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

12/11/2021

**Índice**

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

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

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

[ALTERNATIVA 1](#_vevwl5mo616f) 4

[ALTERNATIVA 2](#_dtywpl9iqvi4) 4

[ALTERNATIVA 3](#_gxxl6zsdwhv3) 5

[ALTERNATIVA SELECCIONADA](#_999qeb7hng5p) 5

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

[FUNCIONAMIENTO](#_wwcacn5i74fu) 5

[ESTRUCTURAS DE DATOS UTILIZADAS](#_jc45zimwe7zm) 6

[**ETAPA DE RESULTADOS**](#_ql4tr8vzd5pi) **7**

[**Conclusión**](#_euatamavva0u) **10**

# **Introducción**

El presente trabajo intenta abordar 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 conceptos importantes tales como: la comprensión de problemas “difíciles”, la utilidad de algoritmos heurísticos, su comparación con la técnica backtracking y cómo estos facilitan la resolución de problemas.

El contexto del problema es el mismo que el presente en la primera parte, se cuenta con una biblioteca y diferentes tipos de libros con sus respectivos puntajes. Se busca la manera de asignar estos libros a los alumnos cumpliendo con lo que imponen las nuevas condiciones.

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 primera entrega.

# **Desarrollo**

## **ANÁLISIS DEL PROBLEMA**

En esta segunda entrega, el problema principal radica en **intentar** maximizar la cantidad de alumnos aprobados en un único período. Esta vez, no se requiere garantizar que la solución sea la ideal sino que se apuntará a que el sistema responda en un mejor tiempo de ejecución (comparado con la primera entrega), logrando una asignación aproximada o de ser posible, la óptima.

Se añade un nuevo dataset de muchos libros que da a entender que esta vez aplicar backtracking no será una opción viable debido a su extenso análisis por los estados posibles que deriva en prolongados tiempos de ejecución. Por este motivo, es necesario plantear y debatir nuevas opciones para dar con la solución al problema.

A continuación se detallarán las alternativas razonadas y el motivo por el cual se optó por una de ellas.

### **ALTERNATIVA 1**

Esta opción consiste en asignar libros continuamente a un alumno hasta que éste apruebe. La asignación no tiene un criterio de selección, no es inteligente, sólo requiere un preprocesamiento de los libros para ordenarlos ascendentemente por puntaje. Luego el algoritmo los va distribuyendo sistemáticamente para intentar maximizar la cantidad de aprobados. Algunas aclaraciones sobre esta elección:

* Los alumnos tenderán a tener una gran cantidad de libros debido a que se comienzan asignando los de menor puntaje;
* Se desperdicia la menor cantidad de puntos posible;

### **ALTERNATIVA 2**

Este método es muy similar al anterior, la única diferencia radica en que el preprocesamiento se da descendentemente por puntaje. Esta alternativa posee como ventajas que el tiempo de ejecución en casos promedio es menor comparado con la alternativa 1, ya que se utilizará una menor cantidad de libros al momento de la asignación. Esto se debe a que al consignar primero los de un puntaje más elevado puede que los alumnos aprueben más rápidamente (serán necesarias menos cantidad de iteraciones). Algunas aclaraciones sobre esta elección:

* Los alumnos tenderán a tener una pequeña cantidad de libros;
* Se pueden llegar a desperdiciar muchos puntos;

### **ALTERNATIVA 3**

Una tercera alternativa para encarar el problema podría ser mediante una distribución de los ejemplares de un libro entre todos los alumnos. La asignación de libros es similar a la repartija en un juego de cartas. Consiste, como las otras alternativas, en avanzar libro por libro sin un criterio inteligente. Se toma el primero, se reparten sus ejemplares entre los alumnos y luego se repite el procedimiento con el siguiente libro comenzando desde el primer alumno sin aprobar. Algunas aclaraciones sobre esta elección:

* Los libros pueden estar preordenados por puntaje ya sea ascendente o descendentemente;
* La cantidad de iteraciones y el desperdicio de puntos puede diferir de acuerdo al orden de los libros;

### **ALTERNATIVA SELECCIONADA**

La alternativa elegida para pasar a la implementación fue la primera. Esto se debe a que se consideró la más equilibrada en cuanto tiempo de ejecución y manejo de puntuación de libros. El desperdicio de puntaje en la asignación es mínimo y su tiempo de ejecución (aunque no es el mejor en algunos casos comparado con otras opciones) es notablemente mejor al backtracking de la primera entrega en vista de la cantidad de iteraciones.

## **IMPLEMENTACIÓN DEL ALGORITMO**

### **FUNCIONAMIENTO**

Para comenzar explicando el funcionamiento del sistema de esta entrega, primero es necesario desarrollar acerca del tipo de algoritmo que se va a utilizar. A veces, en problemas “intratables” lo más favorable es encontrar una **buena** solución pero no la **mejor** debido a que no lo permiten los límites computacionales.

El grupo optó por poner en práctica un algoritmo heurístico. En este caso,al simular que se cuenta con un nuevo dataset con una gran cantidad de libros, se estaría en presencia de un problema que no puede ser resuelto mediante backtracking. La idea de un algoritmo aproximado fue descartada ya que requiere poner en efecto un factor que funciona como una cota de error y permite saber qué tan alejados se está de la solución óptima. A causa de que la búsqueda de este factor conlleva una elevada complejidad y no nos orienta en el objetivo del trabajo, la idea del algoritmo heurístico se presenta como la más acertada. Éste no tiene un criterio de búsqueda inteligente, sino más bien sistemático. El algoritmo se encarga de, en base a las restricciones, ir generando estados buscando una buena solución sin posibilidad de retroceder en esta búsqueda. Este no finaliza hasta que el stock de libros sea nulo, aprueben todos los alumnos o se hayan analizado todos los libros de la biblioteca y no sea posible continuar asignando. Cada iteración del algoritmo representa el destino de un nuevo libro a un alumno. Hasta que el mismo no haya acumulado el puntaje necesario para aprobar, no se va a suceder al próximo. Una vez que el alumno aprobó, se reinicia el próximo libro a asignar comenzando nuevamente desde el primero de la biblioteca, y si este no tiene ejemplares disponibles se pasa al siguiente.

Este algoritmo reduce considerablemente la complejidad temporal en comparación al backtracking de la primera entrega, pasando de ser exponencial a ser polinomial. Su complejidad temporal es O(A\*L) siendo A la cantidad de alumnos y L la cantidad de libros ya que, en el peor de los casos, por cada alumno será necesario revisar todos los libros de la biblioteca.

### **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.

En esta ocasión, se implementó la clase Biblioteca que se encarga de coleccionar los libros con una lista ordenada ascendentemente por puntaje. La implementación del mapa fue descartada debido a que los libros de la biblioteca simplemente serán insertados y recorridos iterativamente, sin hacer uso de funciones como *“find”* u otras que representan una ventaja temporal con esta estructura.

La asignación de libros es una operación disponible dentro de la clase Biblioteca. Esta operación es privada y requiere como parámetros las siguientes variables: CantidadAlumnos, PuntajeAprobado, y una variable de tipo struct Alumnos denominada “Solución” que se va completando en cada iteración sin posibilidad de modificaciones.

Las variables de tipo struct Alumnos 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.

Se establecieron procedimientos privados para: inicializar la solución, saber si el alumno aprobó, y por último una función StockVacío que indica si quedan libros por asignar en la biblioteca. La función que permitía saber si un libro fue asignado se desechó para esta entrega, ya que en virtud de cómo pensamos el algoritmo no es necesario verificar el incumplimiento de esta restricción.

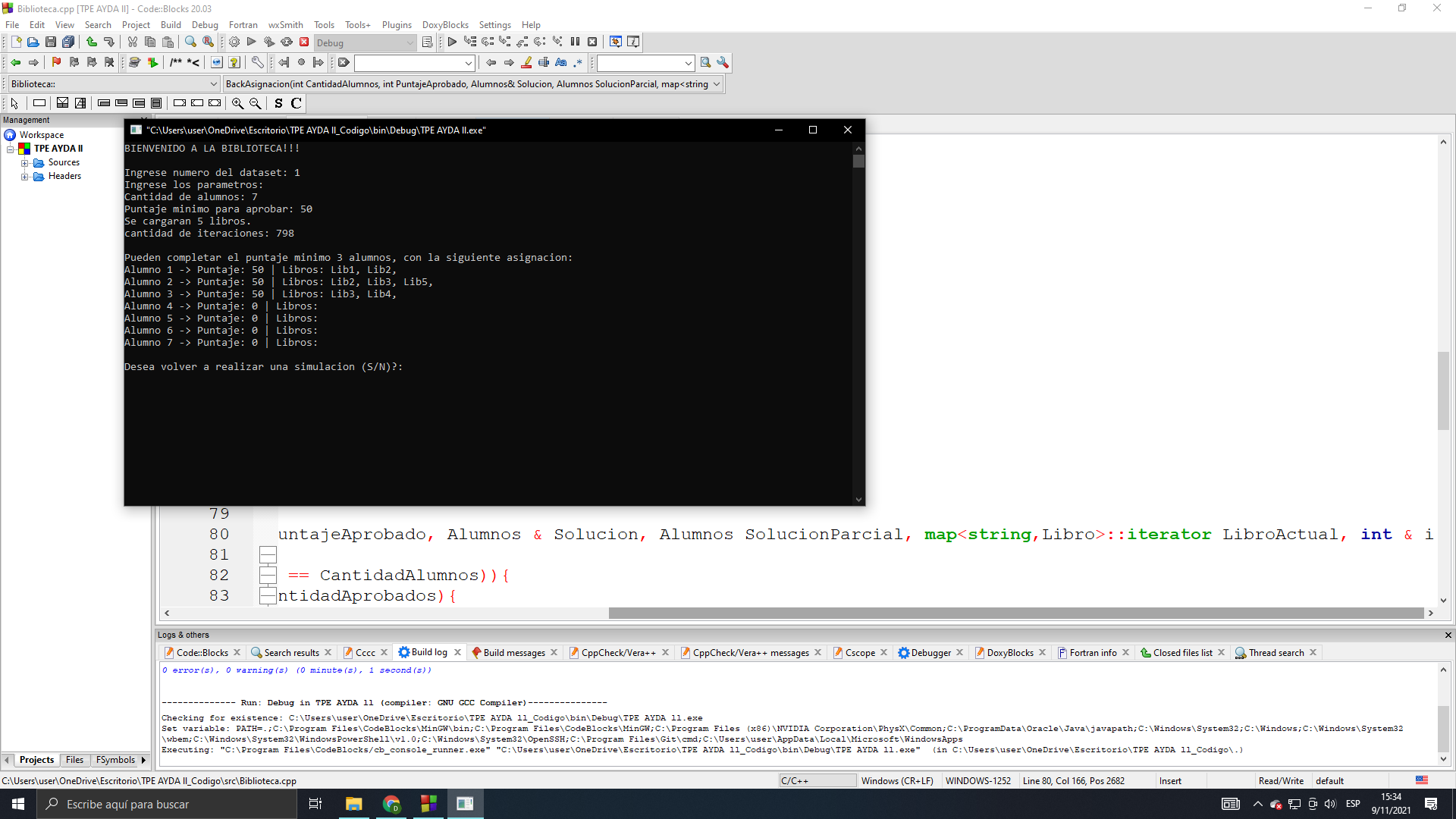
Debido a que los libros se asignan ordenadamente, para saber si todavía hay stock, basta con preguntar si el último libro de la biblioteca tiene ejemplares disponibles. Para efectuar otra iteración es necesario saber si el libro actual en el que se está asignando, todavía posee ejemplares disponibles.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

## **ETAPA DE RESULTADOS**

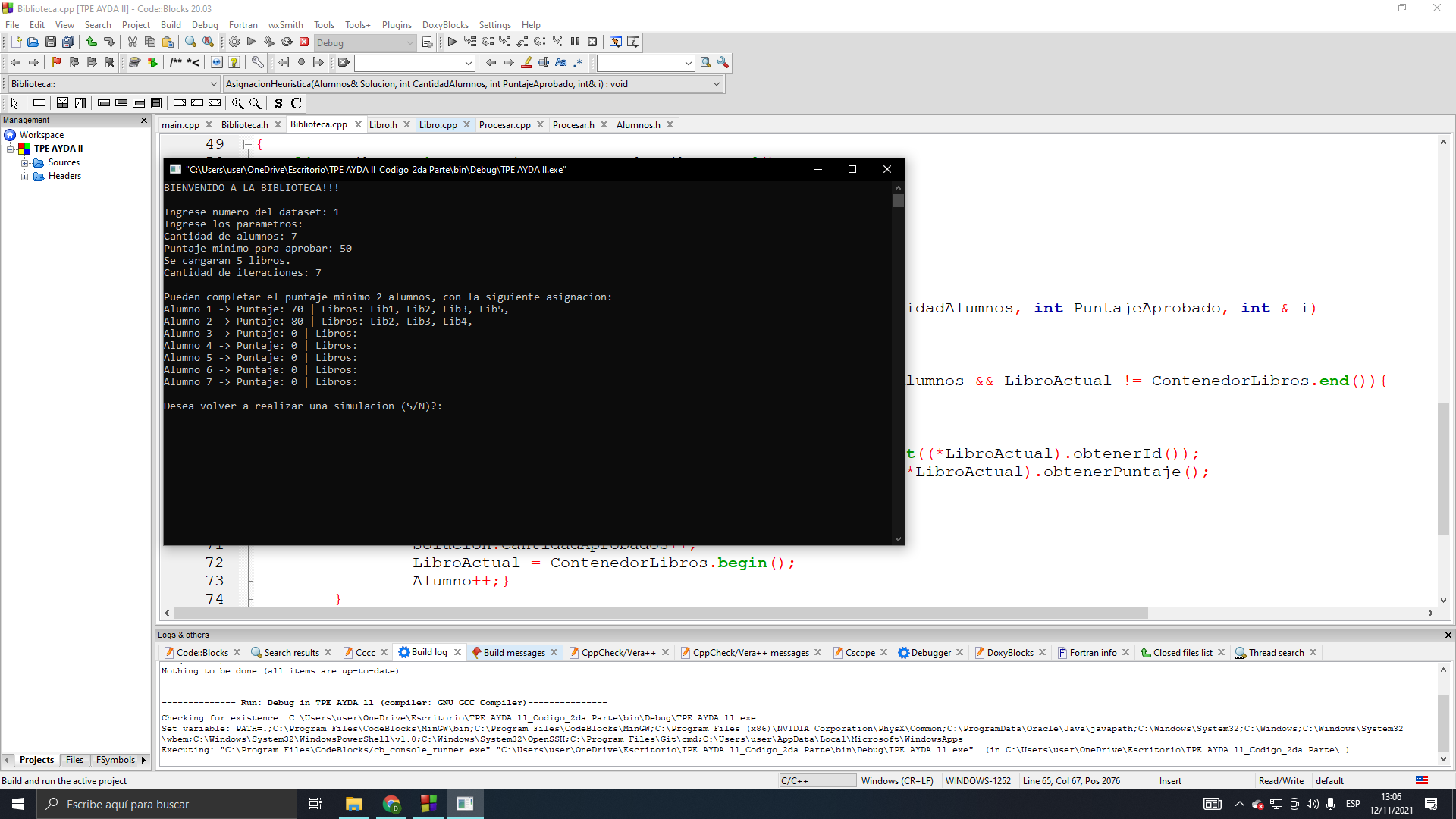
Se seleccionó un conjunto de pruebas ejemplares para visualizar el funcionamiento del algoritmo. Se utilizó una variable que se va incrementando en cada iteración, esta se modifica cuando se presenta un nuevo estado, es decir , en el momento que el algoritmo toma una decisión al asignar un libro a un alumno.

Luego, se procedió a comparar los resultados obtenidos en las dos entregas. Por una lado se presentarán los resultados de backtracking con poda y por otro los del algoritmo heurístico. Cabe aclarar que si se utilizaran pruebas de pocos alumnos y pocos puntos, probablemente por la cantidad de libros de los datasets se logrará hacer que aprueben todos y por lo tanto ambos algoritmos encontrarán la mejor solución. Por eso mismo, para obtener pruebas representativas se necesitan utilizar parámetros un poco más grandes y por esta razón los datasets 2 y 3 no encontrarán una solución en un plazo de tiempo accesible. A continuación, se presentará un testeo con el dataset 1 y con el dataset 2 (este último, para confirmar que ambos encuentran la mejor asignación con parámetros específicos).

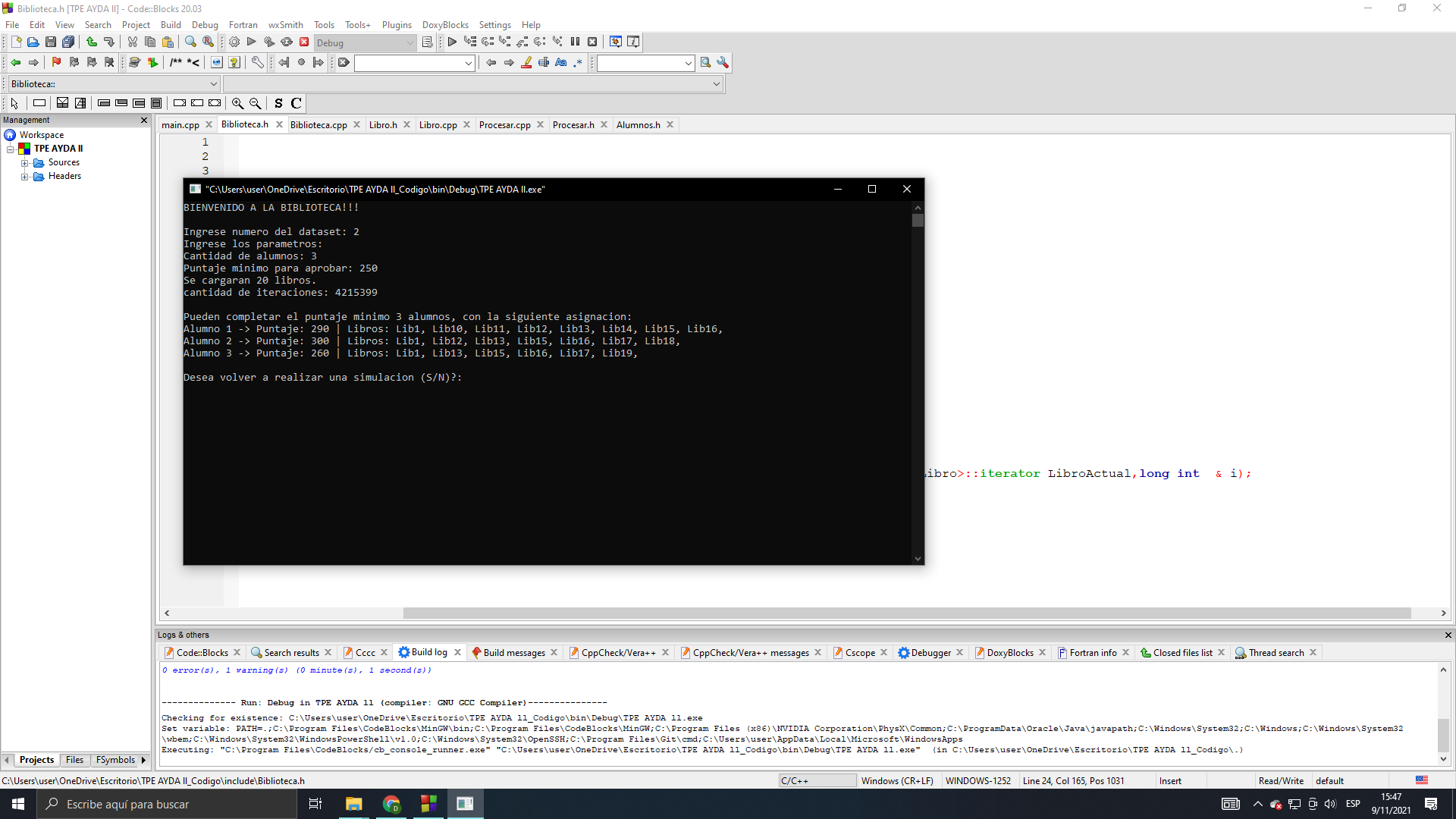
* **Dataset 1:** 7 alumnos y 50 puntos;
  + Backtracking: (*captura de pantalla 1*).
    - Iteraciones: 798
    - Cantidad de aprobados: 3
  + Heurístico: (*captura de pantalla 2*).
    - Iteraciones: 7
    - Cantidad de aprobados: 2
* **Dataset 2:** 3 alumnos y 250 puntos;
  + Backtracking: (*captura de pantalla 3*).
    - Iteraciones: 4215399
    - Cantidad de aprobados: 3
  + Heurístico: (*captura de pantalla 4*).
    - Iteraciones: 31
    - Cantidad de aprobados: 3

****

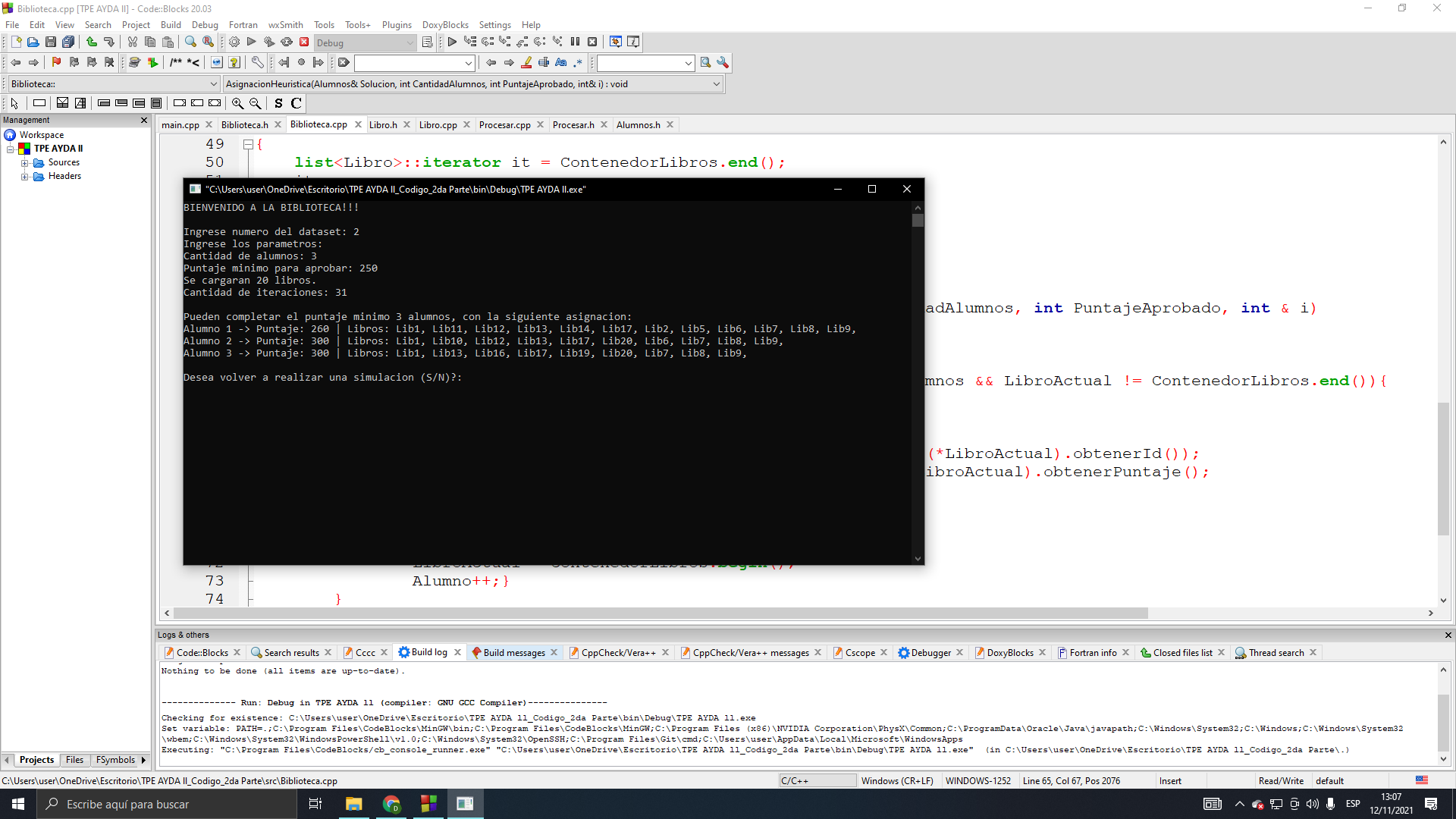
***Captura de pantalla 1: set 1 backtracking***

******

***Captura de pantalla 2: set 1 heurístico***

******

***Captura de pantalla 3: set 2 backtracking***

******

***Captura de pantalla 4: set 2 heurístico***

### **OBSERVACIONES**

Se puede observar que, comparando la cantidad de estados generados (iteraciones), el algoritmo heurístico supera ampliamente al backtracking. En cuanto a la calidad de la solución de la prueba con el dataset 1, se visibiliza que el backtracking retorna la asignación óptima con una cantidad de alumnos aprobados mayor que con el algoritmo heurístico. En cambio, en las capturas referidas al dataset 2 se percibe que la cantidad de aprobados es la misma con ambos algoritmos en vista de que los parámetros ingresados permiten alcanzar la solución óptima.

Basándonos en estas observaciones, si simuláramos una prueba de muchos alumnos y una considerable cantidad de puntos con el nuevo dataset, sin tener en cuenta los límites computacionales, se debería concluir que el backtracking tendrá más aprobados que el algoritmo heurístico. Si la prueba fuera de pocos alumnos, ambos algoritmos encontrarán la mejor solución.

# 

# **Conclusión**

Luego de haber finalizado el trabajo en su totalidad 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 y que en términos de tiempo de ejecución, es conveniente el algoritmo heurístico.

En resumen, si siempre se quiere obtener la solución óptima, es idóneo utilizar la técnica backtracking. En cambio, si lo que se busca es acortar el tiempo de ejecución, iterando la menor cantidad de veces posible y no garantizando siempre la mejor solución, es ventajoso utilizar heurística.

Es importante destacar que ningún algoritmo es mejor que otro, se puede optar por alguno de ellos dependiendo de las necesidades del problema y de la magnitud de la complejidad del mismo.