) con absoluta claridad y precisión.	Identifica la mayoría de los componentes clave, pero la explicación de alguno puede carecer de cierto detalle o claridad.	Identifica algunos componentes clave, pero su descripción es ambigua o incompleta.	No logra identificar los componentes esenciales del problema de la sopa de letras.
	Excelente (10%)	Satisfactorio (8%)	Básico (5%)	Insatisfactorio (0%)
Modelado como Grafo	Modela el tablero como un grafo de forma impecable y rigurosa, detallando con precisión la relación entre nodos (letras) y aristas (adyacencia, incluyendo horizontal, vertical, diagonal).	Modela la sopa de letras como un grafo, pero la explicación de la relación entre los nodos y aristas podría ser ligeramente menos completa o rigurosa.	Modela la sopa de letras como un grafo, pero el modelo es simplificado, incompleto o presenta algunas inconsistencias.	No modela la sopa de letras como un grafo, o su modelo es incorrecto o incoherente.
	Excelente (10%)	Satisfactorio (8%)	Básico (5%)	Insatisfactorio (0%)
Traducción de Reglas a Restricciones del Grafo	Explica de forma cristalina y detallada cómo todas las reglas del juego (uso único, longitud mínima, existencia en diccionario) se traducen en restricciones para la búsqueda en el grafo, mostrando una comprensión profunda de sus implicaciones algorítmicas.	Comprende las reglas principales del juego, pero puede tener pequeñas dificultades para traducir completamente todas las restricciones al modelo del grafo o al algoritmo de búsqueda, o la explicación es un poco menos detallada.	Comprende algunas reglas, pero tiene dificultades significativas para traducirlas a restricciones del grafo o entender sus implicaciones algorítmicas.	Demuestra una comprensión deficiente o errónea de las reglas del juego o de cómo estas afectan la búsqueda de palabras.
Criterio 2: Aplicación de Métodos Analíticos Modernos				
Sub-Criterio	Excelente (10%)	Satisfactorio (8%)	Básico (5%)	Insatisfactorio (0%)

Implementación de DFS y BFS	Implementa los algoritmos DFS y BFS de manera correcta, eficiente y robusta para la búsqueda de palabras en el grafo del tablero, respetando todas las reglas del juego (adyacencia, uso único de letras, longitud mínima).	Implementa DFS y BFS, pero con errores menores o inconsistencias que pueden afectar levemente la eficiencia o la correcta aplicación de alguna regla específica del juego.	La implementación de DFS y/o BFS contiene errores significativos o no aborda completamente las reglas fundamentales del juego.	No implementa correctamente DFS o BFS, o su implementación viola principios básicos de los algoritmos de búsqueda en grafos.
	Excelente (5%)	Satisfactorio (4%)	Básico (2%)	Insatisfactorio (0%)
Precisión en la Medición de Tiempo de Ejecución	Muestra el tiempo que tarda en encontrar todas las palabras en milisegundos de manera precisa, confiable y consistentemente correcta, indicando una buena implementación de la medición.	El tiempo de ejecución se muestra, pero podría haber pequeñas imprecisiones en su medición o la forma en que se presenta.	El tiempo de ejecución se muestra, pero es inconsistente, incorrecto o no se mide adecuadamente.	El tiempo de ejecución no se muestra o la medición es completamente errónea e inútil.
Criterio 3: Resolución de Problemas con Conocimientos Básicos y Avanzados				
Sub-Criterio	Excelente (10%)	Satisfactorio (8%)	Básico (5%)	Insatisfactorio (0%)
Funcionalidad y Cumplimiento de Requerimientos	El programa es completamente funcional, robusto y eficiente, cumpliendo con todos y cada uno de los requerimientos funcionales solicitados (Cargar archivo, Buscar palabras con DFS/BFS, Buscar palabra específica, Mostrar árbol de recorrido BFS y Guardar diccionario).	El programa es funcional en su mayor parte, cumpliendo con la mayoría de los requerimientos, pero puede presentar algunos errores menores o limitaciones aisladas en uno o dos requerimientos funcionales.	El programa presenta fallos significativos o inconsistencias en varios requerimientos funcionales clave, lo que lo hace incompleto o poco útil.	El programa presenta fallos críticos en la mayoría o todos los requerimientos funcionales, haciendo que la solución sea inutilizable o fundamentalmente defectuosa.
	Excelente (5%)	Satisfactorio (4%)	Básico (2%)	Insatisfactorio (0%)
Implementación de Estructuras de Datos (Grafo y Diccionario)	La implementación del grafo subyacente para el tablero es modélica, robusta y eficiente. El manejo del diccionario (carga, búsqueda, adición, guardado) es impecable y optimizado.	La implementación del grafo es adecuada, pero podría haber oportunidades para optimización o mejoras en la eficiencia. El manejo del diccionario es mayormente correcto, aunque con alguna pequeña ineficiencia.	La implementación del grafo o el manejo del diccionario son ineficientes, inconsistentes o presentan errores, afectando el rendimiento o la integridad de los datos.	La implementación del grafo o el manejo del diccionario son fundamentalmente defectuosos o incorrectos, impidiendo la funcionalidad básica del programa.
	Excelente (2,5%)	Satisfactorio (2,0%)	Básico (1%)	Insatisfactorio (0%)
Manejo de Restricción "Uso Único de Letras"	La lógica para asegurar el "no usar la misma letra dos veces en la misma palabra" está implementada correctamente, de forma eficiente y sin fallos, utilizando mecanismos apropiados (ej., set de visitados en el camino actual).	La lógica para el "uso único de letras" está implementada, pero podría ser menos eficiente o menos robusta en ciertos casos límite, o su solución es un poco más compleja de lo necesario.	La lógica para el "uso único de letras" es parcialmente implementada, ineficaz o presenta errores que permiten repeticiones no deseadas.	La lógica para el "uso único de letras" no está implementada o está implementada completamente incorrecta, violando una regla fundamental del problema.

	Excelente (5%)	Satisfactorio (4%)	Básico (2%)	Insatisfactorio (0%)
Visualización del Árbol de Recorrido BFS	La visualización del árbol de recorrido BFS es clara, precisa, útil para el análisis y cumple perfectamente con el requisito, mostrando una comprensión profunda del proceso de búsqueda.	La visualización del árbol BFS es funcional y representa el recorrido, pero podría ser menos óptima, estéticamente mejorable o carecer de algún detalle menor.	La visualización del árbol BFS es confusa, incompleta o presenta errores en la representación del recorrido.	La visualización del árbol BFS no se proporciona o es completamente incorrecta/inútil.
	Excelente (2,5%)	Satisfactorio (2,0%)	Básico (1%)	
Calidad y Estructura del Código	El código fuente es de alta calidad, bien estructurado, exhaustivamente documentado (comentarios, nombres de variables claros) y demuestra mejores prácticas de programación (modularidad, legibilidad).	El código es comprensible y funcional, pero la estructura o la documentación podrían mejorar ligeramente. Podría haber pequeñas inconsistencias en las buenas prácticas.	El código es difícil de entender, mal estructurado o carece de documentación adecuada. Muestra algunas malas prácticas de programación.	

Criterio de Evaluación	Subcriterio	Niveles de Desempeño	Descripción	Puntuación
Fiabilidad	Madurez	Excelente	El programa no presenta fallas y mantiene su funcionamiento incluso con entradas incorrectas.	10%
		Satisfactorio	El programa presenta fallas menores o errores en casos límite, pero se mantiene estable en general.	7%
		Necesita mejorar	El programa presenta fallas frecuentes y genera errores por consola.	3%
		Inaceptable	Las fallas del programa son tan graves que no puede ser usado de ninguna manera útil.	0%
	Tolerancia a fallos	Excelente	El programa maneja entradas incorrectas y errores de usuario de manera robusta y sin interrupciones.	10%
		Satisfactorio	El programa maneja la mayoría de las entradas incorrectas, pero puede bloquearse en algunos casos.	7%
		Necesita mejorar	El programa se bloquea o falla con entradas incorrectas o errores de usuario.	3%
		Inaceptable	No hay acciones que no generen errores.	0%

Criterio de Evaluación	Subcriterio	Niveles de Desempeño	Descripción	Puntuación

Usabilidad	Comprensibilidad	Excelente	La interfaz es intuitiva y fácil de usar, pudiendo entenderse cómo realizar cada una de las funcionalidades sin requerir explicación externa.	5%
		Bueno	La interfaz es generalmente intuitiva, pero algunos controles pueden ser confusos en cuanto a cómo deberían ser usados.	4%
		Satisfactorio	Aunque la mayoría de los controles son confusos, la interfaz logra comunicar efectivamente cómo llegar a cada una de las funcionalidades pedidas.	3%
		Necesita mejorar	Algunas de las funcionalidades se pueden realizar intuitivamente, pero es difícil encontrar cómo deben realizarse las demás.	2%
		Deficiente	La disposición de la interfaz es confusa, interfiriendo con la usabilidad general del programa	1%
		Inaceptable	El programa no tiene una interfaz entendible.	0%
	Capacidad de ser atractivo	Excelente	La interfaz tiene un diseño atractivo y moderno, con una buena disposición de controles y esquemas de colores.	5%
		Bueno	La interfaz contiene elementos que pueden dificultar la lectura, pero está generalmente bien diseñada.	4%
		Satisfactorio	La interfaz es funcional, pero la disposición de los controles y el diseño general es mejorable.	3%
		Necesita mejorar	El diseño de la interfaz interfiere ligeramente con la usabilidad y legibilidad del programa.	2%
		Deficiente	El diseño de la interfaz es descuidado e interfiere seriamente con la usabilidad y legibilidad del programa.	1%
		Inaceptable	El diseño de la interfaz es tan descuidado que el programa es inutilizable.	0%