Proyecto 2: Más Programación Dinámica

I. Descripción General

Este proyecto permitirá la ejecución de varios algoritmos clásicos de Programación Dinámica. Cada algoritmo será implementado con un programa totalmente independiente, pero habrá un programa principal o menú que los controla. Todas las interfaces serán gráficas. La salida será mostrada en la forma de una presentación `Beamer`. Toda la programación debe realizarse en C sobre Linux. No se pueden cambiar las especificaciones de este documento.

II. Entrada y Salida

La interacción con el usuario se hará por medio de interfaces gráficas que deben ser desarrolladas con `GTK` y `Glade`. Debe haber un estilo uniforme en el *look & feel* de todas las interfaces. Cada programa tendrá la posibilidad de grabar archivos con los datos particulares de cada problema ingresado por el usuario, para que puedan ser cargados de nuevo (y posiblemente editados) posteriormente. El formato de estos archivos queda a discreción de cada grupo de trabajo. Se espera que, el día de la revisión, cada grupo cuente con bastantes archivos de pruebas para mostrar las capacidades de su proyecto.

En todos los casos, la salida de los programas será una presentación `Beamer`, lo que implica generar un archivo LaTeX, que será compilado internamente desde el mismo programa usando `pdflatex`. El archivo PDF resultado del paso anterior será desplegado en modo presentación (*full screen*) por su programa usando `evince` o algo equivalente.

III. Menú Principal

Habrá un programa principal que desplegará un menú con todos los algoritmos de Programación Dinámica disponibles. Se quiere que sea visualmente agradable, bien acabado y profesional. Para cada opción ofrecida debe aparecer una "burbujita" con una descripción general cuando el cursor se pose sobre ellos (*tooltip*).

Este menú no hará más que servir de interfaz para mandar a ejecutar el programa seleccionado, usando `system()`. Se pueden tener simultáneamente múltiples ejecuciones de los programas disponibles.

Para este proyecto se deben implementar los siguientes algoritmos:

- Rutas más cortas (Algoritmo de Floyd)
- Problema de la Mochila
- Reemplazo de Equipos
- Árboles Binarios de Búsqueda Óptimos
- Series Deportivas
- Multiplicación de Matrices

Las siguientes secciones de este documento explican lo que se requiere de cada uno de estos programas.

IV. Problema de las rutas más cortas

Usando el algoritmo de Floyd estudiado en clases, este programa resolverá el problema de encontrar las rutas más cortas entre cualquier par de nodos de un grafo con ponderaciones en los arcos. Se espera que la interfaz gráfica sea lo más flexible posible. El usuario debe proporcionar la cantidad de nodos del grafo y las distancias directas entre ellos. Debe de haber un mecanismo para indicar el caso en que las distancias son infinitas (*i.e.*, no ruta directa) entre 2 nodos. Se sugiere que ese sea el valor por defecto para todos los nodos al inicio (excepto la diagonal de la tabla que tendrá siempre ceros). Para este proyecto, el usuario podrá indicar entre 1 y 10 nodos (si el grupo lo desea este límite podría ser más alto).

La salida será una presentación `Beamer` de gran calidad (con tablas, colores, dibujos, imágenes, gráficos, etc.) que incluirá siempre lo siguiente:

- Portada identificando al grupo de trabajo, al semestre del curso, y al algoritmo de Floyd.
- Una explicación general del algoritmo y algunos datos de su inventor.
- Grafo correspondiente al problema original (investigar cual *package* de LaTeXes apropiado para esto).
- Tabla inicial de distancias tal como las ingresó el usuario (`D(0)`).
- Tabla final de distancias mínimas entre cualquier par de nodos (`D(n)`).
- Tabla final de rutas óptimas entre cualquier par de nodos (`P(n)`).
- Lista de todas las rutas óptimas entre todos los pares de nodos.

Si el usuario lo solicita en la interfaz gráfica, también se crearán *slides* con todas las tablas `D(k)` y `P(k)` intermedias.

Trabajo extra opcional 1: el usuario podrá cambiar los nombres de los nodos a voluntad.

Trabajo extra opcional 2: conforme se meten los datos de distancias entre nodos, el programa despliega en la interfaz gráfica el estado actual del grafo. Se sugiere revisar "Cairo".

V. Problema de las mochila

Usando los algoritmos vistos en clase, este programa resolverá el "problema de la mochila", en sus versiones 1/0, *bounded* y *unbounded*. Se espera que la interfaz gráfica sea lo más flexible posible. El usuario debe proporcionar la capacidad máxima de la mochila (entre 0 y 20 como mínimo), la cantidad

de objetos a considerar (entre 1 y 10 como mínimo), y para cada objeto su costo, valor y cantidad disponible (incluido el caso de que esta cantidad fuera infinito).

La salida será una presentación `Beamer` de gran calidad (con tablas, colores, dibujos, imágenes, gráficos, etc.) que incluirá siempre lo siguiente:

- Portada identificando al grupo de trabajo, al semestre del curso, y al Problema de la Mochila.
- Una explicación general del algoritmo y algunos datos de su importancia.
- Tabla inicial de los datos que ingresó el usuario.
- Problema planteado en forma matemática.
- Tabla final del análisis (incluyendo los colores rojo y verde, y los valores de las variable de cada columna).
- Respuesta final en forma matemática.

Trabajo extra opcional 3: incluir un *slide* extra que muestre como se lee la respuesta de la tabla final (algoritmo visto en clases).

VI. Reemplazo de Equipos

Usando los algoritmos vistos en clase, este programa resolverá el "problema de reemplazo de equipos". Se espera que la interfaz gráfica sea lo más flexible posible. El usuario debe proporcionar el costo inicial del equipo, el plazo del proyecto (1 a 30), la vida útil del equipo (1 a 10), y para cada unidad de tiempo de uso del equipo el precio de reventa y el costo del mantenimiento. Opcionalmente, podría indicarse la ganacia recibida por usar el equipo en cada unidad de su tiempo de vida útil.

La salida será una presentación `Beamer` de gran calidad (con tablas, colores, dibujos, imágenes, gráficos, etc.) que incluirá siempre lo siguiente:

- Portada identificando al grupo de trabajo, al semestre del curso, y al Problema de Reemplazo de Equipos.
- Una explicación general del algoritmo y algunos datos de su importancia.
- Tabla inicial de los datos que ingresó el usuario.
- Tabla final del análisis.
- Todos los planes de reemplazos óptimos posibles.

Trabajo extra opcional 4: incluir *slides* extra con cada solución óptima mostrada como un grafo (saltos de rana).

Trabajo extra opcional 5: manejar ganancias asociadas a cada unidad de tiempo del uso del equipo (maximizar ganancia).

Trabajo extra opcional 6: manejar **inflación**, solicitándole al usuario un índice de inflación que será constante a lo largo de todo el proyecto pero que se aplicará a cada unidad de tiempo.

VII. Árboles Binarios de Búsqueda Óptimos

Usando el algoritmo visto en clases, este programa encontrará el árbol binario de búsqueda óptimo que corresponda a una colección de datos. Se espera que la interfaz gráfica sea lo más flexible posible. El usuario debe proporcionar la cantidad de llaves (a lo más 12), el valor de cada una de ellas (hileras), y un cantidad asociada a cada llave (que podría ser o no una probabilidad). Estas cantidades se normalizan a numeros reales entre 0 y 1 que sumen 1.0 antes de empezar el proceso. Note que las llaves no necesariamente sera ingresadas ordenadas.

La salida será una presentación `Beamer` de gran calidad (con tablas, colores, dibujos, imágenes, gráficos, etc.) que incluirá siempre lo siguiente:

- Portada identificando al grupo de trabajo, al semestre del curso, y al Problema de Árboles Binarios de Búsqueda Óptimos.
- Una explicación general del algoritmo y algunos datos de su importancia.
- Tabla inicial de los datos que ingresó el usuario. Se muestran tambien las probabilidades calculadas.
- Tabla **A** y tabla **R**.
- Dibujo del árbol respectivo.

VIII. Series Deportivas

Usando el algoritmo visto en clases, este programa calculará la probabilidad de que un equipo gane una serie deportiva pactada a "mejor de N", dado que se conocen las probabilidades de los partidos individuales, ya sea como locales o como visita. El usuario debe proporcionar la cantidad máxima de partidos de la serie (número impar que vale a lo más 15), las probabilidades de que el equipo de interés gane en casa (p_h) y de visita (p_r), y el formato de la serie (orden de los juegos en casa o de visita - cualquier combinación es válida, empezando en cualquier lugar). Nótese que la probabilidad de que el otro equipo gane de visita (q_r) y en casa (q_h) son los complementos de las probabilidades ya mencionadas.

La salida será una presentación `Beamer` de gran calidad (con tablas, colores, dibujos, imágenes, gráficos, etc.) que incluirá siempre lo siguiente:

- Portada identificando al grupo de trabajo, al semestre del curso, y al Problema de Series Deportivas.
- Una explicación general del algoritmo y algunos datos de su importancia.
- Datos iniciales que ingresó el usuario.
- Tabla de probabilidades.

IX. Multiplicación de Matrices

Usando el algoritmo visto en clases, este programa determinará la manera óptima de asociar una secuencia de matrices a ser multiplicadas de tal manera que se minimice el número de multiplicaciones individuales de entradas de las matrices. El usuario debe proporcionar la cantidad máxima N de matrices a ser multiplicadas (a lo más 15), y las $N + 1$ dimensiones

correspondientes (d_0 a d_N). Recuerde que las dimensiones de la $i-$ésima matriz son $d_{i-1} \times d_i$. Todas las matrices se llamarán A y se distinguirán por el subíndice.

La salida será una presentación `Beamer` de gran calidad (con tablas, colores, dibujos, imágenes, gráficos, etc.) que incluirá siempre lo siguiente:

- Portada identificando al grupo de trabajo, al semestre del curso, y al Problema de Multiplicación de Matrices.
- Una explicación general del algoritmo y algunos datos de su importancia.
- Datos iniciales que ingresó el usuario.
- Tablas de total de multiplicaciones y de colocación de paréntesis.
- Hilera que muestre con paréntesis como asociar las multiplicaciones.

Trabajo extra opcional 7: el usuario podrá indicar un nombre base diferente para las matrices (*e.g.* "Matriz"), todas se distinguen por el subíndice respectivo.

Trabajo extra opcional 8: mostrar un árbol de cómo se deben multiplicar las matrices.

X. Fecha de Entrega

Demostraciones en clase el **Miércoles 28 de Octubre**. Mande además un `.tgz` con todo lo necesario (fuentes, makefile, readme, etc.) a `torresrojas.cursos@gmail.com`. Ponga como subject: `I.O. - Proyecto 2 - Fulano - Mengano`, donde Fulano y Mengano son los 2 miembros del grupo.

Mucha suerte...