**Programación II**

**Grupo: 01**

**Profesor**

**Dr. Edgar Casasola Murillo**

# Comparación rendimiento map STL vs vector STL vs ArbolRojoNegro

**Javier Alfredo Solano Saltachín C17632**

**Omar Fabián Camacho Calvo C11476**

**Archibald Emmanuel Carrion Claeys C01736**

**Alberto González Avendaño C13248**

**31 de julio de 2022**

**ENUNCIADO**

Objetivo de aprendizaje: Dominio de creación de contenedores reutilizables utilizando la técnica de Plantillas y comparación de rendimiento contra Biblioteca Estándar de Plantillas STL.

En esta tarea usted desarrollará una plantilla un árbol para almacenar pares llave-valor (Key,Value) , y será un árbol con inserción en la hojas manteniéndolo equilibrado con la técnica de balanceo de árbol Rojo-Negro. El árbol se debe comparar contra dos contenedores de la STL (un map y un vector con pares (Key,Value) ) Si usted ordena el vector antes de buscar las palabras, el tiempo de búsqueda en el vector debe incluir el tiempo que tomó ordenarlo.

Deberá crear una plantilla de función para llevar a cabo las pruebas de rendimiento.

Para simplificar la tarea, haremos una comparación simple de la siguiente forma:

1. Cargar en un string[ N ] con las N palabras que utilizarán para cada ejecución del experimento. Esto dará igualdad de condiciones para cada uno de los contenedores en los diferentes experimentos.
   Las palabras serán tomadas en secuencia a partir de un archivo de texto que se les proporciona. Cada ejecución tomará cantidades diferentes según se indica en el punto 6.
2. En cada ejecución se deberá cargar en el contenedor que se evalúa, un par compuesto de una palabra y su siguiente palabra. Dicho de otra forma, para cada palabra se va a almacenar **solamente** "la primera aparición de una palabra que aparece inmediatamente después de ella en el texto", si una palabra no aparece es porque no estaba en el texto, a la última palabra cargada en el vector se asocia una palabra vacía "". Si una palabra ya está en el contenedor se descarta, porque ya tiene otra palabra que fue su primera aparición como consecutiva.
3. Para cada ejecución con su ArbolRN: Calcular el tiempo de búsqueda promedio de todas las palabras alternando cada palabra existente con una palabra inválida construida agregando la terminación "NoIsE" a la palabra original. Lo que quiere decir es que se debe buscar una palabra que sí está y luego la palabra inválida construida que se sabe que no está. La búsqueda deberá seguir el orden en que

aparecen en el vector de lectura, sin ordenar y sin eliminar repetidas. No se debe imprimir nada ni guardar en un archivo para que el tiempo de entrada/salida no afecte los cálculos del tiempo de búsqueda en el contenedor.

4. Con el map de la STL se repite el proceso del paso 3 ( usar método emplantillado para tal efecto ).
5. Con un vector de la STL que almacene pares del tipo (K,V) llave/valor, se repite el proceso del paso 3 ( recuerde que si lo ordena previamente deberá sumar ese tiempo al tiempo total antes de promediar ).
6. Escriba en pantalla el tiempo promedio de búsqueda de los tres contenedores al final de una ejecución con:

A) Contenedores con 1.000.000 palabras y su primera palabra siguiente.

A1) Con esos contenedores calcular los primeros 10 tiempos promedio de la siguiente forma: Inicie en 10.000 y terminando en 100.000, calcule el tiempo promedio cada 10.000 palabras y sus 10.000 palabras inválidas correspondientes.

A2) Con esos contenedores calcular los primeros 10 tiempos promedio iniciando en 100.000 e incrementando en 100.000 palabras y sus 100.000 palabras inválidas correspondientes.

La salida de cada experimento A1 y A2 será una tabla con 3 contenedores ( filas ) y 10 columnas de tiempos promedios para cada uno. En la documentación muestre un gráfico comparativo usando el graficador de alguna hoja de cálculo. (LibreOffice o Excel).

B) OPCIONAL POR 10% EXTRA SOBRE NOTA DEL RUBRO DE TAREAS

B1. ¿Puede hacer el experimento con el total de palabras en el archivo datos.txt? (39.151.187 PALABRAS) B2. Con búsquedas en bloques de 3.900.000 palabras válidas y sus correspondientes inválidas.

B3. Caso contrario ¿con cuánto fue el máximo que su grupo pudo calcular los 10 tiempos promedio?

Se adjunta un archivo para pruebas llamado datos.txt disponible en:
https://drive.google.com/file/d/1bjrg6J0kqdIGAghQrocppAqEe-wWUq-5/view?usp=sharing

## FUNCIONAMIENTO DEL PROGRAMA

Para resolver el problema planteado decidimos trabajar con un main.cpp que contiene la mayoría de los métodos que ocuparemos para realizar las pruebas tiempo con los objetos vector<pair<string, string>>, map<string, string>, y ArbolRojoNegro (clase que hicimos nosotros mismos).

**Clases usadas:** Con la finalidad de lograr resolver el problema planteado, decidimos utilizar la siguiente distribución de clases y métodos para maximizar la eficacia de nuestra solución y desarrollar una solución viable y segura:

**ArbolRojoNegro**: es la clase más importante del programa, es nuestra estructura de datos, la llenamos al inicio de la ejecución y después se realizan las pruebas de velocidad. Es importante notar que esta clase es emplantillada para funcionar con una clase K (llave) y V (value) que funciona de manera similar al objector pair<x, y> de la STL.

**Predicado**: es una clase estática que sirve para encontrar una pareja o una dupla en el vector<pair<key, value>> de la STL tomando como criterio de búsqueda el primer valor de la dupla(llave).

## COMANDOS DE COMPILACIÓN Y EJECUCIÓN

Para poder crear manejar todas las pruebas anteriores y llenar los contenedores, usamos un main.cpp que recibe por parámetros todos los parámetros de ejecución.
De esa manera nuestro código es lo más flexible posible.

Para compilarlo usamos el siguiente método:

***g++ *.cpp -o output -std=c++17***

Para poder ejecutar el código usamos el siguiente comando:

***./output.out***

Una vez compilado tenemos en la carpeta donde se ejecuta el código un archivo donde se encuentra el código compilado.

## PARÁMETROS DE EJECUCIÓN

Se ocupa dar un valor inicial a las 4 variables, alimentadas mediante la entrada estándar (cin) en el siguiente orden (separadas por un espacio, debe sustituir apropiadamente), y el resultado tabulado se imprime en la salida estándar (cout).

***cantidadElementosLectura pasoPruebaInicial cantidadPruebas nombreArchivo***

Se puede redirigir la entrada estándar del programa para que acepte un archivo de texto como fuente de los parámetros, y se puede redirigir la salida estándar a otro archivo de texto (será sobreescrito) para tabular los resultados en el software de preferencia.

***./output.out < archivoDeParametros.txt > archivoDeTabulacion.txt***

## ENSAYOS DE EXPERIMENTO A BREVE ESCALA

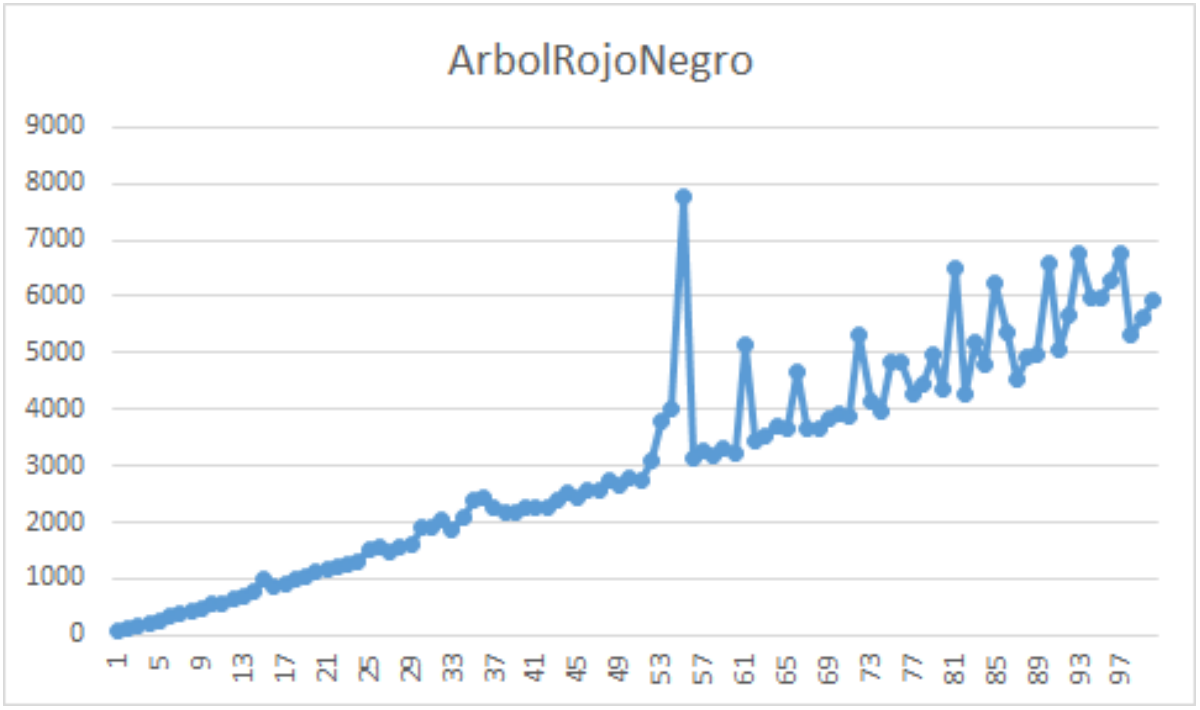
Para realizar una prueba a pequeña escala, recreamos el experimento principal con un menor tamaño de bloque de palabras inicial, y menor cantidad de incrementos subsiguientes. En esta pequeña prueba, por consiguiente, insertamos menos cantidades de palabras en los contenedores.

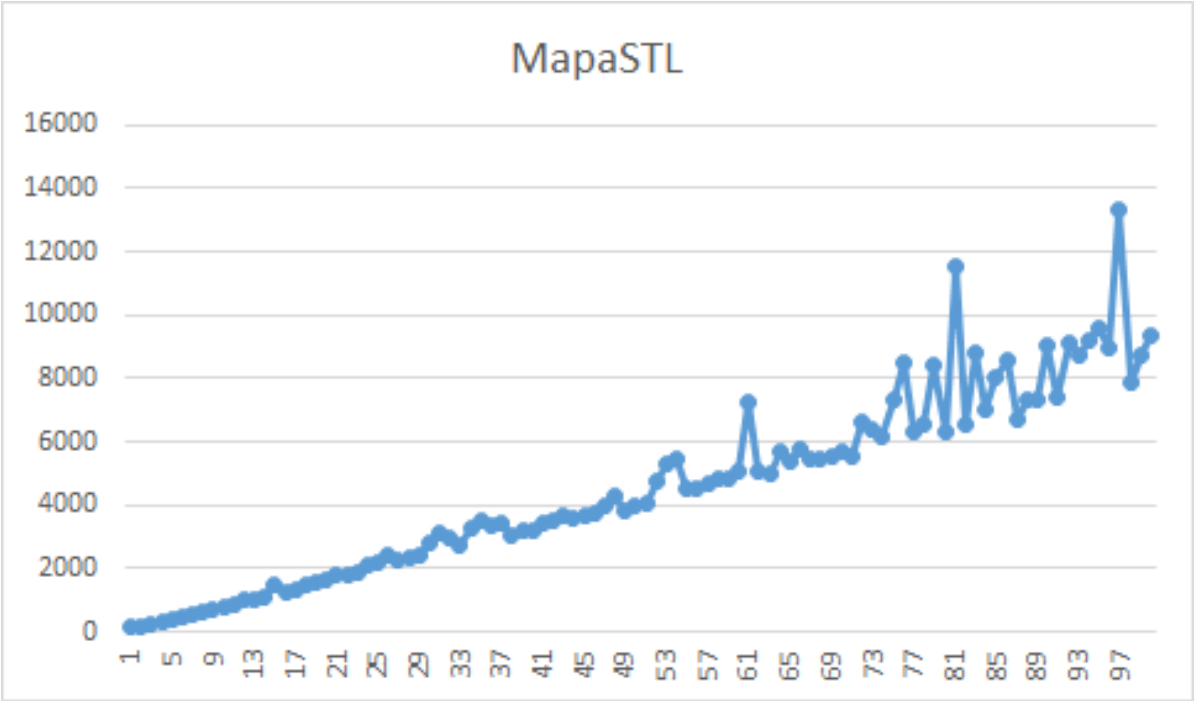
| **PARÁMETROS DE EXPERIMENTO** | | |
|---|---|---|
| *Nombre del parámetro* | *Valor ingresado* | *Breve descripción* |
| cantidadElementosLectura | 30000 | La cantidad de palabras que serán cargadas, considerando repeticiones |
| pasoPruebaInicial | 100 | Tamaño de lote de palabras inicial. Con cada iteración crece en esta cantidad. |
| cantidadPruebas | 100 | Cantidad de iteraciones del experimento. En cada experimento se busca en 10 lotes y se promedian sus tiempos. |
| nombreArchivo | datos.txt | Nombre del archivo cuyos contenidos reflejarán los resultados tabulados |

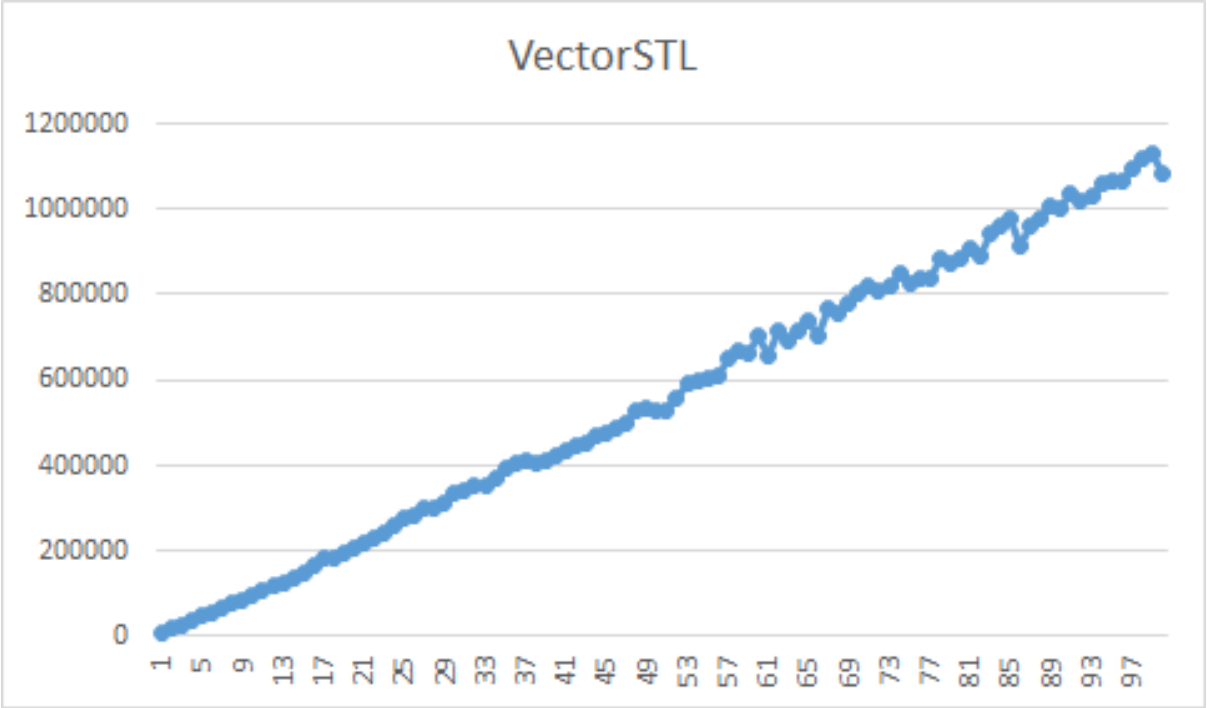
**Interpretación de los resultados:** Nuestro contenedor de datos ArbolRojoNegro es en promedio, mucho más veloz que el vector de parejas de datos de la STL, y ligeramente más veloz que el mapa de datos de la STL. A pesar de esto, todos los contenedores parecen tener un comportamiento lineal de búsqueda respecto al tamaño de elementos (es decir, O(n)).

Esto no es ideal, y el hecho de que el mapa de datos de la STL se comporta linealmente quiere decir que quizás nuestras métricas estén mal calibradas, o el tamaño muestra de este experimento no es lo suficientemente significativo para apreciar su curvatura de orden O(log2).

**Gráficos de tiempos promedios:** El tiempo promedio de los primeros 10 lotes se encuentra en el eje vertical, respecto a la iteración del experimento actual en el eje horizontal.

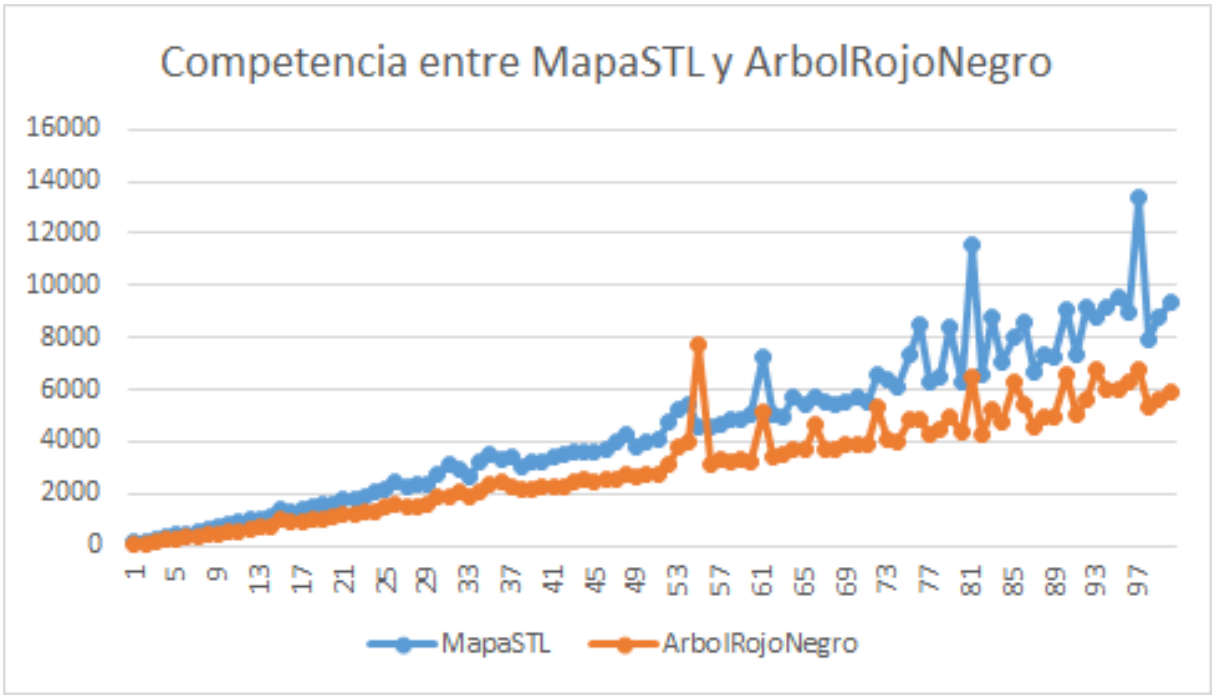


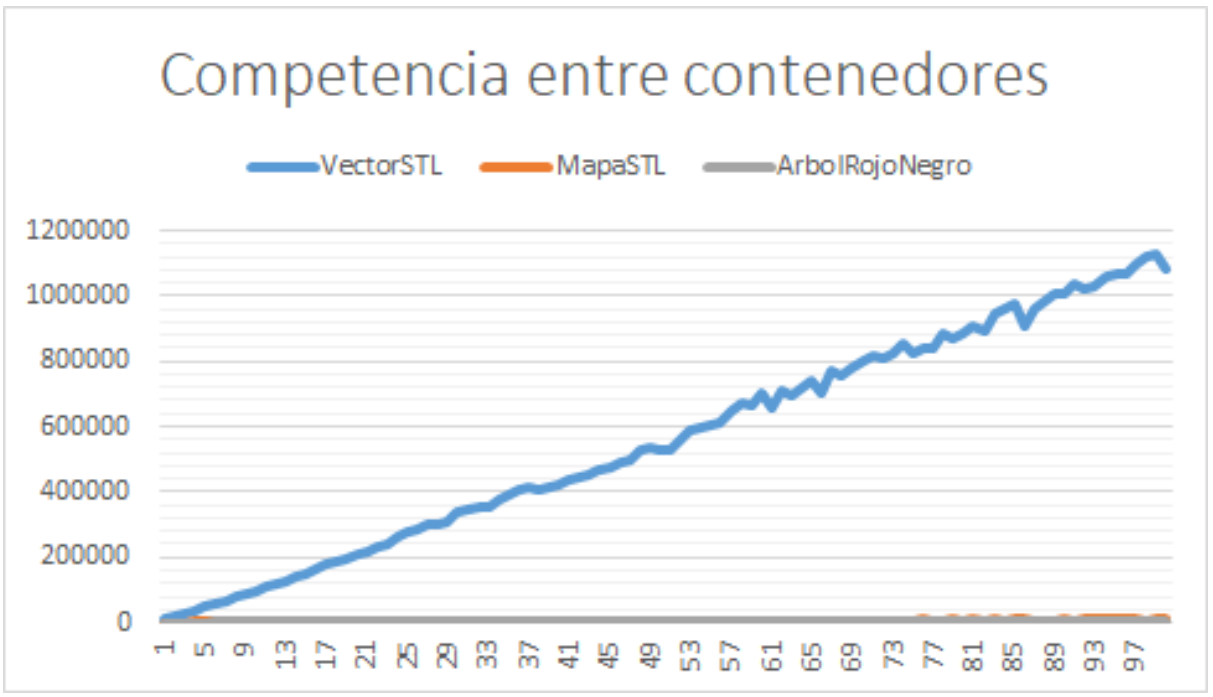


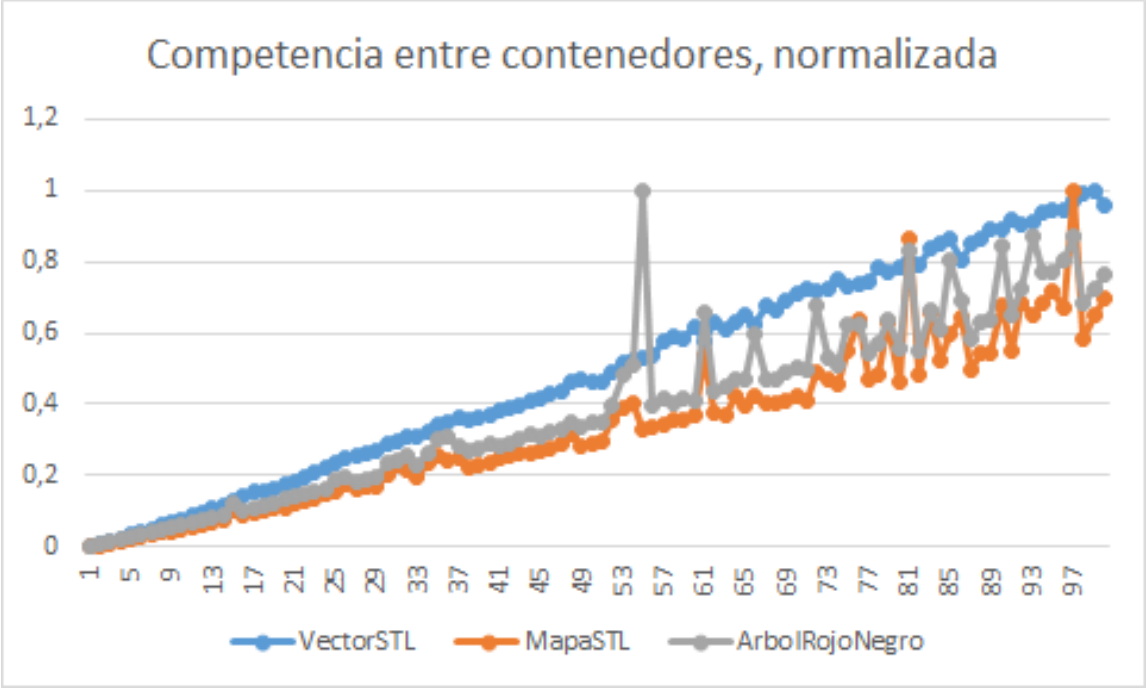




**Gráficos comparativo de tiempos promediados:** Una comparación de los tiempos en una escala absoluta para todos (primer gráfico), y una comparación de los tiempos en una escala normalizada (cada gráfico de tiempos de un contenedor fue normalizado entre 0 y 1).

Nuestro contenedor es mucho más rápido que el vector de la STL, y ligeramente más rápido que el mapa de la STL, pero sigue siendo muy lineal, con un crecimiento a razón lineal muy parecido al del mapa de la STL.




Competencia entre contenedores, normalizada
1,2
1
0,8
0,6
0,4
0,2
0
1
5
9
13
17
21
25
29
33
37
41
45
49
53
57
61
65
69
73
77
81
85
89
93
97
VectorSTL
MapaSTL
ArbolRojoNegro

**Tiempos promedios de los primeros 10 lotes, cada iteración, tabulados:** Estos son los datos del experimento, en cada contenedor (columna) y cada iteración del experimento (fila).

| *VectorSTL* | *MapSTL* | *ArbolRojoNegro* |
|---|---|---|
| 8858,7 | 119,9 | 64,5 |
| 17936,6 | 164,6 | 111,1 |
| 27151,2 | 251,4 | 179,1 |
| 37043,9 | 332,7 | 223,9 |
| 46636,4 | 411,8 | 276,7 |
| 56610,5 | 490 | 332,7 |
| 66992,1 | 577,9 | 389,4 |
| 76304,5 | 663,2 | 447,8 |
| 86831,2 | 722,7 | 474,3 |
| 96917,6 | 804,1 | 552,7 |
| 107198 | 879,9 | 576,1 |
| 117378 | 980,6 | 657,4 |
| 127943 | 1050,3 | 698,2 |
| 139350 | 1126,1 | 761,2 |
| 151510 | 1448,6 | 996,1 |
| 166011 | 1271,3 | 880,4 |
| 182464 | 1365,6 | 921 |
| 183421 | 1512,4 | 983,1 |
| 194637 | 1557,3 | 1030,5 |
| 205652 | 1605,3 | 1137,4 |
| 216652 | 1782,5 | 1182,2 |
| 228823 | 1826,4 | 1197 |
| 239599 | 1900,3 | 1273,6 |
| 260918 | 2100,9 | 1311,2 |

| | | |
|---|---|---|
| 275299 | 2203,6 | 1511,6 |
| 284955 | 2430,3 | 1563,5 |
| 298371 | 2247,2 | 1474,6 |
| 301736 | 2324,8 | 1546,6 |
| 311042 | 2403,4 | 1594,5 |
| 335656 | 2782,2 | 1897,7 |
| 342576 | 3153,22 | 1928,89 |
| 354732 | 2962,56 | 2047,11 |
| 355078 | 2695,56 | 1851,78 |
| 373033 | 3253,25 | 2101,12 |
| 391109 | 3484,75 | 2407,12 |
| 402858 | 3359,5 | 2458,38 |
| 412111 | 3405,38 | 2259,25 |
| 407066 | 3078,57 | 2159,29 |
| 412842 | 3195,14 | 2195,43 |
| 424401 | 3213,57 | 2273,14 |
| 434607 | 3418,71 | 2245,86 |
| 445880 | 3500,71 | 2286,14 |
| 453900 | 3639,5 | 2416 |
| 469962 | 3620,33 | 2525,5 |
| 477941 | 3645,33 | 2457,5 |
| 487912 | 3736,5 | 2553,17 |
| 496952 | 3946,17 | 2584,83 |
| 528098 | 4298,33 | 2758,67 |
| 532662 | 3841,5 | 2678,33 |
| 530133 | 3979,83 | 2772,83 |
| 529232 | 4073 | 2755,8 |

| | | |
|---|---|---|
| 556162 | 4796,2 | 3111,6 |
| 590584 | 5279,8 | 3778,2 |
| 596352 | 5455,8 | 3995,8 |
| 603326 | 4524,8 | 7773,4 |
| 612477 | 4548,8 | 3138,4 |
| 651236 | 4689,8 | 3284,8 |
| 670245 | 4855,2 | 3186,4 |
| 661017 | 4808,6 | 3293 |
| 701842 | 5036,8 | 3219 |
| 658054 | 7218 | 5127 |
| 713871 | 5067,25 | 3446,75 |
| 693446 | 4969 | 3529 |
| 716777 | 5731,5 | 3693,75 |
| 739589 | 5394,75 | 3664,5 |
| 703716 | 5753,25 | 4657 |
| 767881 | 5491 | 3678,5 |
| 754760 | 5466 | 3669,25 |
| 779176 | 5523 | 3848,25 |
| 804977 | 5683,75 | 3925,75 |
| 819506 | 5515,75 | 3891,25 |
| 809514 | 6616 | 5306,25 |
| 820593 | 6359,5 | 4130,5 |
| 850631 | 6146 | 3989,75 |
| 827541 | 7354 | 4859,5 |
| 835450 | 8539,67 | 4849,67 |
| 839528 | 6324,33 | 4263 |
| 883526 | 6526,33 | 4454 |

| | | |
|---|---|---|
| 871818 | 8421,67 | 4959,67 |
| 886257 | 6287,33 | 4347 |
| 910555 | 11581,3 | 6488 |
| 890646 | 6546,67 | 4295 |
| 943465 | 8791,67 | 5191,67 |
| 960364 | 7036 | 4791,33 |
| 975050 | 8029,33 | 6247 |
| 910765 | 8591 | 5387,67 |
| 962849 | 6688,67 | 4547 |
| 979658 | 7298 | 4918 |
| 1009050 | 7295 | 4987,33 |
| 1002570 | 9077,33 | 6587,67 |
| 1035420 | 7371,67 | 5069 |
| 1019260 | 9130,33 | 5661 |
| 1028950 | 8744,33 | 6759,67 |
| 1057440 | 9199,33 | 5993 |
| 1063610 | 9572,67 | 5990,33 |
| 1065730 | 8964,33 | 6262,33 |
| 1095070 | 13347,3 | 6766,33 |
| 1120070 | 7881 | 5311,67 |
| 1128260 | 8763,67 | 5622,67 |
| 1080530 | 9378,67 | 5952 |

Tras esta multitud de pruebas diferentes, pudimos graficar la progresión del algoritmo sobre 10000 palabras. Esto permite apreciar a una pequeña escala (incierto si es significativa o no) la progresión de las búsquedas en los contenedores.

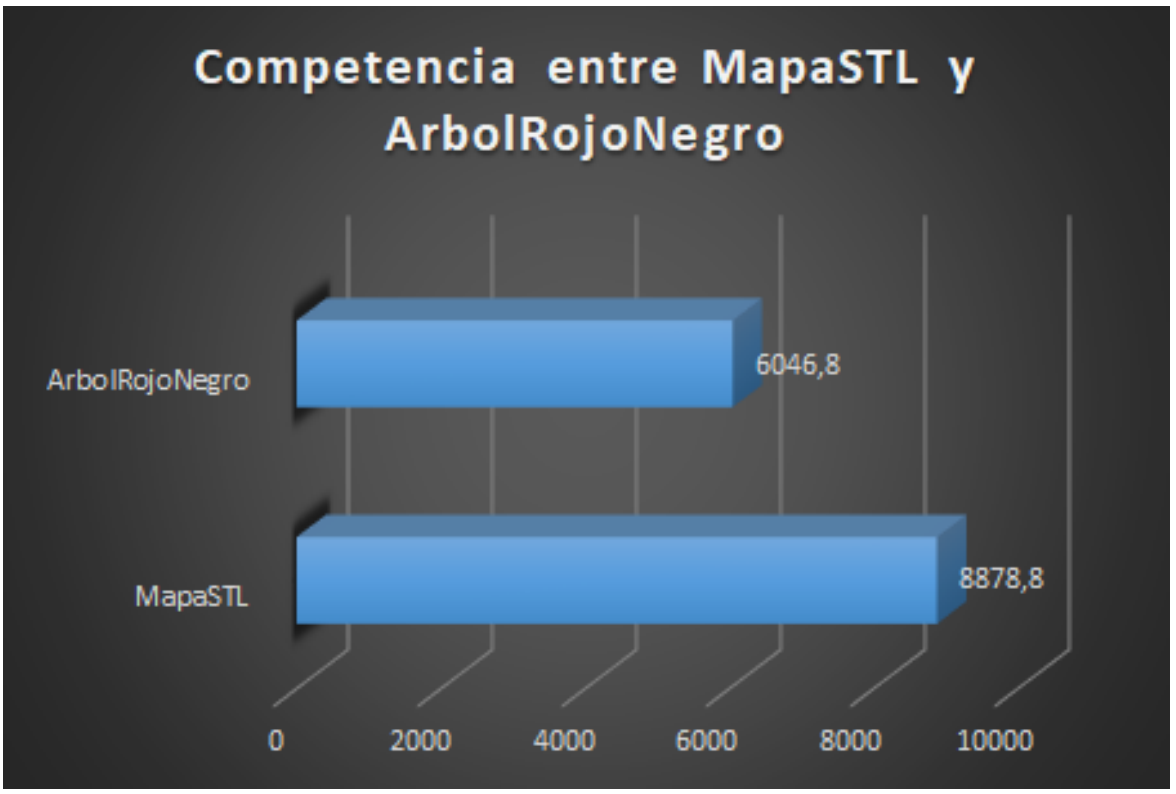
Sería más útil estudiar la progresión de tiempos promedios a mayor escala. Sin embargo, los medios con los cuales contamos para ejecución del proyecto no son los suficientemente potentes (toma demasiado tiempo) y no tiene el nivel de procesamiento adecuado para poder evaluar la cantidad de casos que se presentan en el enunciado (en tiempo para la entrega).

.

**ENSAYO DEL EXPERIMENTO A1**

| **PARÁMETROS DE EXPERIMENTO** | | |
|---|---|---|
| *Nombre del parámetro* | *Valor ingresado* | *Breve descripción* |
| cantidadElementosLectura | 150000 | Estudiamos 50000 palabras para compensar las palabras repetidas que no serán introducidas en los contenedores. |
| pasoPruebaInicial | 10000 | Por indicación del enunciado. |
| cantidadPruebas | 1 | Por indicación del enunciado. |
| nombreArchivo | datos.txt | Para tabular los resultados. |

**Interpretación de los resultados:** Nuestro contenedor ArbolRojoNegro es, en promedio, más rápido que el mapa de datos de la STL y el vector de datos de la STL.





**Tiempos promedios de los primeros 10 lotes, en una sola iteración, tabulados:**

| *VectorSTL* | *MapaSTL* | *ArbolRojoNegro* |
|---|---|---|
| 3100380 | 8878,8 | 6046,8 |

Realizar el experimento (A2) resulta en la práctica imposible para el tiempo de entrega, ya que nuestro poder de procesamiento (y velocidad de ejecución) lo vuelven insostenible.