UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Gºen Ingeniería en Informática

TFG del Grado en Ingeniería Informática

MEMORIA

Generador de preguntas de Programación Dinámica

ProgDinQuiz

Presentado por Asier Alonso Morante

en Universidad de Burgos – <Septiembre de 2016>

Tutores: Juan José Rodriguez Díez

Carlos López Nozal

Índice de contenido

Índice de ilustraciones

Resumen

Se desea desarrollar una herramienta de escritorio que ayude a generar preguntas de test aleatorias (tipo quiz, cloze...) sobre tipos de problema de programación dinámica.

La programación dinámica fue creada para la optimización de problemas complejos. Cuando hablamos de optimzar nos referimos a buscar alguna de las mejores soluciones de entre muchas alternativas posibles.

El diseño de un algoritmo de Programación Dinámica consta de estos pasos:

1. Planteamiento de la solución como un sucesión de decisiones y verificación de que ésta cumple el principio de optimalidad.
2. Definición recursiva de la solución.
3. Cálculo del valor de la solución óptima mediante una tabla en donde se almacenan soluciones a problemas parciales para reutilizar los cálculos.
4. Construcción de la solución óptima haciendo uso de la información contenida en la tabla almacenada.

Hoy en día el uso mas característico que puede tener la programación dinámica es a nivel económico y empresarial, el caso más claro es el de la optimización del uso de los recursos disponibles para la maximización del beneficio total.

Descriptores

Programación Dinámica, generador de preguntas...

Abstract

*{A **brief** presentation of the topic addressed in the project.}*[1]

The main purpose of this project is develop a desktop tool which could generate random test questions in differentes types (quiz, cloze...) about model types of dynamic programming.

The dynamic programming was created to optimize complex problem. When we speak about optimize we are talking about look for one of the best solutions between a lot of posible options.

The design of a Dynamic Programming algorithm consist of theese steps:

1. *Solution planification like one succession of decisions and verify if this plan keeps the Bellman's principle of optimality*
2. *Recursive definition of the solution.*
3. *Calculate the value of best solution by a table which keeps solutions of partial problems to reuse the calculations.*
4. *Build the best solutions using the information saved in the created table.*

Nowadays the main use which could have the dynammic programming is in economic and corporation environment, the main case is the optimization of the available resources to get the maximum profit.

Descriptors

Dynammic Programming, quiz generator...

1

.

I - *INTRODUCCIÓN*

En este proyecto se desarrollará una aplicación capaz de generar preguntas sobre la programación dinámica.

Dentro de la amplitud de problemas que pueden ser resueltos por el método de la programación dinámica se han seleccionado cuatro tipos como ejemplo de problemas de esta metodología, dichos problemas son:

Problema de la mochila, este problema simulará el llenado de una mochila con una capacidad especifica, y seleccionará de entre un número de objetos con peso y un beneficio cada uno, aquellos que sin pasarse del peso de la mochila, maximicen el valor de la mochila.

Problema de Subsecuencia Común Más Larga, este problema tratará dos cadenas de caracteres diferentes entre sí y obtendrá una subcadena entre ambas que contará con los caracteres comunes a las dos primeras, con la condición de que estos caracteres sigan el mismo orden en ambas.

Algoritmo de Floyd, con este problema partiremos de una situación de caminos entre varios puntos, algunos de ellos infinito, (sin caminos entre ellos). Cada camino entre los puntos tendrá un valor o un peso, con el algoritmo de Floyd obtendremos la relación entre todos los caminos, comunicar todos ellos, obteniendo el minimo peso entre todos los caminos.

Multiplicación de Matrices Encadenadas, este problema parte de la situación de un número de matrices encadenadas entre sí para poder ser multiplicadas. El resultado de la multiplicación de las matrices no variará, pero el orden en el que se multipliquen si que hará que varíe el número de operaciones que se realicen.

En esta memoria estarán indicados todos los pasos que se han seguido para desarrollar el producto.

Los objetivos, donde se indican las metas que se plantearon desde el principio de la creación de la aplicación, conceptos teóricos y herramientas utilizadas, así cómo los aspectos mas relevantes del proyecto.

Para finalizar se hablará de otros proyectos relacionados con la aplicación y de las conclusiones obtenidas de la realización de este proyecto.

Además de esta memoria existirán una serie de anexos donde se podrá encontrar diferentes manuales tanto para usuarios, para nuevos programadores, etc.. .

II - Objetivos del proyecto

El objetivo principal de este proyecto es realizar una aplicación de escritorio que ayudará a profesores y alumnos en el aprendizaje y enseñanza de la programación dinámica, dentro de la asignatura de Algoritmia.

1. Objetivos Técnicos.

Los objetivos del proyecto, vistos desde el punto de vista del desarrollo técnico de la aplicación, pueden desglosarse en los siguientes puntos:

Desarrollo de un programa capaz de generar preguntas acerca de programación dinámica con su correspondiente solución o soluciones correctas.

La aplicación será capaz de generar preguntas sobre los diferentes tipos de problemas de programación dinámica (problema de la mochila, subsecuencia común mas larga, matrices encadenadas y problema del viajero)

Realizar una aplicación completa con una interfaz sencilla e intuitiva.

Crear una aplicación robusta y sin errores.

Obtener una aplicación capaz de exportar código a diferentes entornos como son Moodle utilizando para ello tecnologías como JAXP para exportar a xml.

Crear código lo más independiente y reutilizable posible, bien comentado y fácil de asimilar con el objetivo de que pueda seguir siendo desarrollado en el futuro para añadir nuevas funcionalidades a la aplicación o para otras aplicaciones relacionadas con esta.

2. Objetivos Personales

A nivel personal, el objetivo con este proyecto, es asentar todos los conocimientos que se han ido adquiriendo durante la etapa académica y aplicarlos a un entorno más real y menos teórico, así como ganar madurez a la hora de desarrollar un proyecto.

Además, este proyecto me dará la oportunidad de conocer nuevas tecnologías y consolidar aquellas que ya conocía.

III - Conceptos teóricos

En este apartado, se explicarán aquellos aspectos teóricos que han sido utilizados dentro del proyecto.

En primer lugar, se explicará en qué consiste realmente la programación dinámica y después se introducirán los problemas tipos con los que trabaja la aplicación como ejemplo de uso.

1. Programación Dinámica

1.1. ¿Qué es?

La programación dinámica puede verse como una optimización de otros métodos algorítmicos como son el método de “Divide y Vencerás”, la “Vuelta Atrás”... . En estos algoritmos una solución a un problema puede implicar la repetición continua de subproblemas.

En concreto, la programación dinámica suele implicar la utilización de una tabla auxiliar donde almacenar las soluciones a los subproblemas ya calculados, de tal forma que cada subsolución solo se calcule una única vez, reduciendo así el coste en tiempo a cambio de coste en espacio (almacenamiento en una tabla).

Para poder aplicar un programación dinámica se debe comprobar que el problema cumple el principio de optimalidad de Bellman. Si esto sucede se deberá definir recursivamente la solución óptima del problema (en función de los valores de las soluciones para subproblemas de menor tamaño). [1]

1.2. Problema de la mochila

1.2.A. Definición

En algoritmia, el problema de la mochila, comunmente abreviado por KP (del inglés Knapsack problem) es un problema de optimización combinatoria, es decir, que busca la mejor solución entre un conjunto de posibles soluciones a un problema.

Modela una situación semejante a llenar una mochila, incapaz de soportar más de un peso determinado, con todo o parte de un conjunto de objetos, cada uno con un peso y valor específicos. Los objetos colocados en la mochila deben maximizar el valor total sin exceder el peso máximo.

Esta situación se presenta a menudo en los ámbitos económico e industrial, donde la mochila suele representar una restricción presupuestaria y donde la utilidad de los objetos seleccionados se equipara a un beneficio económico por adquirir o llevar a cabo ciertas acciones. [19]

1.3. Ejemplo de resolución del problema de la mochila

Supongamos una mochila con una capacidad de 11 y 5 elementos, con los siguientes pesos y valores cada uno:

Tendremos que:

- n = 5
- C = 11

Objeto	Valor	Peso
1	1	1
2	6	2
3	18	5
4	22	6
5	28	7

Ilustración 1: Tabla valores iniciales de la mochila

Consideramos los beneficios $(b_1, b_2, b_3, b_4, b_5) = (1, 6, 18, 22, 28)$
y los pesos $(p_1, p_2, p_3, p_4, p_5) = (1, 2, 5, 6, 7)$

Rellenaremos una matriz

Considerando que *j* es el elemento que estamos tratando y c la capacidad parcial de la mochila.

Si $\bar{g}_j(C)$es el beneficio (o ganancia total) de una solución óptima de mochila(*j,n,c*), entonces

$$\bar{g}_j(c) = \max\{\bar{g}_{j+1}(c),\ \bar{g}_{j+1}(c - p_j) + b_j\}$$

dependiendo de que el objeto j-ésimo entre o no en la solución (nótese que sólo puede entrar si $c\text{-}p_j \geq 0$).

Además

$$\bar{g}_{n+1}(c) = 0, \text{ para cualquier capacidad } c$$

Para evitar la repetición de cálculos, las soluciones de los subproblemas se deben almacenar en una tabla.

Si creamos una matriz M ($n \times C$) cuyo elemento (j,c) almacena $\bar{g}_j(c)$. Obtendremos la siguiente tabla.

	0	1	2	3	4	5	6	7	8	9	10	11
{ϕ}	0	0	0	0	0	0	0	0	0	0	0	0
{1}	0	1	1	1	1	1	1	1	1	1	1	1
{1,2}	0	1	6	7	7	7	7	7	7	7	7	7
{1,2,3}	0	1	6	7	7	18	19	24	25	25	25	25
{1,2,3,4}	0	1	6	7	7	18	22	24	28	29	29	40
{1,2,3,4,5}	0	1	6	7	7	18	22	28	29	34	35	40

Ilustración 2: Tabla resultado final de la mochila

De la matriz anterior, podemos deducir el valor máximo que puede llevar la mochila, que corresponde con el valor del último elemento de la matriz, en este caso, 40. [20]

1.4. Problema de subsecuencia común más larga (LCS)

1.4.A. Definición

Se trata de encontrar la subsecuencia más larga que es común en un conjunto de secuencias (aunque en la mayor parte solamente se toman dos secuencias). Es diferente del problema de substring común más largo; a diferencia de los substrings, las subsecuencias no necesitan tener posiciones consecutivas en la secuencia original aunque si que deben mantener el mismo orden, esto es, dadas dos cadenas, X e Y, encontrar la longitud de la cadena mas larga,Z, tal que los caracteres de Z aparezcan en el mismo orden dentro de las cadenas X e Y.

La técnica de programación dinámica con la que se intentará resolver este problema es a partir de las soluciones de subproblemas en los que se divide el problema original.

El problema de la subsecuencia común mas larga se aplica en la bioinfórmatica donde es frecuente la necesidad de comparar el ADN de dos o más organismos. Las secuencias de ADN son representadas como una cadena de letras, un posible criterio de similitud entre cadenas de ADN es encontrar las subsecuencia común mas larga con respecto a otras cadenas de ADN. [2]

1.4.B. Ejemplo de resolución de subsecuencia común más larga

Si tenemos dos cadenas,

X = “ABCBDAB” con longitud igual a 7

Y = “BDCABA” con longitud igual a 6

Se construye una matriz que llena con los valores de la subsecuencia común mas larga para cada pareja de valores.

Comparan cada carácter de una de las cadenas con todos los caracteres de la otra cadena, en el caso de que los caracteres coincidan, aumentara en uno el valor que marque la matriz para los dos caracteres anteriores de las dos matrices. En caso de no ser iguales, cogeremos el mayor de los valores de entre el valor de la posición anterior en la matriz, o el valor superior de la misma. Al escribir el valor, se indica con una flecha desde que posición se ha obtenido el valor que se escribirá en la posición en la que nos encontremos.

Quedará una matriz como la siguiente.

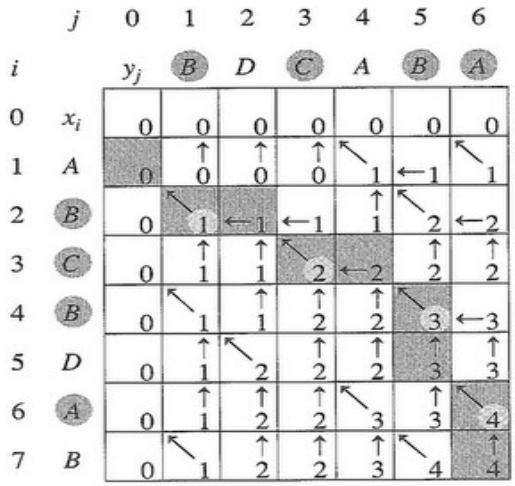

Ilustración 3: Matriz resultado Subsecuencia Común

Una vez terminado de rellenar la matriz, es sencillo encontrar la subsecuencia común más larga ya que solo deberemos situarnos en la última posición y retroceder siguiendo las flechas introducidas anteriormente. Cuando el desplazamiento sea en diagonal significará que el carácter donde se produzca se encontrará en la subsecuencia común final. En este caso el resultado será: **bcba.**

1.5. Algoritmo de Floyd.

1.5.A. Definición

El algoritmo de Floyd intenta resolver el problema de encontrar el camino mas corto entre todos los pares de nodos o vértices de un grafo. Esto es similar a construir una tabla con todas las distancias mínimas entre pares de ciudades de un mapa, indicando la ruta a seguir para ir de la primera ciudad a la segunda. Esto puede verse de la siguiente manera:

- Sea G= (V, A) un dígrafo en el cual cada arco tiene asociado un costo no negativo. El problema es hallar para cualquier par de vértices (*v, w*) el camino más corto de *v* a *w.*
- G= (V, A), V= {1,...,n} y $C_{i,j}$ es el costo del arco que va de i a j.
- El algoritmo calcula la serie de matrices
- $Ak_{i,j}$ significa el costo del camino más corto que va de i a j y que no pasa por algún vértice mayor que k.
- El objetivo es calcular $An_{i,j}$

Además de eso, se busca hallar también el camino más corto entre cada par de nodos, imprimiendo así en el programa no solo la matriz final sino también dichos caminos.

La utilización de este algoritmo es muy útil a la hora de realizar viajes a varios puntos o podría ser utilizado por empresas de transporte que tengan que ir a varios puntos, minimizando los gastos totales para ir de un punto a otro.

1.5.B. Ejemplo de resolución de algoritmo de floyd.

Sea un grafo G con un conjunto de vértices V, numerados de 1 a N. Sea además una función *caminosMinimos(i,j,k)* que devuelve el camino mínimo desde cada i a cada j usando únicamente los vértices de 1 hasta k + 1.

Hay dos candidatos para este camino: un camino mínimo, que utiliza únicamente los vértices del conjunto {1...k}; o bine existe un camino que va desde i hasta k+1, y de k+1 hasta j, que es mejor. Sabemos que el camino óptimo de i a j que únicamente utiliza los vértices de 1 hasta k está definido por *caminoMinimo (i,j,k)*, y está claro que si hubiera un camino mejor de i a k+1 a j, la longitud de este camino sería la concatenación del camino mínimo de i a k+1 (utilizando vértices de (1..k)) y el camino mínimo de k+1 a j (que también utiliza los vértices en (1...k).

Por lo tanto, podemos definir caminoMinimo(i,j,k) de forma recursiva:

caminoMinimo(i,j,k) = min (*caminoMinimo(i,j,k-1)*, *caminoMinimo(i,k,k-1)* + *caminoMinimo(k,j, k -1)*);

caminoMinimo(i,j,0) = pesoArista(i,j).

Esta fórmula es la base del algorimo de Floyd-Warshall. Funciona ejecutando primero *caminoMinimo(i,j,1)* para todos los pares (i,j), usándolos para después hallar *caminoMinimo(i,j,2)* para todos los pares (,i,j)... Este proceso continúa hasta que k=n, y habremos encontrado el camino más corto para todos los pares de vértices (i,j) usando algún vértice intermedio. [4]

1.6. Multiplicación de matrices encadenadas.

1.6.A. Definición

Supongamos que tenemos las matrices M1, M2, ..., M_n, que queremos multiplicar:

$$M = M1 \times M2 \times \ldots \times M_n$$

Puesto que el producto es asociativo, habrá muchas formas de realizar las multiplicaciones. Cada colocación de los paréntesis indica un orden en el que se realizan las operaciones.

Según el orden de las multiplicaciones, el número total de multiplicaciones escalares necesarias puede variar considerablemente.

Sea una matriz *A* de dimensión p x q y *B* de q x r, entonces el producto A x B requiere p x q x r multiplicaciones escalares (método clásico).

Decidamos el orden que decidamos el resultado siempre será el mismo. La diferencia estará en el número de multiplicaciones que implica elegir un orden u otro. [3]

1.6.B. Ejemplo de resolución de multiplicación matrices encadenadas.

Sean las matrices A, B, C y D, de dimensiones: A= 13x5, B= 5x89, C= 89x3 y D= 3x34. Podemos multiplicarlas de 5 formas:

((AB)C)D	Requiere	10.582 = 13·5·89 + 13·89·3 + 13·3·34
(AB)(CD)	"	54.201
(A(BC))D	"	2.856 = 5·89·3 + 13·5·3 + 13·3·34
A((BC)D)	"	4.055
A(B(CD))	"	26.418

Objetivo: obtener un orden de multiplicación que minimice el número de multiplicaciones escalares.

Solución sencilla: estimar el número de multiplicaciones necesarias para todas las ordenaciones posibles. Quedarse con la que tenga menor valor.

Solución utilizando programación dinámica

Definimos NMulti (i, j): el número mínimo de productos escalares necesarios para realizar la multiplicación entre la matriz i y la j (con $i \leqslant j$), es decir:

$$M_i * M_{i+1} * \ldots * M_j$$

Suponemos que las dimensiones se almacenan en un array *d[0...n]*, donde la matriz M_i será de dimensión *d[i-1]* X *d[i]*.

- Si i = j, entonces NMulti(i, j) = 0. No necesitamos realizar ninguna operación.
- Si i = j-1, entonces NMulti(i, j) = $d_{i-1} \cdot d_i \cdot d_{i+1}$. Sólo existe una forma de hacer el producto.

Forma de rellenar la tabla.

Para hallar cada subsolución aplicamos la siguiente función que minimiza el coste total en número de multiplicaciones.

$$M_{i,j} = \{ \min (M_{i,k} + M_{k+1,j} + d_{i-1} \cdot d_k \cdot d_j) \text{ para todo k siendo } i \leqslant k < j \} \quad [5]$$

	j= 1	2	3	4
i=1	0	10.000	1.200	2.200
2		0	1.000	3.000
3			0	5.000
4				0

REF:

Ilustración 4: Tabla final problema Multiplicación de Matrices

WIKILIBROSMATRICES

IV - Técnicas y herramientas

Esta capítulo hará referencia a las técnicas que se han seguido y que han sido útiles en la realización, se explicará brevemente en que consiste cada una de las técnicas o herramientas así como su funcionalidad dentro del proyecto y una dirección web donde encontrar mas información acerca de la herramienta o técnica en cuestión.

1. Herramientas

En esta sección hablaremos de las herramientas que se utilizaran para el desarrollo de la aplicación.

1.1. Java

Será el lenguaje con el que se realizará el desarrollo de este proyecto.

Java es un lenguaje de programación y una plataforma informática comercializada por primera vez en 1995 por Sun Microsystems.

El principal motivo por el que ha sido seleccionado este lenguaje es porque se trata de un lenguaje de propósito general, alto nivel y orientado a objetos.

Java se puede ejecutar sobre cualquier plataforma en cualquier momento, para poder ser ejecutado, simplemente es necesario disponer de una máquina virtual java (JVM)

Para la realización de esta aplicación se ha utilizado la versión 7 de Java, aunque existe una versión 8, se ha considerado utilizar esta, ya que la aplicación no ha requerido utilizar ninguna de las nuevas características que pueda ofrecer la nueva versión de Java. [5]

Página web de la herramienta: https://www.java.com/es/

1.2. Eclipse (Eclipse IDE for Java EE Developers)

Eclipse es una comunidad para indiviuos u organizaciones que quieren colaborar en el desarrollo de software de código abierto. Estos proyectos estarán enfocados en la creación de una plataforma de desarrollo compuesta de frameworks extensibles, herramientas y rutinas para el desarrollo, despliegue y manejo de software durante el ciclo de vida.

Eclipse es el IDE(entorno integrado de desarrollo) elegido para la creación y desarrollo de la aplicación de escritorio.

Eclipse puede ser el IDE más conocido para crear aplicaciones de escritorio de tipo Java, frente a Netbeans. El motivo por el que ha sido seleccionado el primero para el desarrollo de la aplicación es por tener un mayor conocimiento sobre él adquirido durante los años de la carrera.

La versión con la que se trabaja en este desarrollo, es la versión Mars Release (4.5.0) la penúltima versión en aparecer, actualmente la versión mas novedosa es la nueva versión Neón de eclipse. [6]

Página web de la herramienta: https://eclipse.org/org/

1.3. Github

Github es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones GIT.

Además del control de versiones, también ofrece la posibilidad de gestión de tareas.

Es la herramienta que se utilizará durante el desarrollo del proyecto, se escogió esta herramienta debido a que era la única que se conocía al iniciar el proyecto, ya que había sido utilizada en alguna de las asignaturas de la carrera.

Esta herramienta daba la posibilidad de tener un “punto común” con los tutores. En ella se crearían todas las tareas que se llevarían a cabo durante el desarrollo de la aplicación y se podrían crear los hitos que marcarían las reuniones que se mantendrían a lo largo del proyecto.

Github posee una API con eclipse, que permite subir directamente todo el código desde la propia herramienta al repositorio, para poder ver una mejor evolución del desarrollo de la aplicación. [8]

Página web de la herramienta: https://github.com/

1.4. Moodle

Moodle es una plataforma diseñada para los educadores, administradores y estudiantes con un sistema solido, seguro e integrado para crear entornos de aprendizaje personalizados.

Moodle ofrece un poderoso conjunto de herramientas de aprendizaje y entornos de aprendizaje colaborativos que fortalecen la enseñanza como el aprendizaje.

Moodle es una aplicación web y es accesible desde cualquier lugar del mundo, el contenido en la plataforma de Moodle es fácilmente alcanzable y consistente entre los diferentes navegadores y dispositivos web.

El uso de esta herramienta ha sido muy alto ya que el objetivo primero de la aplicación es el de crear cuestiones capaces de ser exportables a esta herramienta, creando preguntas que moodle pueda reconocer. [9]

Página web de la herramienta: https://docs.moodle.org/

1.5. WindowBuilder

WindowBuilder está compuesto de SWT Designer y Swing Designer y facilita la creación de aplicaciones Java gráficas sin dedica mucho tiempo a escribir código.

Utiliza un diseñador visual y herramientas de capas para crear simples formularios en ventanas complejas, mientras irá generando el código Java por uno mismo.

Permite añadir controles de forma sencilla utilizando una técnica de drag-and-drop, añadir manejadores de eventos a los controles, cambiar varias propiedades usando un editor de propiedades.

WindowBuilder se utiliza como plug-in de Eclipse. [10]

Página web de la aplicación: https://eclipse.org/windowbuilder/download.php

1.6. Gagawa

Gagawa es una herramienta OpenSource de generación de código HTML en Java y PHP.

Permite a los desarrolladores crear código HTML validado de forma dinámica. Gagawa permite de forma sencilla crear un documento HTML utilizando objetos para representar cada elemento, permitiendo añadirle texto, atributos o otras etiquetas.

Para utilizarlo solo es necesario descargar el fichero .jar y añadirlo al proyecto que se vaya a desarrollar.

El uso de esta herramienta en el proyecto ha sido para parsear desde java a Html las preguntas que deseaban ser exportadas a este último formato. [11]

Página de descarga de la librería: https://code.google.com/archive/p/gagawa/downloads

1.7. Java Dom Parser

Es una API de Java que permite el procesado de ficheros xml, entre otros, a partir del DOM de cada fichero.

Para utilizar la herramienta Java DOM Parser deberemos importar la librería de java, *javax.xml* En ella aparecerán todas las interfaces que implementarán las clases que se necesitan para el tratamiento de un fichero xml en Java.

Se utilizó está librería porque pareció la forma mas sencilla para mapear un fichero xml en java, puesto que solo se requería utilizarse en una clase, a diferencia del formato Html, no se decidió descargarse ninguna otra librería para parsear datos y se decidió trabajar con esta. [12]

Más información sobre la librería y como usarse:
https://www.tutorialspoint.com/java_xml/java_dom_parser.htm

1.8. iTextPdf

iTextPdf es una herramienta que permite crear, editar y leer ficheros de tipo PDF de forma dinámica desde diferentes lenguajes de programación.

Actualmente es una herramienta de pago, aunque se encuentra por la versión 7. En el proyecto se ha utilizado la versión 5.1 ya que ha sido mas fácil de obtener y para los requisitos de la aplicación ha sido más que suficiente.

En la aplicación se ha utilizado para la exportación de las preguntas en formato PDF. [13]

Página web de la herramienta: http://itextpdf.com/

Página web de descarga de la versión 5.1: https://sourceforge.net/projects/itext/

1.9. JavaHelp

JavaHelp es una expansión de Java que facilita la programación de las ventanas de ayuda en las aplicaciones java.

Con JavaHelp se pueden crear las ventanas típicas de ayuda de las aplicaciones informáticas, en las que sale en el lado izquierdo un panel con varias pestañas: índice de contenidos, búsqueda, temas favoritos, índice alfabético, etc. En el lado derecho sale el texto de la ayuda. [14]

2. Técnicas

2.1. Preguntas tiempo cloze

Son tipos de preguntas que permiten incrustar en texto zonas que deben ser completadas por los alumnos.

Existen varios formatos de preguntas de tipo de cloze. [9]

- Respuesta corta (SHORTANSWER o SA o MW), donde no son importantes minúsculas/MAYÚSCULAS,
- Respuesta corta (SHORTANSWER_C o SAC o MWC), donde deben coincidir minúsculas/MAYÚSCULAS,
- Respuesta numérica (NUMERICAL o NM),
- Opción múltiple (MULTICHOICE o MC o MCS), representada como un menú desplegable en-línea dentro del texto
- Opción múltiple (MULTICHOICE_V o MCV o MCVS), representada como una selección de columna vertical de botones para elegir, o
- Opción múltiple (MULTICHOICE_H o MCH o MCHS),representada como una hilera horizontal de botones para elegir,

2.2. Metodología Scrum

Scrum es un modelo de desarrollo ágil basado en la adaptación continua a las circunstancias de la evolución del proyecto. Emplea la estructura de desarrollo ágil incremental basada en iteraciones y revisiones.

Se comienza con la visión general del producto, especificando y dando detalle a las funcionalidades o partes que tienen mayor prioridad de desarrollo y que pueden llevarse a cabo en un periodo de tiempo breve (normalmente de 30 días).

Cada uno de estos periodos de desarrollo es una iteración que finaliza con la producción de un incremento operativo del producto.

Revisión de las Iteraciones: Al finalizar cada iteración (normalmente 30 días) se lleva a cabo una revisión con todas las personas implicadas en el proyecto.

Desarrollo incremental: El desarrollo incremental implica que al final de cada iteración se dispone de una parte del producto operativa que se puede inspeccionar y evaluar.

Desarrollo evolutivo: El desarrollo Scrum va generando el diseño y la arquitectura final de forma evolutiva durante todo el proyecto. No los considera como productos que deban realizarse en la primera “fase” del proyecto.

La principal idea del proyecto era seguir esta metodología de desarrollo ya que se consideraba como la mejor a la hora de la obtención de un producto, en el capitulo de Dificultades Encontradas se explican los problemas que han existido a la hora de usar esta metodología. [15]

2.3. Patrones de diseño:

Puesto que uno de los objetivos era realizar una aplicación reutilizable por otros desarrolladores, una de las principales técnicas que se han pretendido utilizar son los patrones de diseño, ya que son estandares de programación y que pretenden dar soluciones a problemas semejantes, por lo tanto un programador que no conozca los patrones de diseño,simplemente puede buscar información sobre ellos y ver su funcionamiento y aplicarlo sobre el código.

Los patrones de diseño también consiguen facilitar los códigos, algo importante sobre todo para aplicaciones sobre las que se desea obtener una evolución en un futuro, añadiendolas nuevas funcionalidades.

En este proyecto han sido dos los patrones de diseño utilizados:

Patrón de diseño Builder: es usado para permitir la creación de una variedad de objetos complejos desde un objeto fuente (en nuestro proyecto la clase Pregunta).

El objeto fuente se compone de distintas partes que contribuyen individualmente a la creación de los objetos complejos (en nuestro proyecto, las clases PreguntaMochila, PreguntaSubsecuencia, etc...). [18]

Para poder ver mejor como se ha utilizado el patrón en el proyecto, visitar el capitulo de Aspectos Relevantes. 3.1

Patrón de diseño Estrategia, permite tener un grupo de clases de las cuales, el cliente en tiempo de ejecución, seleccionará cual de ellas es la que más le conviene, e intercambiarlo dinámicamente según sus necesidades. [17]

En el caso de nuestro proyecto se utiliza a la hora de exportar las preguntas generadas por la aplicación, ya que el cliente será quién decida a que formato desea exportar.

Para ver como funciona, acudir al capítulo Aspectos Relevantes dónde se explica como se utiliza dicho patrón. 3.2

V - Aspectos relevantes del desarrollo del proyecto

En este apartado se introducen los aspectos mas relevantes del proyecto.

1. Introducción

El sistema fue desarrollado bajo JAVA (JDK 1.7.0_17), y se usó Swing para la interfaz. Para el desarrollo del código Swing se utilizó la herramienta integrada de Eclipse, Window Builder.

De las 6 fases (o etapas) en las que se decidió desarrollar el proyecto ProgDin, solo se buscó implementar las tres primeras. Esto es debido a razones de tiempo, porque además se estaban realizando los planes para probar los métodos, clases y subsistemas codificados para dichas fases.

Actualmente ProgDinQuiz, elabora una recomendación curricular de acuerdo con las especificaciones dadas, pero en futuras versiones debe optimizarse la ejecución de estas tareas y mejorar la robustez del sistema, pues, actualmente el software no contempla casos imprevistos.

En las siguientes etapas de este manual, se explica el funcionamiento de los procesos críticos, y se enumeran sugerencias para optimizar el sistemas en futuras versiones.

2. Ciclo de Vida

Inicio del proyecto: Durante un periodo de tiempo previo a comenzar a programar, se dedicó un tiempo a estudiar cual podría ser la mejor forma de la realización del producto, forma que tendrán las ventanas, métodos que tendrán las clases que generarán todos los problemas, relación de interfaces, funcionalidad total que ofrecerá el programa.

Organización y Preparación: Mediante el intercambio de mensajes, se acordó que las reuniones se harían mediante la aplicación Hangouts de Google. En las primeras reuniones se trabajó sobre un ejemplo de ventana que serviría como modelo del resto de problemas.

Ejecución del trabajo: Después de corregir los errores de la primera versión de la ventana sobre la que estábamos trabajando y acordar todas las funcionalidades del producto. A partir de esa ventana modelo se procederá a desarrollar el resto problemas y de preguntas.

Cierre del proyecto: Cuando ya se ha realizado el desarrollo de la primera versión del producto completo y finalizado, se procederá a la documentación del mismo y a la realización de testes funcionales corrigiendo errores surgidos y añadiendo funcionalidad

2.1. La aplicación:

ProgDinQuiz es una aplicación orientada a objetos constituida por 30 clases desarrolladas en J.D.K 1.7.0_17 desarrollada utilizando la herramienta Eclipse, que puede ejecutarse en ambientes Windows, Linux y Solaris.

Básicamente, esta aplicación es capaz de generar cuatro tipos diferentes de problemas típicos de la programación dinámica, recibiendo solamente unos parámetros que será diferente por cada problema.

La aplicación generará los problemas con valores aleatorios, obtenidos de una semilla única que tendrá cada problema. La semilla se creará a través de los parámetros de cada problema y del tiempo del sistema en el momento de generar cada problema.

Cada semilla devolverá siempre los mismos valores aleatorios, lo que nos permitirá la recuperación de los problemas ya generados, introduciendo la semilla correspondiente a cada uno de ellos.

Con los problemas ya generados la aplicación será capaz de exportar dichos problemas a tres formatos diferentes.

- **HTML**, formato para poder ser visualizado en un navegador web
- **XML**, es el formato mas interesante ya que nos permitirá exportar las preguntas a Moodle.
- **PDF,** si los problemas quieren ser exportados en formato papel, como si fuesen problemas de un examen.

2.2. La interfaz

Las siguientes secciones pretenden dar a conocer el funcionamiento "menos evidente" de la interfaz del sistema ProgDinQuiz, así como de los errores, ausencias y futuras modificaciones y/o mejorías.

En principio, la primera mejora que se requiere viene del campo estético. Aunque funcional, la interfaz está lejos de ser visualmente satisfactoria.

Todas las clases pertenecientes a la interfaz estén identificadas por el sufijo "*Frame*"

El listado completo de las mismas es el siguiente:

InicioFrame: esta clase será el menú principal de la herramienta.

KnapsackFrame: clase donde se introducen los parámetros deseados para la generación del problema de la mochila

LCSFrame: esta clase recibirá los parámetros para generar los problemas de subsecuencia común mas larga, también tendrá un JtextPane que permitirá tener una vista preeliminar de cada problema generado.

TSPFrame: esta clase recibirá los parámetros de generación de un problema del viajante, TSP, y mostrará en un JtextPane una vista preeliminar de los problemas generados.

MultiplicaMatricesFrame: esta clase creará los problemas de multiplicación encadenada de matrices a partir de los parámetros recibidos y mostrará también una vista preeliminar de todos ellos.

ExportarFrame: En esta ventana existen dos opciones, la opción de Exportar problemas, donde se permite exportar todos los problemas que se han generado y que están almacenados en la lista *ProblemasGenerados* al formato que se seleccione.

La segunda opción que da la pantalla es la de *Recuperar Problema* introduciendo una semilla. La herramienta generará un problema con esa semilla y mostrará el problema con esa semilla en un JtextPane qué mostrará el problema recuperado

A continuación se habla acerca de algunos de los aspectos más relevantes de algunas de las clases. Otras fueron omitidas dado que poco se puede aportar a lo ya escrito en su documentación respectiva.

3. Arquitectura

Definir la arquitectura fue una de las fases más complejas que se encontró durante el desarrollo, ya que desde el primer momento se trató de realizar el proyecto pensando en una futura extensión.

La funcionalidad de generar los problemas fue la más llevadera, ya que solo exigía crear la lógica de problemas en Java. Una vez obtenida dicha lógica, el siguiente paso fue el de generar un Frame por cada problema, donde el usuario pudiese introducir los parámetros deseados. Los problemas generados se añaden a una lista de Problemas con la que trabajará la aplicación y que se eliminará al cerrar la aplicación.

Uno de los objetivos que se tuvo en cuenta para la aplicación fue la posibilidad de poder extender la aplicación en un futuro, añadiendo nuevos tipos de problemas de programación dinámico, nuevas preguntas, o también, podía llegar a evolucionar la aplicación añadiendo nuevos formatos a los que exportar los problemas generados por la aplicación.

Para lograr crear una aplicación con un código más sencillo a la hora del mantenimiento o a la hora de la evolución de la misma, se han utilizado diferentes patrones de diseño.

Otro aspecto relevante de la aplicación es la capacidad de recuperar problemas ya generados por parte de la aplicación, para ello se ha utilizado una semilla para cada problema, siendo esta única por cada uno de ellos.

Mediante dicha semilla, se obtendrán los diferentes valores para los problemas de programación dinámica.

A continuación se detalla cómo se ha realizado las funcionalidades más importantes del proyecto.

3.1. Funcionalidad de Exportar

Para realizar la actividad de exportar, es necesario que la lista con los problemas generados tenga al menos un problema creado que pueda ser exportado.

Para esta función se ha pensado que la mejor forma para desarrollarla es usando el Patrón de Diseño, *Estrategia*, de tal manera que existe una interfaz (Exportar) de la que van a depender todas las clases que definan los formatos a los que se puedan exportar los problemas generados (*ExportarHTML, ExportarXML,...*).

De esta forma se consigue que haya independencia entre todos los formatos, siendo muy sencillo la inclusión de nuevos formatos a los que se puedan exportar los ficheros, sin afectar en gran medida al código de la aplicación y sin tener que llegar a modificar un excesivo número de clases.

Entre las clases que exportan a las extensiones existentes y la interfaz que obtendrá los métodos que tendrán las clases que la implementan, hay una clase, ExportarEstrategia, que recibirá un parámetro con el tipo al que se quieren exportar los problemas. Ese parámetro determinará a que clase llamará el método ExportarEstrategia.

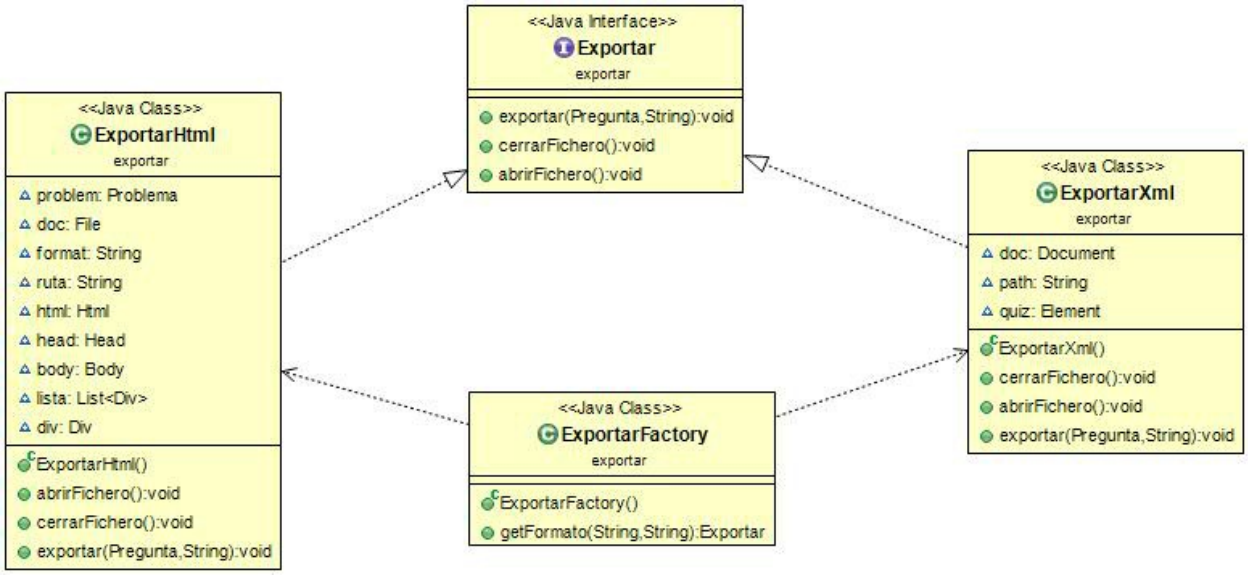

Ilustración 5: Diagrama Clases funcionalidad Exportar

3.2. Funcionalidad generar preguntas

La funcionalidad de la generación de preguntas es la actividad clave de la aplicación. Para la creación de esta funcionalidad se utiliza el patrón de diseño, *Builder,* con él, se define una clase, Pregunta, de la que extienden todas las clases que definirán los diferentes tipos de preguntas que habrá.

En el caso de querer añadir una nueva pregunta solo será necesario introducir una clase que extienda de esa clase Pregunta.

Existirá una clase abstracta que actuará como directora, PreguntaDirector, su función será la de crear o devolver preguntas, la clase PreguntaDirector tendrá un constructor, PreguntaBuilder, que será la clase que determine a qué clase llamar, según el tipo de pregunta recibida. Las clases a las que llamará son los constructores concretos, en el caso de esta aplicación, uno por cada tipo de problema.

El número de incógnitas que tendrá la pregunta, se definirá a la hora de generar un problema. Cada problema tendrá un slider donde se indicará el porcentaje de incógnitas que tendrá cada problema.

A la hora de imprimir las respuestas de cada pregunta se generará un número aleatorio entre 0 y 100, si este número aleatorio se encuentra por encima del porcentaje del problema, aparecerá como incógnita al exportar la pregunta.

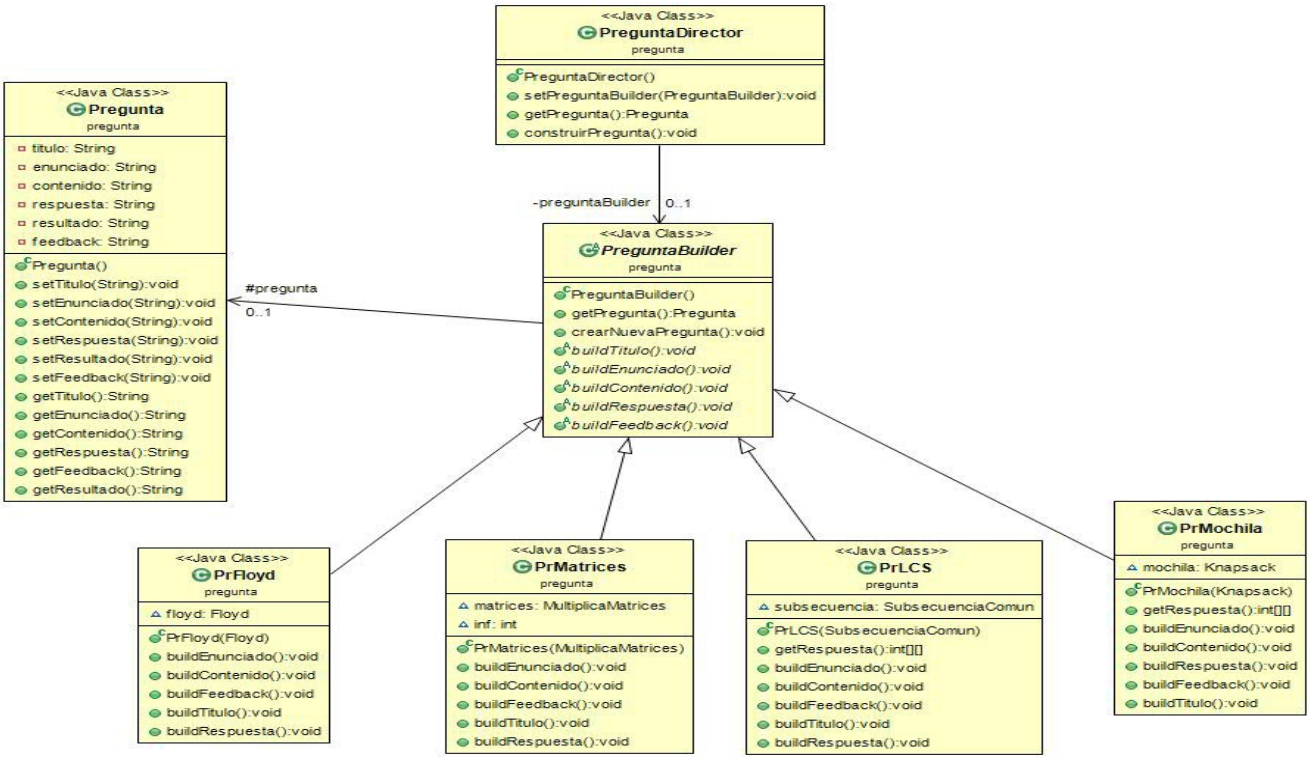

Ilustración 6: Diagrama de clases funcionalidad Generar Pregunta

3.3. Funcionalidad recuperar

Otra de las funcionalidades que admite la aplicación es la de recuperar problemas. En la pantalla de exportar se puede realizar la recuperación de un problema.

Es quizás una de las partes mas interesantes de la aplicación ya que resulta de gran utilidad poder obtener un problema deseado.

Para obtener dichos problemas, surgió la necesidad de crear una semilla, única para cada problema, de la que se obtendrían los mismos números aleatorios siempre. Por lo tanto, con dicha semilla, siempre se obtiene la misma serie de números aleatorios, la causa que permite recuperar problemas con la semilla.

La formación de la semilla se realiza uniendo diferentes datos propios de cada problema. Los dos primeros dígitos de la semilla corresponderán al tipo de problema al que pertenezca, todas las semillas del mismo tipo de problemas tendrán los mismos dos dígitos de comienzo.

Los siguientes caracteres de la semilla corresponderán con los valores introducidos por el usuario para la generación de problemas (capacidad de la mochila, longitud de secuencias, etc.).

Para terminar las semillas y evitar que existan dos iguales, se rellenan el resto de caracteres de la semilla con la fecha del momento en el que se genera el problema en milisegundos.

El motivo por el que el tamaño de las semillas es tan largo es precisamente para darle independencia a todas las semillas y no tener dos iguales.

Una vez recuperado un problema, éste, puede ser añadido a la lista de Problemas Generados y puede ser exportado a cualquier formato de nuevo.

4. Posibles extensiones o mejoras

1. Crear una pantalla de administración de los problemas, en la que aparezca una lista o una tabla con todos los problemas generados y poder visualizar todos los problemas que se han creado o se han recuperado, poder seleccionar los que se quieren exportar o poder ver las características con las que se ha creado cada problema.

2. Dados el aumento de uso de servicios de almacenamiento remoto, se sugiere como mejora la posibilidad de subir los ficheros directamente a un servicio de nuble como puede ser Google Drive o Icloud.

3. Crear una pantalla de ajustes de la aplicación donde poder determinar los valores máximos de los elementos o poder ajustar diferentes características según las preferencias del usuario que trabaje con la aplicación.

4. Poder exportar los ficheros a un formato JSON, para poder intercambiar toda la información generada por la aplicación con servicios o aplicaciones web.

5. En una versión muy mejorada, se podría crear una herramienta web, basada en la aplicación, integrada dentro del sistema Moodle que permita generar el problema dentro de la propia aplicación de Moodle sin necesidad de generar un fichero xml intermedio.

VI - TRABAJOS RELACIONADOS

En este apartado describiremos brevemente los trabajos de otros compañeros que por su similitud han estado relacionados con este y han servido de modelo para muchas de las funciones de la aplicación.

PLQuiz - Este proyecto presenta una aplicación capaz de generar preguntas aleatorias sobre problemas de algoritmos de análisis léxico que pueden ser exportadas a Moodle.

Generación de preguntas para la docencia on-line de Estructuras de Datos - Este proyecto trata sobre la creación de una aplicación capaz de generar de forma automática preguntas relacionadas con la asignatura de Estructuras de Datos.

Quiz-Grafos – En este proyecto se presenta una aplicación capaz de generar problemas sobre la creación y realización de grafos.

VII - Conclusiones y líneas de trabajo futuras

Estos mas de 5 meses que se han dedicado al desarrollo de la herramienta han servido para madurar conceptos de java que se habían aprendido durante la duración de la carrera.

Se ha completado un aprendizaje en la introducción y uso de patrones de diseño dentro de un código para permitir que el desarrollo sea mas sencillo y sobretodo mas claro y reutilizable a la hora de poder ser utilizado por otros desarrolladores en caso de querer añadir funcionalidades a la herramienta.

También este desarrollo ha sido realmente útil para comprender la dificultad que entraña la planificación y diseño de una aplicación, así como la importancia de conocer y utilizar herramientas de control de versiones, fundamentales a la hora de realizar un proyecto donde intervengan más de una persona, tanto como desarrolladores o simplemente controlando y verificando la evolución del código.

1. Dificultades Encontradas

La principal dificultad encontrada ha sido la inexperiencia en la gestión de proyectos y el desconocimiento de herramientas de control de versiones, lo que ha llevado a realizar una mala planificación que ha ido lastrando todo el desarrollo del proyecto.

La inclusión en el mundo laboral también ha restado tiempo al desarrollo del producto y no ha permitido añadir tantas funcionalidades como en un principio se deseaban.

2. Lineas de Trabajo Futuras

No ha sido posible actualizar el panel de visualización de forma inmediata, una mejora principal sería la actualización del panel de visión al realizar un cambio sobre el fichero sobre el que se carga.

Dados el aumento de uso de servicios de almacenamiento remoto, se sugiere como mejora la posibilidad de subir los ficheros directamente a un servicio de nuble como puede ser Google Drive o Icloud.

En una versión muy mejorada, se podría crear una herramienta web integrada dentro del sistema Moodle que permita con unos parámetros generar el problema dentro de la propia aplicación de Moodle sin necesidad de generar un fichero xml intermedio y tener que importarlo.

Adaptar la aplicación para poder ser usada por dispositivos móviles o tablets.

Posibilidad de una opción de realización de exámenes creando aleatoriamente preguntas de todos los tipos de problemas.

VIII - REFERENCIAS

Bibliografía

1: , (2016, 01). Algoritmia/Programación dinámica - Wikilibros. Es.wikibooks.org. Obtenido 01, 2016, de https://es.wikibooks.org/wiki/Algoritmia/Programaci%C3%B3n_din%C3%A1mica, 2016,
19: , (2016, 01). Problema de la mochila. Es.wikipedia.org. Obtenido 01, 2016, de http://es.wikipedia.org/wiki/Problema_de_la_mochila, ,
20: , (2016, 01). webdiis.unizar.es. Webdiis.unizar.es. Obtenido 01, 2016, de http://webdiis.unizar.es/asignaturas/EDA/ea/slides/4-Programacion%20dinamica.pdf, ,
2: Daniel Cam, (2016, 01). Subsecuencia común mas larga con programación dinámica - Código Informático. Xcodigoinformatico.blogspot.com.es. Obtenido 01, 2016, de http://xcodigoinformatico.blogspot.com.es/2012/09/subsecuencia-comun-mas-larga-con.html, ,
4: , (2016, 01). Algoritmo de Floyd-Warshall. Es.wikipedia.org. Obtenido 01, 2016, de https://es.wikipedia.org/w/index.php?title=Algoritmo_de_Floyd-Warshall&oldid=92511423, ,
3: , (2016, 01). Optimización del Producto de Matrices - Wikilibros. Es.wikibooks.org. Obtenido 01, 2016, de https://es.wikibooks.org/wiki/Optimizaci%C3%B3n_del_Producto_de_Matrices, ,
5: , (2016, 01). Programación Dinámica. Dis.um.es. Obtenido 01, 2016, de http://dis.um.es/~domingo/apuntes/Algoritmica/0203/prodin.ppt, ,
6: , (2016, 01). java.com: Java y Tú. Java.com. Obtenido 01, 2016, de https://www.java.com/es/, ,
8: , (2016, 01). Introducción — Conociendo GitHub 0.1 documentation. Conociendogithub.readthedocs.io. Obtenido 01, 2016, de http://conociendogithub.readthedocs.io/en/latest/data/introduccion/, ,
9: , https://docs.moodle.org/29/en/About_Moodle, ,
10: , (2016, 01). Obtenido 01, 2016, de https://eclipse.org/windowbuilder/, ,
11: , (2016, 01). Google Code Archive - Long-term storage for Google Code Project Hosting.. Code.google.com. Obtenido 01, 2016, de https://code.google.com/archive/p/gagawa/downloads, ,
12: , (2016, 01). Java DOM Parser - Overview. www.tutorialspoint.com. Obtenido 01, 2016, de https://www.tutorialspoint.com/java_xml/java_dom_parser.htm, ,
13: , (2016, 01). iText. Itextpdf.com. Obtenido 01, 2016, de http://itextpdf.com/, ,
14: , (2016, 01). JavaHelp. Es.wikipedia.org. Obtenido 01, 2016, de https://es.wikipedia.org/w/index.php?title=JavaHelp&oldid=64558539, ,
15: , (2016, 01). Navegapolis | Home. Navegapolis.net. Obtenido 01, 2016, de http://www.navegapolis.net/files/s/NST010_01.pdf, ,
16: , (2016, 01). Patrón de diseño. Es.wikipedia.org. Obtenido 01, 2016, de https://es.wikipedia.org/wiki/Patr%C3%B3n_de_dise%C3%B1o, ,
18: , (2016, 01). Builder (patrón de diseño). Es.wikipedia.org. Obtenido 01, 2016, de https://es.wikipedia.org/wiki/Builder_(patr%C3%B3n_de_dise%C3%B1o), ,
17: , (2016, 01). Strategy (patrón de diseño). Es.wikipedia.org. Obtenido 01, 2016, de https://es.wikipedia.org/wiki/Strategy_(patr%C3%B3n_de_dise%C3%B1o), ,

Impreso en Burgos el viernes, 16 de septiembre de 2016