Universidad Nacional del Centro de la

Provincia de Buenos Aires

**FACULTAD DE CIENCIAS EXACTAS**

Ingeniería de Sistemas



**Trabajo Práctico Especial**

**Análisis y Diseño de Algoritmos II**

**GRUPO 11**

Bedini Crocci Pia pbedini@alumnos.exa.unicen.edu.ar

Burckhardt David burck432@gmail.com

22/10/2021

**Índice**

[**Introducción**](#_nutaooww4rcs) **3**

[**Desarrollo**](#_mlj5ajczwtks) **4**

[**MODELADO DEL PROBLEMA**](#_6u8b50wmz6j9) **4**

[ÁRBOL DE EXPLORACIÓN](#_vevwl5mo616f) 4

[Tipos de estados](#_acaqj36gun8r) 4

[Criterio de ramificación y profundidad del árbol](#_fyodst51d8a5) 4

[DESCRIPCIÓN DEL ÁRBOL](#_dtywpl9iqvi4) 4

[INCORPORACIÓN DE PODAS](#_gxxl6zsdwhv3) 5

[**IMPLEMENTACIÓN DEL ALGORITMO**](#_fbw86jofot9t) **5**

[FUNCIONAMIENTO](#_wwcacn5i74fu) 5

[ESTRUCTURAS DE DATOS UTILIZADAS](#_jc45zimwe7zm) 5

[**RESULTADOS CON Y SIN PODA**](#_ql4tr8vzd5pi) **6**

[**Conclusión**](#_e6lis0r6vlgp) **9**

# **Introducción**

El presente trabajo intenta tratar los temas vistos hasta la fecha en la cátedra Análisis y Diseño de Algoritmos II.

El objetivo es trasladar los conocimientos teóricos a la práctica, aplicándolo en el lenguaje propuesto por la cátedra, C++, utilizando el IDE Code Blocks. A su vez, este documento contribuirá en la fijación de los conceptos importantes tales como: la resolución de problemas “difíciles” y los algoritmos de búsqueda exhaustiva, particularmente backtracking teniendo en cuenta todo lo que ésto implica, como lo es el problema a resolver, su solución, las restricciones, las soluciones factibles y el espacio de soluciones

Para contextualizar, es preciso aclarar que el servicio presente en esta parte del trabajo requiere realizar un sistema con un criterio de selección de libros para una biblioteca. Dicha biblioteca, cuenta con una cantidad de alumnos dada que asisten a ella y con un puntaje de aprobación estipulado para una asignatura. Además, la misma posee distintos tipos de libros con sus respectivos datos, entre ellos, sus ejemplares limitados.

El problema principal radica en maximizar la cantidad de alumnos aprobados en un único período (asegurando **siempre** la mejor asignación). Para esto, se empleará un modelo simplificado de una biblioteca escolar.

Un dato no menos importante es que la biblioteca abre sus puertas una vez cada tres meses y la información será provista al sistema días antes de la apertura, por lo que el tiempo de demora en la asignación no es relevante.

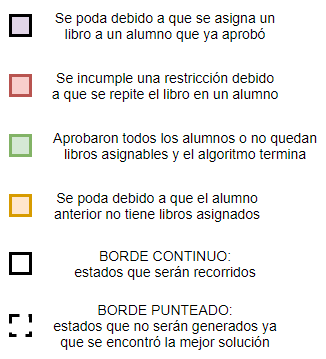
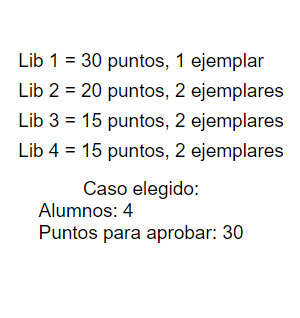
Es en ese momento en el que los alumnos retiran los libros y no los devuelven hasta la próxima apertura.

A continuación, se argumentarán las decisiones tomadas en relación al código programado analizando la generación de estados, las soluciones obtenidas y el tiempo de ejecución respecto de la implementación con y sin poda.

# **Desarrollo**

## **MODELADO DEL PROBLEMA**

### **ÁRBOL DE EXPLORACIÓN**



#### **Tipos de estados**

**Estado inicial:** Aquel estado del que parten todos los estados, en nuestro caso, es referido en el momento en que los alumnos todavía no tienen libros asignados

**Estado intermedio:** Estados que pueden ser expandidos, se pueden visualizar cuando al menos un alumno tiene asignado al menos un libro pero todavía quedan alumnos sin aprobar o libros sin asignar

**Estado final:** son estados que representan soluciones factibles, en este caso nos encontramos ante uno de estos cuando aprobaron todos los alumnos o el stock de libros se encuentra vacío.

#### **Criterio de ramificación y profundidad del árbol**

El árbol se ramifica tantas veces como alumnos se tengan y se avanza en profundidad a medida que se asigna un nuevo libro. En este caso, al elegir 4 alumnos, cada estado se va a ramificar en, como máximo, 4 estados nuevos posibles en los cuales el nuevo libro se asignará a un alumno distinto. Esto se repite a lo sumo tantas veces como libros haya en el stock inicial

### **DESCRIPCIÓN DEL ÁRBOL**

A continuación se procederá a explicar qué es y cómo está conformado un estado del árbol. Cada estado es la asignación de un nuevo libro a un alumno en particular. En cuanto a los datos, se puede destacar que cada estado contiene tantos conjuntos de libros como alumnos haya. Los conjuntos están ordenados según el alumno (el primer conjunto corresponde al primer alumno). En nuestro caso, un estado final representa una solución factible, por eso, no hay diferencias entre un estado final y un estado solución. Es importante aclarar que no es necesario que aprueben todos los alumnos para que este estado sea una solución ya que tal vez el stock de libros se haya vaciado.

Por otro lado, el problema nos aporta restricciones que permiten, por una parte, establecer un criterio de ramificación más acotado, y por la otra, determinar qué estados pueden seguir siendo explorados y cuáles no. Debido a que el alumno acumula puntos leyendo libros, no tiene sentido que lea dos veces el mismo libro, y por esta razón, no es posible asignar más de un ejemplar de un mismo libro a un solo alumno. Además se debe tener en cuenta que la cantidad de ejemplares por libro es limitada y por lo tanto, no es viable asignar infinitamente un libro por más que tenga el puntaje necesario para aprobar.

### **INCORPORACIÓN DE PODAS**

Al estar trabajando con problemas que requieren una búsqueda exhaustiva y conllevan la exploración de muchos estados posibles, es lógico pensar en alguna manera de recortar esta búsqueda. La forma correcta de realizar esto, es estableciendo un criterio que nos permita identificar estados que no podrán ser soluciones. Debido al contexto del problema, se observó que si un alumno alcanzó el puntaje para aprobar no es necesario continuar asignándole más libros, evitando así, tener entre muchos estados, un alumno con todos los libros asignables posibles. Además es viable realizar una poda ya que si todavía quedan libros disponibles en la biblioteca pero los alumnos ya han aprobado, no es necesario seguir ramificando, lo que lleva a toparse con un estado final. Por último, una poda un poco más rebuscada pero decisiva, surge de saber que todos los alumnos son iguales entre sí, y que el hecho de que los alumnos no tengan ningún puntaje hasta el momento hace que sea indistinto colocar un libro en un alumno o en otro. Sabiendo esto, si se quiere asignar un libro a un alumno que no tiene libros, se debe verificar que el alumno anterior no tenga puntaje 0 (que tenga al menos un libro).

## **IMPLEMENTACIÓN DEL ALGORITMO**

### **FUNCIONAMIENTO**

Para comenzar explicando el funcionamiento del sistema primero es necesario comentar acerca de la técnica que se va a utilizar. Para la resolución del problema se utilizó la técnica backtracking. Ésta no tiene un criterio de búsqueda inteligente, sino más bien sistemático. Se encarga de, en base a las restricciones, ir generando estados en profundidad buscando la mejor solución. Se decidió utilizar esta técnica ya que se consideró la más apropiada para aplicar al problema en cuestión. El algoritmo no termina hasta que el stock de libros sea nulo o hayan aprobado todos los alumnos. Cada iteración del algoritmo representa la asignación de un nuevo libro a un alumno. Hasta que el libro no tenga más ejemplares, no se va a suceder al próximo libro. Todo backtracking debe regresar si no encontró la solución por ese camino, regreso que se puede hacer desasignando los libros y reintentando esta asignación en un nuevo alumno.

### **ESTRUCTURAS DE DATOS UTILIZADAS**

Para la implementación se tomó la decisión de usar una clase Libro en la que se definió el objeto Libro con sus respectivos datos brindados por la cátedra. Los mismos son: Id, Título, Autor, Género, Páginas, Puntaje y Ejemplares. Se optó por agregar un campo al objeto llamado “EjemplaresDisponibles” que va llevando la cuenta de la cantidad de libros de ese tipo que quedan asignables. Se tuvo en cuenta que no es una correcta decisión de implementación el hecho de que el mismo objeto se encargue de llevar su propio conteo de ejemplares, pero fue una resolución aceptada por la cátedra.

Por otro lado, se implementó la clase Biblioteca que se encarga de coleccionar los libros en un map que posee como clave a la Id y como elemento al Libro. La asignación de libros es una operación disponible dentro de la clase Biblioteca. Se había optado por realizar la asignación de libros por fuera de la clase pero ésto implicaba utilizar un iterador propio de la clase Biblioteca que era constante y no permitía modificar el stock de libros, por esa razón esta elección fue rechazada.

El backtracking en particular es una operación privada de la clase Biblioteca y requiere como parámetros las siguientes variables: CantidadAlumnos, PuntajeAprobado, LibroActual (iterador del mismo tipo que la Biblioteca) y por último dos soluciones: una denominada “Solución” que contiene la mejor solución hasta el momento, y una solución parcial que se va modificando en cada iteración del algoritmo denominada “SoluciónParcial”.

Las soluciones son structs Alumnos que contienen dos campos: la cantidad de aprobados hasta el momento inicializada en 0 y un vector que contiene la información de los alumnos. Este vector es de tipo struct PerfilAlumno y a su vez contiene dos campos: una variable del puntaje acumulado hasta el momento y un set de tipo Libro que contiene los libros asignados de ese alumno. Estos dos structs citados se colocaron en un archivo alumno.h.

Inicialmente, se había discutido acerca de agregar una clase Solución/Alumno que contenga los structs mencionados anteriormente, decisión que fue descartada poco después. Se establecieron procedimientos privados para la inicialización de las soluciones, para saber si un libro fue asignado a un alumno, para saber si este alumno aprobó, y por último una función StockVacío que indica si quedan libros por asignar en la biblioteca. Es preciso aclarar que esta última función se realizó sin llevar una variable cantidad de libros para la biblioteca (algo que fue motivo de debate).

Debido a que los libros se asignan ordenadamente, para saber si todavía hay stock, basta con preguntar si el último libro de la biblioteca tiene ejemplares disponibles. Para efectuar otra iteración del backtracking es necesario saber si el libro actual en el que se está asignando todavía posee ejemplares disponibles, esto se verifica fácilmente con el campo extra del objeto Libro que se mencionó anteriormente.

## **RESULTADOS CON Y SIN PODA**

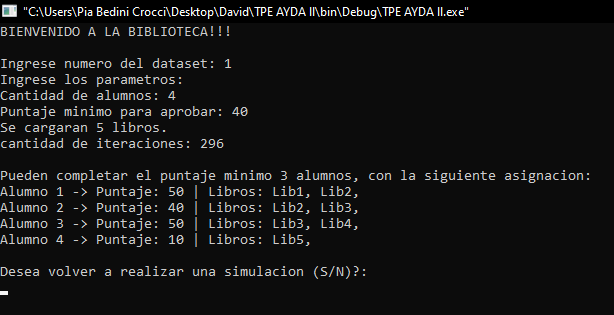
Se seleccionó un conjunto de pruebas representativas para visualizar el funcionamiento del algoritmo. Se realizaron ejecuciones con y sin poda. Se utilizó una variable que se va incrementando en cada iteración (para la entrega quedará comentada). Para omitir la poda se requirió tener en cuenta varias consideraciones. Fue necesario el uso de una nueva función llamada “AproboRecien” que indica si el **último libro** que se le asignó fue suficiente para aprobar o no. De esta manera se logra tratar de manera correcta al campo CantidadAprobados de la solución. Además fue necesario llevar registro de si quedan libros asignables hasta el momento preguntando si el iterador de libros está al final de la biblioteca. La poda, como se explicó anteriormente, verifica que un alumno no haya aprobado todavía, y como todos los alumnos son iguales, el algoritmo asegura que el alumno anterior tenga al menos un libro para que sea posible asignarle un libro al alumno actual.

Son importantes los resultados de cada testeo, ya sea con o sin poda arrojen la misma cantidad de alumnos aprobados, independientemente de los libros que se le hayan asignado a cada alumno. De ser así, se puede afirmar que el algoritmo funciona de manera correcta.

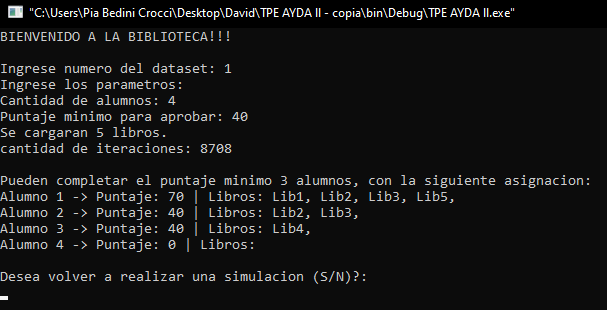
Para el primer dataset se eligieron 4 alumnos y 40 puntos, la cantidad de estados fue de 296 con poda (*captura de pantalla 1*) y 8708 sin poda (*captura de pantalla 2*).

Para el segundo dataset se eligieron 3 alumnos y 30 puntos, la cantidad de estados fue de 128386 con poda (*captura de pantalla 3)* y 3213642 sin poda (*captura de pantalla 4*).

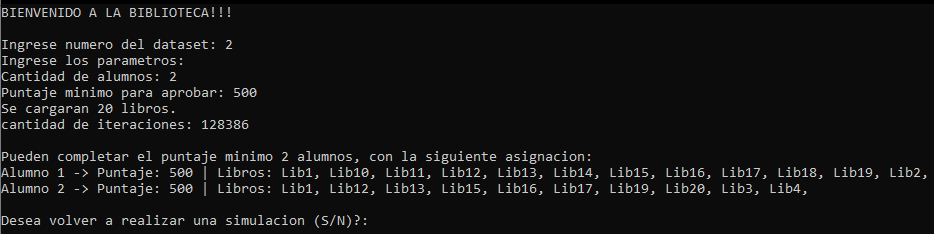
El tercer dataset tiene una cantidad de libros tan grande que, conlleva demasiado tiempo su testeo sin poda, por esa razón se optó por omitir sus pruebas.



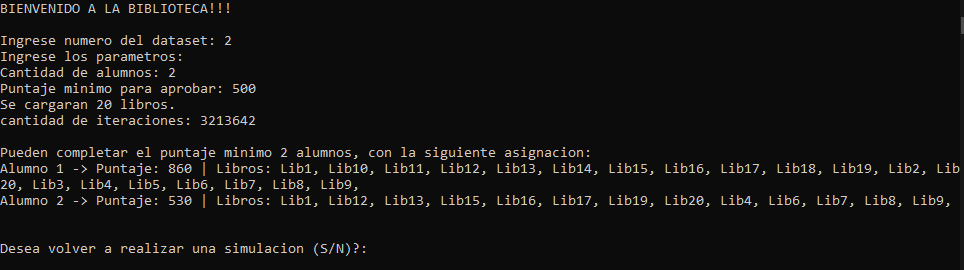
***Captura de pantalla 1: set 1 con poda***

******

***Captura de pantalla 2: set 1 sin poda***

******

***Captura de pantalla 3: set 2 con poda***

******

***Captura de pantalla 4: set 2 sin poda***

# 

# **Conclusión**

Luego de haber finalizado la primera entrega de este trabajo y haber analizado y profundizado los temas propuestos por la cátedra, se puede concluir que el backtracking es una técnica de búsqueda exhaustiva eficiente para problemas de complejidad exponencial.

Asimismo, fue posible apreciar el trabajo de un programador y la adaptación a los requerimientos de un cliente (en este caso la biblioteca de una escuela) además de trabajar con un caso verosímil.

Otra de las ventajas de la realización de este trabajo, fue el lograr interiorizar el lenguaje de programación (C++). Esto permitió fijar conceptos teórico-prácticos, poniendo a prueba, a su vez, nuestra capacidad de programar un código que responde a un problema de tal dificultad. Esto no significa que la estrategia utilizada haya sido la mejor que se podría haber empleado, pero fue la que más se adaptó a nuestros conocimientos.