

IE-0217 Estructuras abstractas de datos y algoritmos para ingeniería

Proyecto 1: ABB vs Árboles Rojo y Negro

Timna Belinda Brown Ramírez

B61254

`timna.brown@ucr.ac.cr`

`belindabrownr@gmail.com`

I-2019

Tabla de contenidos

1. Reseña del programa	2
2. Funcionamiento del programa	2
3. Experimentos realizados	2
4. Marco teórico	2
5. Resultados obtenidos	4
6. Conclusiones	10
7. Apéndice	14
7.1. Código fuente	15

1. Reseña del programa

El objetivo de este proyecto es crear un algoritmo que desarrolle el funcionamiento de los Árboles Búsqueda Binaria (ABB) y los Árboles Rojo y Negro con el fin de realizar un análisis de la complejidad de cada uno y su debida comparación.[2]

2. Funcionamiento del programa

El proyecto fue desarrollado en C++, está conformada por una carpeta nombrada Proyecto1_B61254denton

El algoritmo de negro y rojo toma como fuente un los datos de entrada por el usuario y el de búsqueda binaria corre pruebas ya creadas.

3. Experimentos realizados

Se realizaron pruebas para asegurar que el código no tuviera errores de sintáxis por lo que se compiló, sucesivamente se realizaron las modificaciones necesarias para que ejecutara adecuadamente las instrucciones generales. Con lo que respecta al sistema operativo en el que funciona, es ejecutable sobre cualquier plataforma que interprete C++ por medio de su propio intérprete. El programa recibe las instrucciones realizadas por medio del teclado con el fin de volver más accesible su ejecución.[4]

Por otra parte, respecto a las variables de respuesta. Por lo mencionado, fue necesaria la creación y consumación de pruebas experimentales tal que se validan mediante test unitarios ya fue necesario descomponer las funciones de los árboles en comportamientos cualificables. [3]

4. Marco teórico

Un ABB tiene la característica de que todos los subelementos de un nodo a su izquierda están los menores y a su derecha los mayores almacenados. Por otr lado, la complejidad de los árboles negro y rojo es mayor, ya que es un árbol de búsqueda binaria de datos con asignación de colores.

Poseen las siguientes características: Los nodos tienen un color propio ya sea rojo o negro. Aparte de todo lo que cumple un árbol BB se debe cumplir lo siguiente:

- Nodo rojo o negro
- Raíz negra
- Hojas negras
- Hijos de todo nodo rojo son negros
- Cada camino tiene la misma cantidad de nodos negros y esto se denomina altura del árbol

5. Resultados obtenidos

```
Belindas-MacBook-Air:Arboles_ABB belindabrown$ make
g++ -g --std=c++11 -Wall *.cpp -o a.exe
./a.exe

*****

Tester de arboles ABB #1

*****

-----

Agregando valores

-----

Agregar 35
Agregar 12
Agregar 50
Agregar 8
Agregar 17
Agregar 32
Agregar 53
Agregar 3
Agregar 10
Agregar 29
Agregar 57

-----

Comprobando cantidad de elementos entre árboles

-----

Como resultado obtenemos que:
No son semejantes

-----

Prueba de altura entre árboles

-----

Como resultado obtenemos que:
No son iguales en altura
```

Figura 1: Resultadps ABB

```

-----

Prueba de espacio en los árboles

-----

Realizando un recorrido a lo ancho del primer árbol
35 12 50 8 17 53 3 10 32 57 29

Realizando un recorrido a lo ancho del segundo árbol
53 12 57 8 50 59412480 3 10 17 32 29
Como resultado obtenemos que:
No esta lleno

Como resultado obtenemos que:
No esta completo

-----

Eliminado hojas

-----

Borrando hoja 29 1
Realizando un recorrido a lo ancho35 12 50 8 17 53 3 10 32 57
Borrando nodo con 1 hijo 53 1
Realizando un recorrido a lo ancho35 12 50 8 17 57 3 10 32
Borrando nodo con dos hijos 121
Realizando un recorrido a lo ancho35 8 50 3 17 57 10 32

*****

Tester de arboles ABB #2

*****

El arbol (29,11,28,18,31): No esta lleno
El arbol (29,11,28,18,31,8): No esta lleno
El arbol (29,11,28,18,31,8) No esta completo
El arbol (29,11,28,18,31,8,9,5,10,12) Como resultado obtenemos que:
No esta completo

```

Figura 2: Resultados ABB

```

Borrando nodo con 1 hijo 53 1
Realizando un recorrido a lo ancho 35 12 50 8 17 57 3 10 32
Borrando nodo con dos hijos 121
Realizando un recorrido a lo ancho 35 8 50 3 17 57 10 32

*****

Tester de arboles ABB #2

*****

El arbol (29,11,28,18,31): No esta lleno
El arbol (29,11,28,18,31,8): No esta lleno
El arbol (29,11,28,18,31,8) No esta completo
El arbol (29,11,28,18,31,8,9,5,10,12) Como resultado obtenemos que:
No esta completo
Los arboles (50,46,12,23,48,53,57,68,80) y (28,29,9,7,11,32,12,90,17)
Como resultado obtenemos que:
No son isomorfos
Los arboles(50,46,12,23,48,53,57,68,80) y (53,80,23,57,46,12,50,68,48)
Como resultado obtenemos que:
Si son semejantes

////////////////////////////////////

Vamos a realizar las siguientes operaciones:

////////////////////////////////////

3 10 8 17 32 35 50 57
Borrando hoja 32 1
3 10 8 17 35 50 57
Borrando hoja 29 0
3 10 8 17 35 50 57
Borrando hoja 28 0
3 10 8 17 35 50 57
Borrando hoja 7 0
3 10 8 17 35 50 57
Borrando hoja 12 0
3 10 8 17 35 50 57
Borrando hoja 90 0
3 10 8 17 35 50 57
Borrando hoja 17 1
3 10 8 35 50 57
Borrando hoja 9 0
3 10 8 35 50 57
Borrando hoja 11 0
3 10 8 35 50 57
rm a.exe
rm -rf a.exe.dSYM
Belindas-MacBook-Air:Arboles_ABB belindabrown$ █

```

Figura 3: Resultados ABB

```

Belindas-MacBook-Air:Arboles_Rojo_y_Negro belindabrown$ make
g++ -g --std=c++11 -Wall *.cpp -o a.exe
./a.exe

*****
Árbol rojo y negro
*****

1. Agregar elemento al árbol
2. Buscar un elemento
3. PRE-ORDER
4. POST-ORDER
5. Eliminar un elemento del árbol
6. Salir
*****
Escoja una opción >>>> 1

Va a ser insertado ... 45
-----
Se agregó el valor
-----

>>>>Escoja una opción >>>> 1

Va a ser insertado ... 23
-----
Se agregó el valor
-----

>>>>Escoja una opción >>>> 1

Va a ser insertado ... 45 67
-----
Se agregó el valor
-----

>>>>Escoja una opción >>>> Favor digite una opción dentro del menú

>>>>Escoja una opción >>>> 1

Va a ser insertado ... 67
-----
Se agregó el valor
-----

>>>>Escoja una opción >>>> 1

Va a ser insertado ... 2
-----
Se agregó el valor
-----

```

Figura 4: Resultados Negro y Rojo

```

Va a ser insertado ... 3
-----
Se agregó el valor
-----

****Escoja una opción **** 1

Va a ser insertado ... 1
-----
Se agregó el valor
-----

****Escoja una opción **** 1

Va a ser insertado ... 1
-----
Se agregó el valor
-----

****Escoja una opción **** 1

Va a ser insertado ... 4
-----
Se agregó el valor
-----

****Escoja una opción **** 1

Va a ser insertado ... 78
-----
Se agregó el valor
-----

****Escoja una opción **** 3
Pre-Order **** 45[Negro]**** 3[Rojos]**** 1[Negro]**** 1[Rojos]**** 2[Rojos]**** 23[Negro]**** 4[Rojos]**** 67[Rojos]**** 45[Rojos]**** 78[Rojos]

****Escoja una opción **** 4
Post-Order **** 1[Rojos]**** 2[Rojos]**** 1[Negro]**** 4[Rojos]**** 23[Negro]**** 3[Rojos]**** 45[Rojos]**** 78[Rojos]**** 67[Rojos]**** 45[Negro]

****Escoja una opción **** 2

Va a ser buscado ****1
Se encontró el valor ****

****Escoja una opción **** 3
Pre-Order **** 45[Negro]**** 3[Rojos]**** 1[Negro]**** 1[Rojos]**** 2[Rojos]**** 23[Negro]**** 4[Rojos]**** 67[Rojos]**** 45[Rojos]**** 78[Rojos]

****Escoja una opción **** 4
Post-Order **** 1[Rojos]**** 2[Rojos]**** 1[Negro]**** 4[Rojos]**** 23[Negro]**** 3[Rojos]**** 45[Rojos]**** 78[Rojos]**** 67[Rojos]**** 45[Negro]

****Escoja una opción ****

```

Figura 5: Resultados Negro y Rojo


```

Va a ser insertado ... 1
-----
Se agregó el valor
-----

****Escoja una opción **** 1

Va a ser insertado ... 4
-----
Se agregó el valor
-----

****Escoja una opción **** 1

Va a ser insertado ... 78
-----
Se agregó el valor
-----

****Escoja una opción **** 3
Pre-Order **** 45[Negro]**** 3[Rojos]**** 1[Negro]**** 1[Rojos]**** 2[Rojos]**** 23[Negro]**** 4[Rojos]**** 67[Rojos]**** 45[Rojos]**** 78[Rojos]

****Escoja una opción **** 4
Post-Order **** 1[Rojos]**** 2[Rojos]**** 1[Negro]**** 4[Rojos]**** 23[Negro]**** 3[Rojos]**** 45[Rojos]**** 78[Rojos]**** 67[Rojos]**** 45[Negro]

****Escoja una opción **** 2

Va a ser buscado ****4
Se encontró el valor *****

****Escoja una opción **** 3
Pre-Order **** 45[Negro]**** 3[Rojos]**** 2[Negro]**** 1[Rojos]**** 23[Negro]**** 4[Rojos]**** 67[Rojos]**** 45[Rojos]**** 78[Rojos]

****Escoja una opción **** 4
Post-Order **** 1[Rojos]**** 2[Rojos]**** 1[Negro]**** 4[Rojos]**** 23[Negro]**** 3[Rojos]**** 45[Rojos]**** 78[Rojos]**** 67[Rojos]**** 45[Negro]

****Escoja una opción **** 5

****Eliminar el valor **** 1

****Escoja una opción **** 3
Pre-Order **** 45[Negro]**** 3[Rojos]**** 2[Negro]**** 1[Rojos]**** 23[Negro]**** 4[Rojos]**** 67[Rojos]**** 45[Rojos]**** 78[Rojos]

****Escoja una opción **** 4
Post-Order **** 1[Rojos]**** 2[Negro]**** 4[Rojos]**** 23[Negro]**** 3[Rojos]**** 45[Rojos]**** 78[Rojos]**** 67[Rojos]**** 45[Negro]

****Escoja una opción **** 6

Salir *****
rm a.exe
rm -rf a.exe.dSYM
Belinda-MacBook-Air:Arboles_Rojo_y_Negro belindabrown$

```

Figura 6: Resultados Negro y Rojo

6. Conclusiones



Figura 7: Estadística de código completo ABB



Figura 8: Estadística de código completo árbol rojo y negro

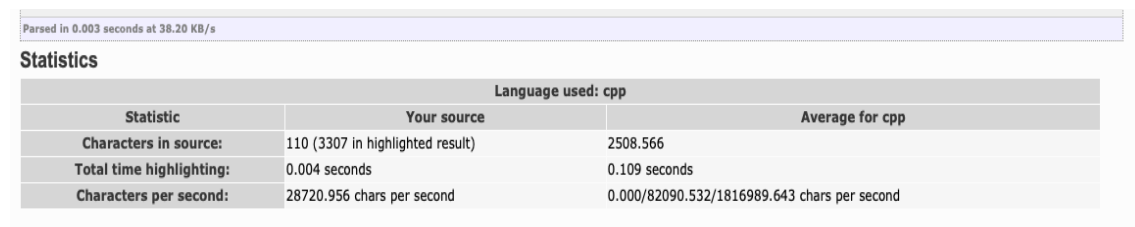


Figura 9: Estadística de insertar en ABB

Parsed in 0.006 seconds at 263.65 KB/s		
Statistics		
Language used: cpp		
Statistic	Your source	Average for cpp
Characters in source:	1726 (12515 in highlighted result)	2508.533
Total time highlighting:	0.007 seconds	0.109 seconds
Characters per second:	238364.516 chars per second	0.000/82097.107/1816989.643 chars per second

Figura 10: Estadística de insertar en árbol rojo y negro

Parsed in 0.003 seconds at 38.20 KB/s		
Statistics		
Language used: cpp		
Statistic	Your source	Average for cpp
Characters in source:	110 (3307 in highlighted result)	2508.566
Total time highlighting:	0.004 seconds	0.109 seconds
Characters per second:	28720.956 chars per second	0.000/82090.532/1816989.643 chars per second

Figura 11: Estadística de eliminar en ABB

Parsed in 0.003 seconds at 38.20 KB/s		
Statistics		
Language used: cpp		
Statistic	Your source	Average for cpp
Characters in source:	110 (3307 in highlighted result)	2508.566
Total time highlighting:	0.004 seconds	0.109 seconds
Characters per second:	28720.956 chars per second	0.000/82090.532/1816989.643 chars per second

Figura 12: Estadística de eliminar en árbol rojo y negro

Parsed in 0.020 seconds at 342.60 KB/s		
Statistics		
Language used: cpp		
Statistic	Your source	Average for cpp
Characters in source:	6994 (48134 in highlighted result)	2508.936
Total time highlighting:	0.021 seconds	0.109 seconds
Characters per second:	327315.111 chars per second	0.000/82128.441/1816989.643 chars per second

Figura 13: Estadística de implementación de ABB

Parsed in 0.005 seconds at 232.39 KB/s		
Statistics		
Language used: cpp		
Statistic	Your source	Average for cpp
Characters in source:	1295 (10551 in highlighted result)	2508.885
Total time highlighting:	0.006 seconds	0.109 seconds
Characters per second:	206997.854 chars per second	0.000/82133.694/1816989.643 chars per second

Figura 14: Estadística de implementación de árbol rojo y negro

Referencias

- [1] Kroah-Hartman G Corbet. J, Rubini. A. *Linux Coding*. O'Reilly books, 1998.
- [2] Computer Science Labs. *Tecnology- commands*. O'Reilly books, 2018.
- [3] Mark Summerfield. *Programming in C++: A Complete Introduction to the C++ Language*. Anaya Multimedia, 2009.
- [4] A. M. Turing. *On computable numbers with an application to the Entscheidungs problem*. Proceedings of the london mathematical society, 1997.

7. Apéndice

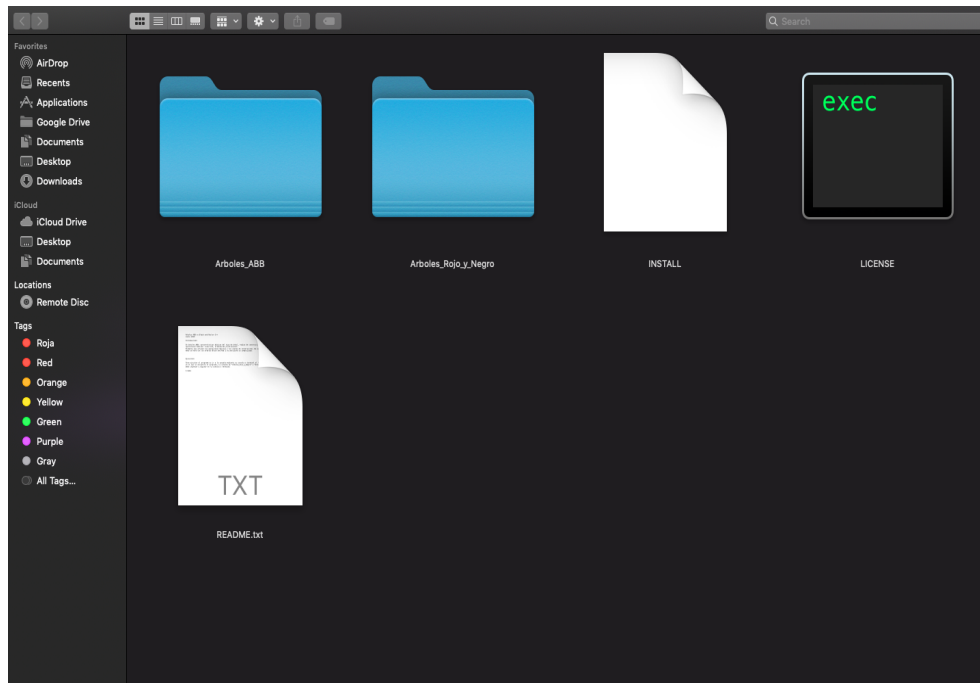


Figura 15: Dentro de Proyecto1_B61254

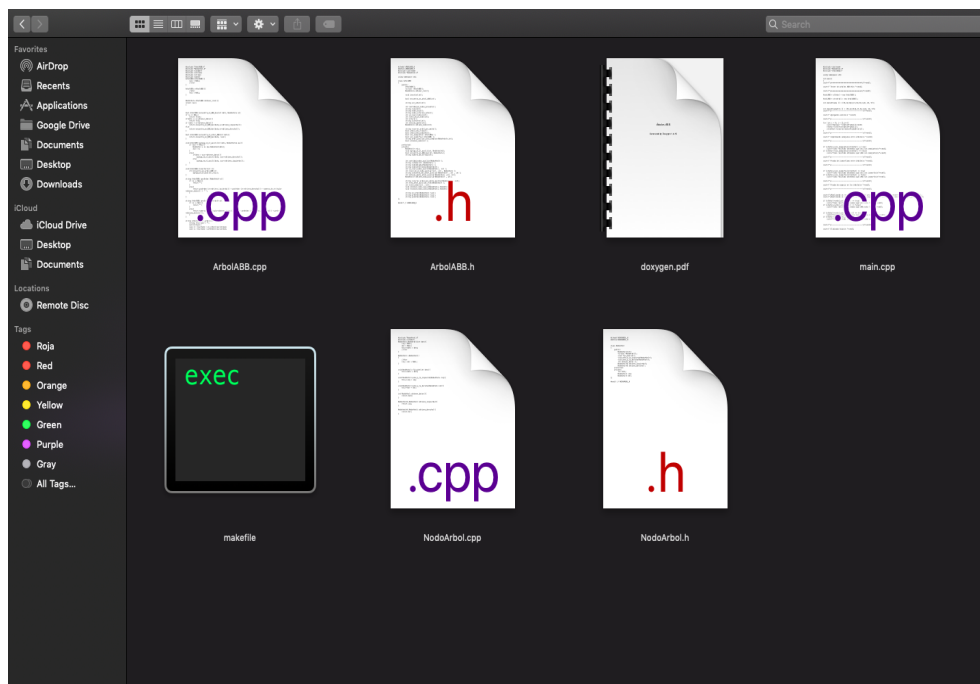


Figura 16: Dentro de Arboles_ABB

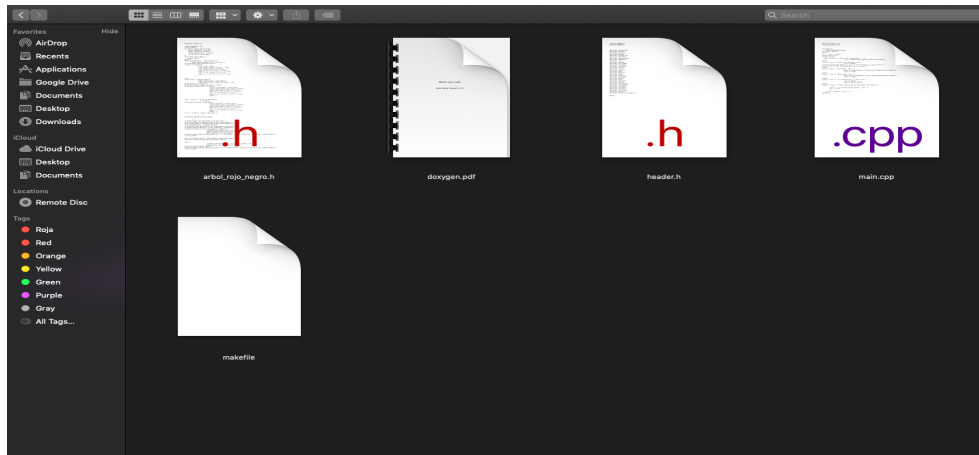


Figura 17: Dentro de Arboles_Rojo_y_Negro

7.1. Código fuente

Todos tienen en común lo siguiente:

```
Arboles ABB vs Black and Red en C++
Junio 2019

Introducción:

En árboles ABB: características básicas del tipo de árbol, reglas de construcción,
operaciones básicas: inserción, eliminación, localización.
Ejemplos que aclaran las operaciones básicas y las reglas de construcción. De este mismo
modo se hará con los árboles Black and Red y se analizará su complejidad.

Ejecución:

Para ejecutar el programa es ir a la carpeta mediante su consola o terminal al folder
en el que se encuentra el programa, la carpeta de "Arboles_Rojo_y_Negro" o "Arboles_ABB"
Debe ingresar y digitar en la consola o terminal:

$ make
```

Figura 18: Readme[2]

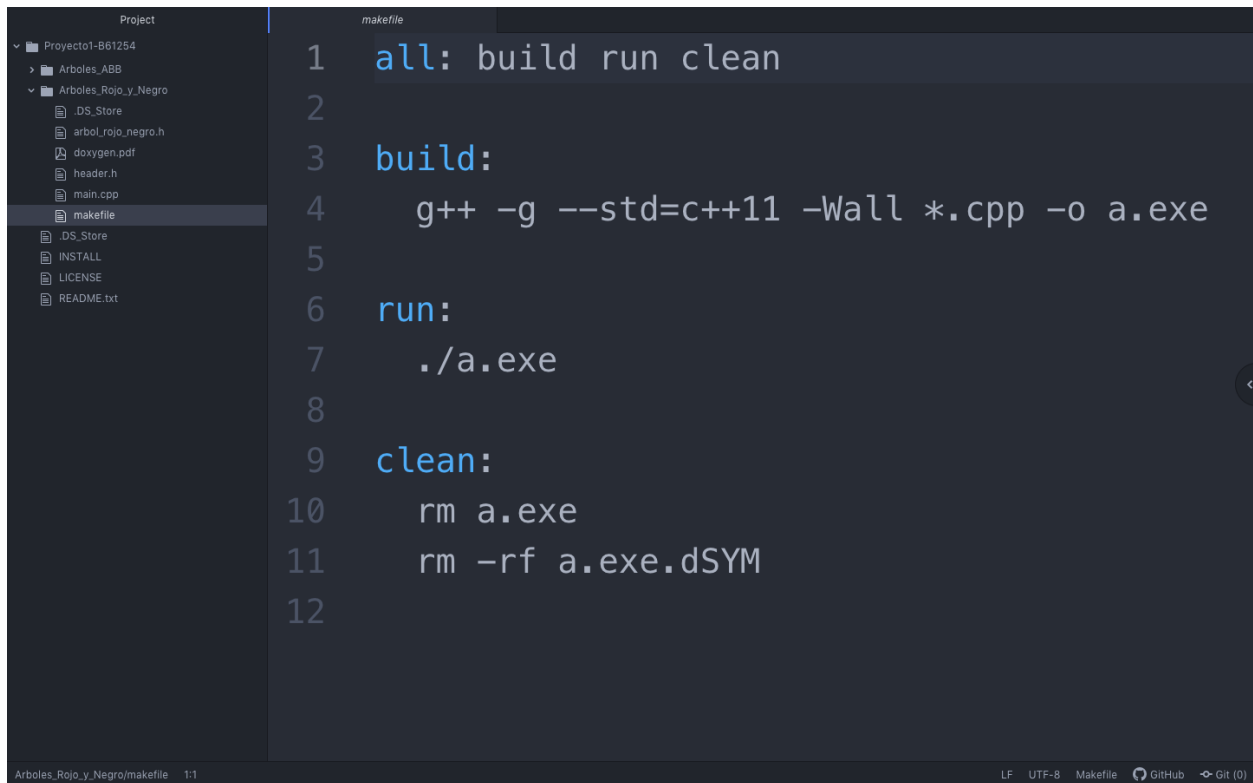
Proyecto comparación de árboles ABB vs black and red en C++

License Apache 2.0

Se distribuye un Makefile con 3 reglas:

- * build: compila los fuentes.
- * run: ejecuta un corrida de ejemplo.
- * clean: borra los binarios.

Figura 19: Install



```
1 all: build run clean
2
3 build:
4     g++ -g --std=c++11 -Wall *.cpp -o a.exe
5
6 run:
7     ./a.exe
8
9 clean:
10    rm a.exe
11    rm -rf a.exe.dSYM
12
```

Figura 20: Makefile[1]

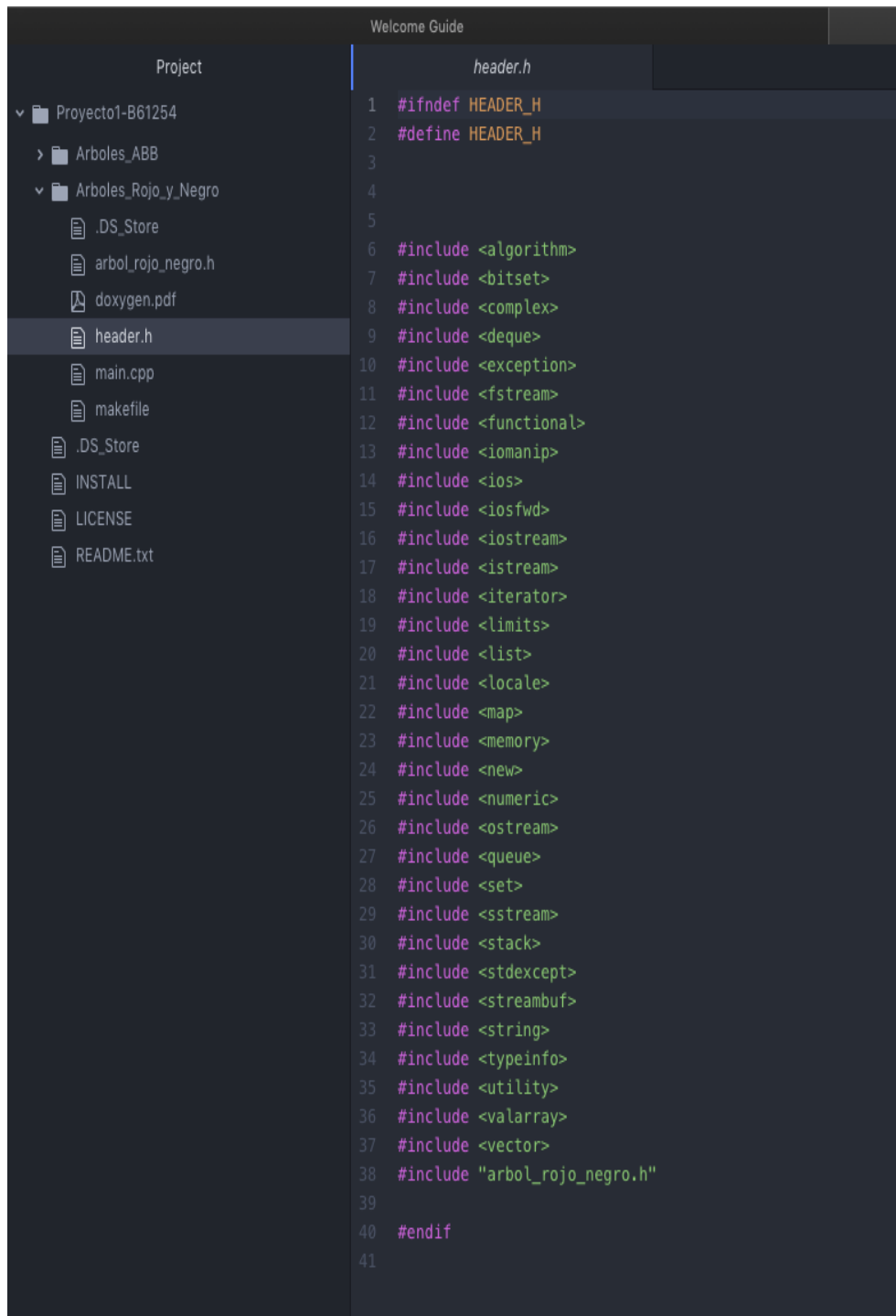


Figura 21: Código de header.py