UNIDAD TEMÁTICA 4 – GRAFOS DIRIGIDOS

PRACTICOS DOMICILIARIOS INDIVIDUALES - 7

Un importante estudio de proyectos de construcción tiene en cartera numerosos proyectos de gran porte. Cada proyecto está formado por una gran cantidad de tareas, y a su vez esas tareas tienen un orden de precedencia entre sí, es decir, si la tarea "A" precede a la tarea "B", antes de hacer "B" debe haberse completado "A".
El gerente de cada proyecto necesita conocer una secuencia de precedencias válida para la ejecución del proyecto. El analista ha definido que, si el problema es representado por un grafo dirigido en el que los vértices son las tareas y las aristas son las precedencias, un sencillo y eficiente algoritmo puede devolver, dada la tarea final, la lista de todas las tareas que es necesario realizar previamente, en un orden de precedencia válido.

PARTE 1: Funcionalidades a desarrollar

La clase principal se denomina "**Proyecto**", y tiene su correspondiente método "**main**".
Desarrollar e implementar:

- **En TGrafoDirigido: public LinkedList<Tarea> ordenParcial ()**
- **Implementar el correspondiente método – recursivo - a nivel del vértice.**
- **En TGrafoDirigido: public void listarTareas (LinkedList orden) / muestra por consola la secuencia de tareas**

PARTE 2: PROGRAMA

1. En el método "main" de la clase "**Proyecto**", implementa lo necesario para aplicar los TDA y métodos desarrollados.
2. Cargar una linked list para los vértices que representan las tareas, a partir del archivo **"tareas.txt"**
3. Cargar una lista de aristas (TAristas) que representan las precedencias, a partir del archivo **"precedencias.txt"**
4. Instanciar un grafo dirigido que represente el proyecto, con las listas generadas en 2 y 3
5. Invocar al método "**ordenParcial**", para ***todo el proyecto***.
6. Invocar al método "**listarTareas**" para mostrar por consola el resultado obtenido.
7. Emitir un archivo "**orden.txt**" que contenga la secuencia de tareas, con una tarea en cada línea. Este archivo debe contener un nombre de tarea por cada línea, de acuerdo al orden parcial obtenido (culminando con la tarea **"*Fin*"** que indica el fin del proyecto).
8. Repetir los pasos 3 a 7, pero cargando el archivo de aristas **"precedencias2.txt",** y emitiendo un nuevo archivo de salida **"orden2.txt"**

PARTE 3: TEST CASES

Desarrollar los **casos de prueba** ("test cases") necesarios para verificar la corrección de los métodos implementados.

ARCHIVOS DE ENTRADA

- "**tareas.txt**": lista de tareas del proyecto, sin un orden en particular. El archivo contiene los datos de una tarea por línea, separados por comas

  COD_TAREA, TIEMPO_TAREA
- "**precedencias.txt**": cada línea representa la relación de dependencia entre dos tareas del proyecto. En cada línea se indica el código de tarea precedente, seguido del código de tarea consecuente, separados por una coma

  COD_TAREA _A , COD_TAREA _B

NOTA IMPORTANTE: LOS ARCHIVOS **PUEDEN** CONTENER ERRORES. SE **DEBEN** IMPLEMENTAR LAS PRECAUCIONES BASICAS PARA EVITAR QUE EL SISTEMA COLAPSE ANTE ERRORES O INCONGRUENCIA DE DATOS.

RUBRICA DE CALIFICACIÓN: se utilizarán los siguientes criterios en la evaluación del trabajo remitido:

1. **EJECUCIÓN: 30%**
    - Correcta lectura de los archivos de datos y creación de las estructuras definidas.
    - Consideraciones de seguridad con respecto a los datos – chequeos de consistencia y corrección realizados
    - Medidas de seguridad para verificar que el algoritmo diseñado puede efectivamente invocarse
    - Ejecución en tiempo razonable
    - Emisión de resultados correctos, en los archivos de salida requeridos.

2. **DESARROLLO de FUNCIONALIDAD REQUERIDA 35%**
    - Cumplimiento de las interfaces publicadas
    - Corrección del método solicitado
    - Implementación de chequeos necesarios para cumplimiento de la funcionalidad requerida
    - Programa principal, generación correcta del archivo de salida.

3. **CALIDAD DEL CÓDIGO (10 %)**
    - Nombres de variables y métodos
    - Aplicación correcta y rigurosa del paradigma de programación orientada a objetos
    - Invocación racional y eficiente de métodos y funciones
    - Encapsulación, modularidad
    - Utilización apropiada de clases de colecciones y genéricos necesarios

4. **PRUEBAS DE UNIDAD 25%.**
    - Calidad de los tests desarrollados, todas las condiciones normales y de borde, se testean todos los métodos, uso del enfoque inductivo en los tests.