**REPÚBLICA BOLIVARIANA DE VENEZUELA MINISTERIO DE EDUCACIÓN SUPERIOR UNIVERSIDAD JOSÉ ANTONIO PÁEZ FACULTAD DE INGENIERÍA**

**Evaluacion #4**

**Programacion Entera y Lineal**

**Profesor:**

Ing.: María García

**Alumno:**

Omaris Carreño CI: 27.927.570

Jefferson Ayala CI: 29.694.682

José faneite CI: 14024801

Santiago Martínez CI: 27502148

**Sección: 108C1**

**San Diego, Estado Carabobo, noviembre**

**Resume**: La presentre documentacion se realiza para desallorar y explicar dos ejercicio de la materia Metodo Cuantativo con la Profesora Maria Gracia. Los ejecicios estan dividido en II Parte.

**PARTE 1**

Eres un desarrollador de software y tu tarea es optimizar la planificación de asignación de tareas en un proyecto. Tienes un conjunto de tareas con sus duraciones y dependencias entre ellas.

El objetivo es minimizar la duración total del proyecto asignando tareas a los recursos disponibles, garantizando que las dependencias se cumplan. Debes utilizar programación entera lineal para resolver en este problema.

Implementa un programa en Java utilizando la biblioteca OR-Tools de Google para resolver este problema de programación entera lineal. El programa debe tener las siguientes características:

1. Definir variables de decisión: Representa la asignación de cada tarea a un recurso como variables de decisión.

2. Establecer la función objetivo: Minimiza la duración total del proyecto sumando las duraciones de las tareas ponderadas por las asignaciones.

3. Agregar restricciones: Asegura que las dependencias entre las tareas se cumplan y que los recursos disponibles se utilicen de manera óptima.

4. Resolver el problema de programación entera lineal utilizando OR-Tools.

5. Mostrar la planificación óptima de asignación de tareas y la duración total del proyecto.

**Diagrama de flujo**

1. Inicio del programa.
2. Carga de la biblioteca OR-Tools.
3. Obtención de datos de entrada:
   * Se obtiene el número de tareas (numTareas).
   * Se obtiene el número de recursos (numRecursos).
   * Se obtienen las duraciones de las tareas en forma de cadena (duracionesStr).
   * Se obtienen las dependencias entre tareas en forma de cadena (dependenciasStr).
4. Verificación de la longitud de dependenciasStr:
   * Se verifica si la longitud de dependenciasStr es un múltiplo de 2.
   * Si no es un múltiplo de 2, se muestra un mensaje de error y se sale del programa.
5. Creación del array de dependencias:
   * Se crea un array bidimensional llamado dependencias para almacenar las dependencias entre tareas. Los pares de dependencias se extraen de dependenciasStr.
6. Creación del solucionador de programación lineal entera (MPSolver):
   * Se crea un solucionador de programación lineal entera llamado "AsignacionTareas" con el tipo de problema "CBC\_MIXED\_INTEGER\_PROGRAMMING".
7. Definición de variables de decisión:
   * Se crean las variables de decisión llamadas asignacionTareas, que representan la asignación de cada tarea a un recurso. Estas variables son binarias (0 o 1) y se crean en una matriz.
8. Definición de la función objetivo:
   * Se define la función objetivo para minimizar la duración total del proyecto. Los coeficientes de la función objetivo se configuran en función de las duraciones de las tareas.
9. Agregación de restricciones de dependencia:
   * Se agregan restricciones para garantizar que las dependencias entre tareas se cumplan. Se recorre el array dependencias y se crea una restricción para cada dependencia. Las restricciones están relacionadas con las variables de asignación de tareas.
10. Resolución del problema:
    * Se resuelve el problema de programación lineal entera utilizando el solucionador.
    * Se verifica si se encontró una solución óptima.
11. Impresión de resultados:
    * Si se encontró una solución óptima, se muestra la planificación óptima de asignación de tareas y la duración total del proyecto. Esto se hace recorriendo las variables de asignación de tareas y sumando las duraciones de las tareas asignadas a cada recurso.
    * Si no se encontró una solución óptima, se muestra un mensaje indicando que no se encontró una solución óptima.
12. Fin del programa.

El código resuelve un problema de asignación de tareas con duraciones y dependencias utilizando la biblioteca OR-Tools de Google.

1. \*\*Carga de la Biblioteca OR-Tools:\*\*

- Se carga la biblioteca OR-Tools utilizando `Loader.loadNativeLibraries();`.

2. \*\*Obtención de Datos de Entrada:\*\*

- Se obtienen los siguientes datos de entrada desde campos de texto o similar:

- `numTareas`: El número de tareas.

- `numRecursos`: El número de recursos.

- `duracionesStr`: Las duraciones de las tareas como una cadena de texto separada por comas.

- `dependenciasStr`: Las dependencias entre tareas como pares de números enteros separados por comas.

3. \*\*Creación del Array de Dependencias:\*\*

- Se verifica que la longitud de `dependenciasStr` sea un múltiplo de 2 para asegurarse de que los pares estén completos.

- Se crea un array bidimensional llamado `dependencias` para almacenar las dependencias. Los pares de dependencias se extraen de `dependenciasStr` y se almacenan en este array.

4. \*\*Creación del Solucionador de Programación Lineal Entera:\*\*

- Se crea un solucionador de programación lineal entera (`MPSolver`) llamado "AsignacionTareas" con el tipo de problema "CBC\_MIXED\_INTEGER\_PROGRAMMING".

5. \*\*Definición de Variables de Decisión:\*\*

- Se crean variables de decisión llamadas `asignacionTareas` que representan la asignación de cada tarea a un recurso. Estas variables son binarias (0 o 1) y se crean en una matriz.

6. \*\*Definición de la Función Objetivo:\*\*

- Se define la función objetivo para minimizar la duración total del proyecto. Los coeficientes de la función objetivo se configuran en función de las duraciones de las tareas.

7. \*\*Agregación de Restricciones de Dependencia:\*\*

- Se agregan restricciones para garantizar que las dependencias entre tareas se cumplan. Se recorre el array `dependencias` y se crea una restricción para cada dependencia. Las restricciones están relacionadas con las variables de asignación de tareas.

8. \*\*Resolución del Problema:\*\*

- Se resuelve el problema de programación lineal entera utilizando el solucionador.

- Se verifica si se encontró una solución óptima.

9. \*\*Impresión de Resultados:\*\*

- Si se encontró una solución óptima, se muestra la planificación óptima de asignación de tareas y la duración total del proyecto. Esto se hace recorriendo las variables de asignación de tareas y sumando las duraciones de las tareas asignadas a cada recurso.

- Si no se encontró una solución óptima, se muestra un mensaje indicando que no se encontró una solución óptima.

El código obtiene datos de entrada, crea un modelo de programación lineal entera, define variables de decisión y restricciones, resuelve el problema y muestra la solución óptima, si es posible. El resultado es la asignación óptima de tareas a recursos y la duración total del proyecto.

Como introducir los datos en el Panel:

1. **Número de Tareas:** Ingresa un número entero que represente la cantidad de tareas, por ejemplo, "5".
2. **Número de Recursos:** Ingresa un número entero que represente la cantidad de recursos, por ejemplo, "3".
3. **Duraciones (separadas por comas):** Ingresa las duraciones de las tareas como una lista de números enteros separados por comas, donde cada número representa la duración de una tarea. Por ejemplo, "3,2,4,5,1" para cinco tareas.
4. **Dependencias (pares separados por comas):** Ingresa las dependencias entre tareas como pares de números enteros separados por comas. Cada par debe representar que una tarea depende de otra. Por ejemplo, "0,1,1,2,3,4" indica que la tarea 0 depende de la tarea 1, la tarea 1 depende de la tarea 2 y la tarea 3 depende de la tarea 4.

* Los números deben ser enteros.
* Los elementos de la lista de duraciones deben estar separados por comas.
* Los pares de dependencias deben estar separados por comas y formar una secuencia válida de dependencias.

**PARTE II**

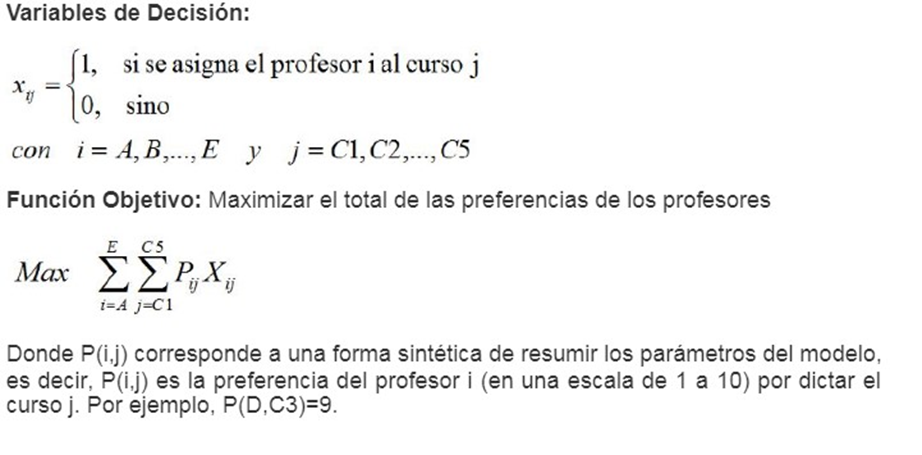
Una universidad está programando las clases para el próximo semestre académico

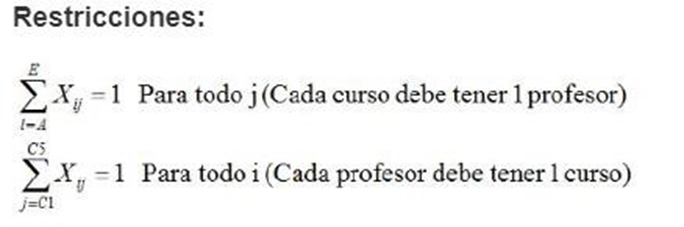
y requiere buscar la mejor asignación posible de profesores a los distintos cursos que se deben dictar.

Considere que existen 5 profesores: A, B, C, D, E y 5 cursos (asignaturas): C1, C2, C3, C4, C5. Adicionalmente, los profesores han manifestado sus preferencias por dictar los distintos cursos en una escala de 1 a 10, donde 10 es la máxima puntuación y 1 la mínima puntuación o preferencia. Se asume que cada profesor es apto para dictar cualquier curso, Independiente del puntaje de su preferencia. La siguiente tabla resume las puntuaciones que asigna cada profesor:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **CURSOS** |  |  | **PROFESORES** |  |  |  |
| **A** | **B** | **C** |  | **D** | **E** |
| **C1** | 5 | 8 | 5 |  | 9 | 7 |
| **C2** | 7 | 2 | 3 |  | 6 | 8 |
| **C3** | 9 | 10 | 8 |  | 9 | 8 |
| **C4** | 8 | 7 | 9 |  | 7 | 8 |
| **C5** | 6 | 9 | 9 |  | 10 | 5 |

Se ha establecido como criterio que cada profesor debe dictar sólo un curso y a la vez que cada curso obviamente debe tener un profesor. En base a lo anterior se desea encontrar la asignación de profesores que maximice el total de las preferencias.





Ejercicio Parte II

1. Definición de la Matriz de Preferencias:

• Se define una matriz bidimensional llamada preferencias que almacena las puntuaciones de preferencia de los profesores por cada curso. Cada fila representa a un profesor (A, B, C, D, E), y cada columna representa un curso (C1, C2, C3, C4, C5). Las puntuaciones van de 1 (mínima preferencia) a 10 (máxima preferencia).

2. Llamada a la Función de Asignación:

• Se llama a la función asignarProfesoresCursos para encontrar la asignación óptima de profesores a cursos en base a las preferencias.

3. Inicialización de Variables:

• En la función asignarProfesoresCursos, se inicializan las siguientes variables:

• numProfesores: Almacena el número de profesores (5 en este caso).

• numCursos: Almacena el número de cursos (5 en este caso).

• asignacion: Un arreglo que representa la asignación de profesores a cursos. Se inicializa con valores -1 para indicar que ningún profesor ha sido asignado inicialmente.

4. Bucle de Asignación de Cursos:

• Se inicia un bucle que recorre todos los cursos uno por uno.

5. Inicialización de Variables para la Evaluación de Preferencias:

• Se inicializan las siguientes variables para rastrear al mejor profesor y la máxima puntuación:

• mejorProfesor: Almacena el índice del profesor con la máxima preferencia para el curso actual. Inicializado en -1.

• maxPuntuacion: Almacena la máxima puntuación encontrada hasta ahora para el curso actual. Inicializado en -1.

6. Bucle de Evaluación de Preferencias de Profesores:

• Se inicia un bucle que recorre todos los profesores uno por uno.

7. Verificación de Elegibilidad de Profesor:

• Se verifica si el profesor aún no ha sido asignado a un curso (asignacion[profesor] == -1) y si su preferencia para el curso actual es mayor que la puntuación máxima encontrada hasta ahora.

8. Actualización de la Asignación si se Encuentra un Mejor Profesor:

• Si se cumple la condición anterior, se actualizan las variables mejorProfesor y maxPuntuacion con los valores actuales.

9. Salida del Bucle de Evaluación de Preferencias:

• Se sale del bucle de evaluación de preferencias después de evaluar todas las preferencias de los profesores para el curso actual.

10. Asignación del Profesor al Curso:

• Si se encontró un profesor adecuado para el curso actual (mejorProfesor != -1), se asigna ese curso al profesor en el arreglo asignacion.

11. Fin del Bucle de Asignación de Cursos:

• Se sale del bucle de asignación de cursos después de asignar todos los cursos a los profesores.

12. Mostrar Resultados:

• Finalmente, se muestra en la consola la asignación de profesores a cursos, indicando qué profesor se asigna a qué curso.

En resumen, el programa se encarga de asignar profesores a cursos según sus preferencias para maximizar la puntuación total de preferencias. Esto se logra evaluando las preferencias de cada profesor para cada curso y asignando el curso al profesor con la preferencia más alta disponible. El resultado es la asignación óptima de profesores a cursos.