Taller de Programación II

Trabajos del Segundo Cuatrimestre de 2014

1 Introducción

La Universidad de Buenos Aires ha profundizado en los últimos años su política de inclusión de tecnologías para el fortalecimiento de las prácticas educativas en la Universidad. Para seguir fortaleciendo esta línea de acción, la UBA ha aprobado en el mes de mayo de 2014 una nueva convocatoria del programa UBATIC.

El programa intenta acompañar y fortalecer aquellas iniciativas innovadoras desarrolladas por los docentes universitarios que fomenten una inclusión genuina de las TIC y una aproximación al campo de las nuevas tecnologías que, superando las visiones técnico-instrumentales, favorezca una mirada reflexiva, crítica, creativa y responsable de su utilización como así también el fortalecimiento del trabajo en redes y la construcción compartida de conocimiento.

En el marco de este programa, la FIUBA desarrolla un proyecto institucional que persigue la articulación entre cátedras y departamentos de la facultad, que comprende las siguientes líneas:

- Diseño y desarrollo de materiales y/o recursos para la enseñanza (simulaciones, infografías multimedia, murales interactivos, animaciones, digitalización de recursos, bancos de imágenes, etc.).
- Creación de software y aplicaciones para favorecer la comprensión de temas disciplinares complejos, el desarrollo de competencias digitales del siglo XXI y/o para el enriquecimiento de la enseñanza (simuladores para la enseñanza, juegos, aplicativos de cálculos, videojuegos educativos, aplicaciones de realidad aumentada, entornos inmersivos u otras para el trabajo con dispositivos móviles o tabletas digitales, etc.).

Para este proyecto, la facultad estableció un repositorio de objetos de enseñanza y aprendizaje, y desde esta asignatura, pretendemos aportar aplicaciones que permitan la comprensión, ejercitación y autoevaluación de aprendizajes y de habilidades para la resolución de problemas, en temas disciplinares (tecnologías básicas y avanzadas) y de ciencias básicas. Con ese propósito, el cuatrimestre pasado ya se ha desarrollado una aplicación para la asignatura Computación, de formación básica para la mayoría de las carreras de la facultad, y se ha seleccionado a dos de los mejores trabajos para integrarlos al mencionado repositorio.

2 Objetivos

Desarrollar una aplicación para la enseñanza y aprendizaje de temas de alguna asignatura de la facultad.

La aplicación deberá comprender la resolución detallada y justificada de problemas característicos de algún tema de una asignatura, debiendo contemplarse dos modalidades de ejecución:

- **Modo aprendizaje**: resolución paso a paso con avance a solicitud del aprendiz (a siguiente paso o a resultado final).
- **Modo autoevaluación**: resolución paso a paso con elaboración guiada del resultado de cada paso a cargo del aprendiz, y confirmación de la corrección del resultado con opción a reintentar, en caso de que fuere incorrecto, o a ver el resultado correcto.

El aprendiz debe poder optar por la modalidad de ejecución, y asimismo, por la posibilidad de **persistir** los pasos que transite cualquiera sea la modalidad de ejecución.

Cada grupo de trabajo deberá proveer, en la aplicación que desarrolle, tantas variantes o técnicas para resolver problemas cuantos integrantes tenga el grupo.

3 Temas

El tema que desarrolle cada grupo estará determinado por el tutor que se le asigne, quien a su vez determinará requerimientos más detallados del mismo.

Se especifica para cada tema, un conjunto de variantes o técnicas entre las que cada grupo deberá escoger, según el número de sus integrantes y con la aprobación del tutor.

3.1 Grafos (Patricia)

Armado de grafo, orientado o no, con estas operaciones:

- Alta/baja de vértice
- Alta/baja de arco
- Borrado de grafo
- Grafo aleatorio dada la cantidad de vértices
- Ponderación de arcos

Una vez armado el grafo, se debe poder efectuar según especificaciones[1]:

- Recorrido en profundidad (puede ser algoritmo de Tarjan u otro)
- Recorrido en anchura
- Prueba de aciclidad

Si el grafo es orientado, además:

- Si es acíclico, recorrido topológico en anchura y en profundidad
- Obtención de cerradura transitiva
- Obtención de componentes fuertemente conexas

Si se trata de un grafo orientado con aristas ponderadas:

- Algoritmo de caminos mínimos con un mismo origen (Dijkstra). Debe verificarse que las ponderaciones sean no negativas.
- Algoritmo de caminos mínimos entre todos los pares de nodos (Floyd)
- Algoritmo de flujos máximos (Ford-Fulkerson)

Si el grafo es no orientado:

- Algoritmo de árbol de expansión de coste mínimo

3.2 Criptología (Alejandro, Marcelo y Arturo)

Para todos los métodos se debe mostrar una introducción explicativa, permitir (con indicaciones) que el aprendiz ingrese el texto claro y la clave de cifrado, y graficar, en la modalidad solicitada, tanto el cifrado como el descifrado.

---

[1] En cada caso:

- Vista paso a paso de la ejecución del algoritmo en el grafo, con colores que faciliten la comprensión de los pasos
- Posibilidad de deshacer los pasos
- Visualización en un panel del pseudocódigo, con resaltado de la sentencia que se ejecuta
- Conteo de número de sentencias ejecutadas
- Posibilidad de variar la velocidad de ejecución

3.2.1 Cifrados para Confidencialidad con Clave Privada (Simétricos)

De Texto (Criptología Clásica)

- Cifrado Afín (Z_{26} o Z_{256} –caracteres)
  Se debe mostrar los caracteres con inverso multiplicativo en el conjunto para la elección de claves.
- Cifrado de Permutación (Z_{26}^m –bloques de m caracteres, con m a elección del aprendiz)
  Se debe permitir establecer la permutación pseudoaleatoriamente o por parte del aprendiz.
- Vigenère (Z_{26}^m –bloques de m caracteres, con m determinado por la longitud de la clave determinada por el aprendiz)
- Cifrado de Hill
  Se debe permitir al aprendiz la elección del tamaño de bloque de cifrado entre 2 y 3, y orientarlo en la determinación de la matriz de claves (por prueba y error y justificación de invalidez o validez de cada intento).

De Flujo o En Cadena (*Stream Ciphers*)

- NLFSR(n) (*n bits non lineal feedback shift register*)
  Se debe permitir que el aprendiz determine (con indicaciones) a n y a la función no lineal de retroalimentación.
- RC4 (permutación de palabras de 8 bits)
  Se debe permitir que el aprendiz determine la clave (5 a 32 caracteres) y mostrar primero el procesamiento de la clave (KSA) y luego la generación del flujo (PRGA).

En Bloques (*Block Ciphers*)

- DES(64, 56) (*Data Encryption Standard*)
  Se debe permitir que el aprendiz determine la clave de 7 caracteres (56 bits) y mostrar el esquema de generación de subclaves, y el cifrado de bloques (de 64 bits) en base a la red Feistel.
- Variantes del DES (Doble DES(64, 112), Triple DES(64, 112))
- IDEA(64, 128)
- BLOWFISH(64, 32-448)
- TWOFISH(128, 256)
- CAST(64, 128)
- AES(128, 256) (*Advanced Encryption Standard*)

3.2.2 Cifrados para Confidencialidad con Clave Pública (Asimétricos)

- RSA (Rivest-Shamir-Adleman)
- ElGamal
- Intercambio de Claves (Diffie-Hellman)

Para estos métodos se debe introducir a la problemática de la obtención de números primos muy grandes (mayores que 10^{100}), p. ej., aludiendo al test de Miller-Rabin, pero las demostraciones se pueden hacer con números manejables, para ejemplificar la validez matemática de los esquemas; también se debe contemplar la determinación de números primitivos (con inverso multiplicativo) y graficar la exponenciación rápida.

3.2.3 Cifrados para Certificación de Autenticidad (Firma Digital)

- RSA como Esquema de Firma
- ElGamal como Esquema de Firma

Para estos métodos se debe comenzar con la especificación formal de esquema de firma.

3.2.4 Cifrados para Garantía de Integridad (Funciones Hash o de Digesto de Mensajes)

- MD5(512, 128)
- SHA-1(512, 160)
- SHA-2(512, 256)

3.3 Organización de Archivos (Arturo)

Se debe graficar la evolución de un archivo de registros de dato de longitud variable compuestos por:

- A, campo numérico de hasta 3 dígitos, con valor obligatorio en el registro y único en el archivo, y almacenado como cadena de caracteres con prefijo de longitud
- B, carácter en mayúscula, con valor opcional en el registro
- C, campo alfabético con nombres de personas (en minúsculas y sin acentos, pudiendo ser compuestos, pero con los nombres componentes separados por un solo espacio), también con valor obligatorio en el registro y único en el archivo, almacenado con prefijo de longitud

Una representación posible para un registro de dato sería: $^{14}{}_{4}349H_{8}$agustin

Donde 14 es un carácter representado por su valor decimal e implica la longitud en bytes del registro (incluyendo al mismo campo de control), y 4 y 8 son caracteres representados por sus valores decimales e implican longitudes de campos (también incluyéndolos en la longitud). Los campos de control se representan como supra o subíndices para denotar su función y formato de representación (ocupan un byte), o se puede adoptar otra convención equivalente, p. ej., usando colores de fondo distintivos. Para representar valores nulos del campo B, se puede usar el carácter Ø (*Latin Capital Letter O With Stroke*, con código ASCII decimal 216) o al espacio de no separación (*No-Break Space*, con código ASCII decimal 160), pero siempre respetando la misma convención.

Las opciones para la organización del archivo, a elección del aprendiz, pueden ser:

1. Secuencial en bloques de 64 bytes, con el primer bloque para control de espacio libre en el archivo, con la lista de bytes libres de cada bloque, almacenada como cadena secuencial de caracteres con prefijos de longitud[2]
   a. Con índices B o B* con nodos de 64 bytes para los campos A, B y C
   b. Con índices B+ o B# con nodos de 64 bytes para los campos A, B y C[3]
   c. Con índices Directos con Dispersión Extensible por bits Sufijos (Dispersión Modular en Z_t con t Tamaño de Tabla de Dispersión y Potencia de 2) con bloques de 64 bytes para los campos A, B y C
2. Árbol B o B* con nodos de 64 bytes, clave de organización A e índices con igual organización para los campos B y C
3. Árbol B+ o B# con nodos de 64 bytes, clave de organización A e índices con igual organización para los campos B y C, con prefijos de claves en nodos índice (internos) del árbol índice para el campo C

[2] P. ej., para representar un archivo cargado solamente con dos registros:

0|^{3}1a^{3}55^{3}33
1|$^{14}{}_{4}349H_{8}$agustin$^{17}{}_{3}86Ø_{12}$lucia belen

Donde en el bloque 0, 1a es la opción de organización del archivo, 55 (64-3-3-3) son los bytes libres del mismo bloque 0, y 33 (64-14-17) los del bloque 1.

[3] Los índices B+ o B# para el campo C deben organizarse con abreviatura frontal de claves (*front coding*) en los nodos de secuencia (nodos hoja), y prefijos mínimos de claves en nodos índice (nodos internos).

4. Directo con Dispersión Extensible por bits Sufijos (Dispersión Modular en Z_t con t Tamaño de Tabla de Dispersión y Potencia de 2) con bloques de 64 bytes, clave de organización A e índices con igual organización para los campos B y C

Para la opción 1, los índices por el campo A deben referir a los registros de datos por el número de bloque en que se encuentren (índices primarios) almacenado como cadena de caracteres, y **el orden de los valores del campo A debe ser numérico, pese a su almacenamiento en formato de cadena de caracteres**.

Para todas las opciones, los índices para los campos B y C deben referir a los registros de datos por el valor del campo A (índices secundarios) y los índices para el campo B no deben referir a registros de datos con valores nulos para dicho campo (índices selectivos), y como son índices cuyos registros refieren a uno o más registros de datos (índices de clasificación), sus registros deben estar compuestos por el valor del campo B, la cantidad de registros de datos que tienen ese valor, y la lista ordenada **numéricamente** de valores del campo A correspondientes a esa cantidad de registros de datos:

$$^{8}H^{2}1_{4}349$$

$$^{15}M^{2}3_{3}61_{4}124_{4}647$$

Donde los prefijos de longitud de cantidades de registros se representan como supraíndices para denotar que corresponden a campos de control, igual que el de la longitud del registro. Todo valor representado como supra o subíndice se interpreta como un valor ASCII que ocupa un byte.

Todo archivo índice debe reservar el nodo o bloque 0 para almacenar la lista de números de nodos o bloques libres, y eventualmente el número relativo del nodo raíz para los árboles.

Sea cual fuere la organización elegida por el aprendiz, la aplicación debe implementar realmente a todos los archivos pero sólo mostrar en la interfaz lo que solicite el mismo:

- Sólo el archivo de datos
- Sólo alguno de los índices, que en cualquier caso debe incluir al archivo de datos en un panel independiente

Para representar archivos con organización de árbol, se debe graficar el estado de cada nodo en una línea, y cada nodo debe contener como primeros campos de control el número de nivel (0 para las hojas), la cantidad de registros que contiene y la cantidad de bytes libres; el nodo 0 siempre debe representarse primero, con el número relativo de la raíz, la cantidad de bytes libres en el mismo, la cantidad de nodos libres del archivo, y, si hubiera nodos libres, la lista de números relativos correspondientes en el orden en que se liberaron. La disposición de los nodos debe distinguir su nivel con la longitud de sangrado, y todo nodo descendiente debe figurar a continuación de su padre (preorden):

$$0|^{2}3^{3}57^{2}0$$

$$3|^{2}1^{2}2\ldots$$

$$1|^{2}0^{2}3\ldots$$

$$4|^{2}0^{2}2\ldots$$

$$2|^{2}0^{2}1\ldots$$

Para representar archivos directos con dispersión extensible se debe graficar el estado de sus bloques, uno por línea, y en orden relativo. En el bloque 0 se debe almacenar su cantidad de bytes libres, la cantidad de bloque libres, y si hubiera bloques libres, la lista de números relativos correspondientes en el orden en que se liberaron; también se almacena la longitud de la tabla y los números relativos de bloques del mismo archivo donde se almacena (siempre ha de comenzar en el 1, y cada número relativo de bloque de la tabla

se almacena en un byte, como un valor ASCII, y se representa en la interfaz por su valor decimal, separado de los demás por un espacio).

Como el tamaño de las unidades de almacenamiento para todos los archivos es el mismo (64 bytes), cada archivo organizado puede empaquetarse en un mismo archivo físico con el nombre que haya provisto el aprendiz, o puede implementarse con un archivo físico para cada componente de la organización (el archivo de datos y los archivos índice, que deberán tener el mismo nombre que el de datos con un posfijo o extensión que los caracterice). En cualquier caso, la representación de las unidades de almacenamiento debe efectuarse en una línea. Y encabezada por el correspondiente número relativo de unidad en el archivo al que pertenezca.

En las aplicaciones para este tema, debe distinguirse la persistencia de los archivos organizados de la que el aprendiz pueda solicitar de una ejecución demostrativa.

Las opciones que debe proveer la aplicación para trabajar con archivos son:

1. Archivos
   1.1. Crear un archivo vacío solicitando el nombre y la organización y mostrando la carpeta actual donde almacenarlo, con posibilidad de navegación; si hubiera un archivo abierto, preguntar al aprendiz si quiere cerrarlo previamente para conservar eventuales cambios, o si quiere descartarlo (borrar los archivos físicos); si el nombre provisto coincidiera con un archivo existente en la carpeta destino, advertir la circunstancia y preguntar si se desea reemplazarlo o cambiar el nombre para almacenar uno nuevo
   1.2. Crear un archivo cargado, solicitando el nombre, la organización y una cantidad de registros para cargarlo pseudoaleatoriamente[4] (ídem 1.1)
   1.3. Abrir un archivo mostrando la carpeta actual, con posibilidad de navegación; si hubiera un archivo abierto, preguntar al aprendiz si quiere cerrarlo previamente para conservar eventuales cambios, o si quiere descartarlo (borrar los archivos físicos); si el archivo escogido fuera de respaldo secuencial (ver opción 4.4), solicitar la organización con que se desea recuperarlo
   1.4. Cerrar el archivo, persistiendo los últimos cambios; si el aprendiz cerrara la aplicación sin antes haber cerrado un archivo, preguntar si quiere cerrarlo previamente para conservar eventuales cambios, o si quiere descartarlo (borrar los archivos físicos)
2. Vista
   2.1. Archivo de Datos (por defecto: una vez creado o abierto un archivo, se debe graficar su estado en un panel de visualización)
   2.2. Índice, proveyendo la opción, mediante botones radiales o convención equivalente, de escoger B o C (al cambiar una opción de vista, se debe agregar o refrescar un panel de visualización para el índice escogido)
3. Modo
   3.1. Aprendizaje (por defecto), con opción de no persistir (también por defecto) o de persistir los estados del archivo tras cada operación
   3.2. Autoevaluación (ídem 3.1 para estados correctos)
4. Operaciones
   4.1. Buscar registros de datos (solicitando el valor del campo que corresponda a la vista activa)
   4.2. Agregar un registro de datos (solicitando los valores de cada campo: A, B y C, y normalizando valores de C –convirtiendo a minúsculas, sustituyendo caracteres con acento y verificando que si el nombre es compuesto, la separación sea con un solo espacio)

[4] Se debe disponer de un archivo con nombres de persona almacenados en registros de longitud fija para poder escogerlos pseudoaleatoriamente por su número relativo de registro.

4.3. Borrar registros de datos (solicitando el valor del campo que corresponda a la vista activa)
4.4. Respaldar el archivo (se almacena sólo el archivo de datos como archivo secuencial de registros de longitud variable, en un archivo nuevo con el mismo nombre que el de datos, con un posfijo o extensión que lo caracterice).

Para las opciones 4.1 y 4.3, si el campo involucrado es el C, se debe normalizar los valores que ingrese el aprendiz según se aclara en la opción 4.2.

Para la graficación paso a paso de todas la operaciones, se debe mostrar un panel de incidencias en el que se agregue, para cada paso, un registro compuesto por la indicación del archivo actual (Dat, IxA, IxB o IxC), el número relativo de la unidad actual (de bloque o de nodo), y una descripción de la incidencia; paralelamente, en el panel de visualización del archivo o índice correspondiente, se debe representar de algún modo dinámico (p., ej., con cambios de coloración), la implicancia de la incidencia.

Las descripciones de incidencias deben estar codificadas y parametrizadas para la generación de sus registros. Un ejemplo de una secuencia de incidencias para el alta de un registro de datos con clave de organización 123 en un archivo organizado como árbol B+ y asumiendo que cuando se abre se bufferiza su raíz, que corresponde, por caso, al nodo 3, sería:

Arch	**#u**	**Incidencia**
Dat	3	Búsqueda de clave mayor o igual a ***123***
Dat	3	Lectura de unidad ***1***
Dat	1	Búsqueda de clave mayor o igual a ***123***
Dat	1	Inserción de registro fallida por desborde
Dat	3	Inserción de unidad ***5*** a la derecha de ***1*** y balanceo de carga
Dat	3	Escritura de unidades ***1*** y ***5***
Dat	3	Inserción de clave ***348*** con sucesor ***5*** en unidad ***3***
Dat	3	Escritura de unidad ***3***

Los números enfatizados en cursiva corresponden a los parámetros de incidencia.

4 Plan de Trabajo

Los hitos para el desarrollo del trabajo, cuyos plazos de cumplimiento y requisitos se deben acordar con cada tutor son:

- Acuerdo de tema y técnicas asociadas según la cantidad de integrantes del grupo y coordinación de horarios de consulta con el tutor
- Informe de tecnologías a utilizar para el desarrollo del trabajo (plataforma, IDE, frameworks, repositorio, etc.)
- Diseño de interfaces de a aplicación (p., ej., con mockups)
- Modularización de la aplicación con interfaces de cada módulo, e informe de distribución de tareas entre integrantes del grupo
- Diagrama de clases de cada módulo
- Demostración de incrementos de funcionalidad implementados
- Demostración final y entrega de documentación, código fuente e instalador para plataformas Linux y Windows

5 Producto Final

El producto final debe comprender una carpeta de desarrollo, con la documentación del producto final:

- Indicación del tema y técnicas desarrolladas
- Tecnologías utilizadas para el desarrollo (plataforma, IDE, frameworks, repositorio, etc.)
- Diagrama de módulos y diagramas de clases por módulo, y distribución de tareas entre integrantes del grupo

También se debe entregar, en formato digital, el código fuente, e, independientemente, instaladores o archivos de implementación en versiones separadas para plataformas Linux y Windows, con documentación instructiva para la instalación y requisitos, p. ej., versión de JRE (en PDF).

6 Evaluación

La calificación del trabajo resultará del promedio de calificaciones del proceso de desarrollo y del producto, ponderado por oportunidad de la entrega final.

En la calificación del proceso de desarrollo contempla:

- Calidad del cumplimiento de hitos del plan de trabajo (corrección de documentación y cumplimiento de plazos acordados)
- Distribución de tareas y organización del grupo
- Interacción con el tutor (frecuencia, oportunidad y razonabilidad de consultas –eficiencia en el análisis y ajustes de funcionalidad)

En la calificación del producto se considera:

- Funcionalidad (corrección, completitud, eficiencia)
- Interfaz (coherencia, claridad, facilidad de comprensión y uso, ayuda contextual)
- Documentación de instalación
- Documentación técnica

El promedio de las calificaciones previas se pondera según la oportunidad de la entrega final:

- Hacia el final del período de cursado, 100%
- Durante el primer período de evaluaciones integradoras postcursado, 95%
- Durante el segundo período de evaluaciones integradoras postcursado, 85%
- Durante el tercer período de evaluaciones integradoras postcursado, 75%