Tarea 3 - Diccionarios

CC4102 - Diseño y Análisis de Algoritmos

Alumnos : Sebastián González
Patricio Isbej
Profesor : Gonzalo Navarro
Auxiliar : Jorge Bahamonde
Fecha : 16 de diciembre de 2015

Resumen

En este trabajo se analizó 4 estructuras distintas para diccionarios, implementandolas, ejecutandolas para distintos casos y comparando para uno su rendimiento ante inserciones, consultas y eliminaciones.

1. Introducción

Para este trabajo se compararon los siguientes tipos de diccionarios

- Árbol de Busqueda Binaria convencional
- Árbol AVL

Árbol de Busqueda Binaria convencional consiste en el árbol donde los elementos son agregados y eliminados a medida que se pide, sin realizar ningún tipo de modificación u optimización cuando se hacen las operaciones.

Árbol AVL revisa al momento de insertar o eliminar elementos si los hijos de cada nodo involucrado difieren en sus tamaños por más de 1, en caso de ocurrir, realiza un rebalanceo de manera que se cumpla la condición

Para cada estructura se realizaron los siguientes tests

- 2^4 a 2^{14} inserciones
- 2^4 a 2^{15} búsquedas para las estructuras de mayor tamaño
- 2^4 a 2^{15} borrados

2. Hipótesis

2.1. Árbol de Busqueda Binaria convencional

Se espera que para cantidades pequeñas de datos el tiempo de creación de esta estructura sea rápida, al no realizar rebalanceos ni cambios extras al realizar operaciones.

A medida que aumente la cantidad de elementos, se espera que el tiempo de búsqueda e inserción tenga forma logarítmica.

2.2. Árbol AVL

Como este árbol realiza re-balanceos, se espera que el tiempo de creación tome más tiempo que el convencional.

Se espera que para una mayor cantidad de elementos, se comporte mejor que el anterior para todas las operaciones, al encontrarse más comprimido debido a los re-balanceos realizados (va a tener menor altura que el anterior, lo que significa un mejor peor caso).

3. Diseño e implementación

Se desarrollaron todas las estructuras de datos en C. Esta decisión está dada por familiaridad con el lenguaje y por la facilidad para el cálculo del tamaño total de las estructuras. Los árboles estudiados fueron implementados como estructuras, siguiendo una misma firma para el llamado a las operaciones estudiadas. De esta forma todos los métodos pueden seguir la misma bateria de tests.

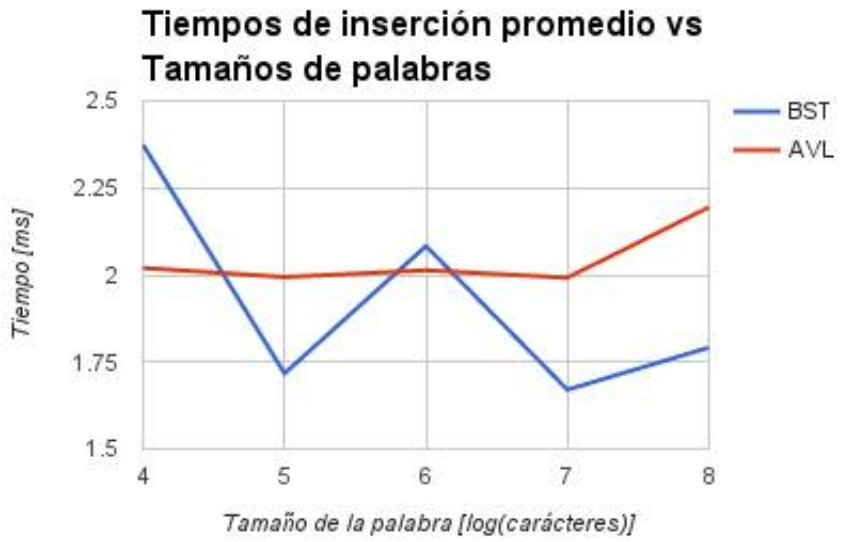
Para el manejo de strings, se trabajó con una tabla de dos entradas. La primera tabla es un único arreglo de carácteres que contiene todo el universo estudiado. La segunda tabla es un arreglo que indica el índice en el cuál inicia cada string. De esta forma los nodos sólo hacen referencia a los strings almacenados. Con este método nos aseguramos que las operaciones estudiadas no consideren el tiempo de alocación o desecho de memoria reservada para strings. A pesar de esto, el tamaño calculado de los nodos si considera el tamaño del string almacenado. Esta decisión fue tomada para facilitar la alocación y desecho de memoria para los strings.

Con el fin de generar los universos de datos, se trabajó con dos script auxiliares en Phyton. El primer script genera archivos de texto con listas de palabras aleatorias de tamaño uniforme. Las palabras generadas tienen un tamaño de 2^4 a 2^8 carácteres yel archivo consta de 2^{15} palabras.

El segundo script toma un archivo de texto plano cualquiera y genera un archivo *pto* que es utilizable por la tarea. Un archivo pto es un archivo binario que contiene la tabla de strings y la tabla de índices, además de algunos datos para facilitar la lectura del archivo mismo. El texto de input es filtrado, eliminando carácateres extraños y el archivo pto es generado.

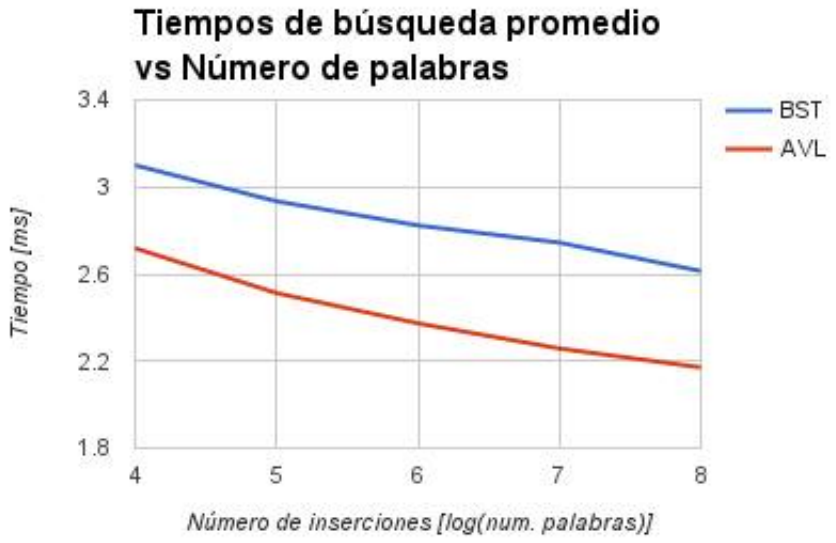
4. Experimentos y resultados

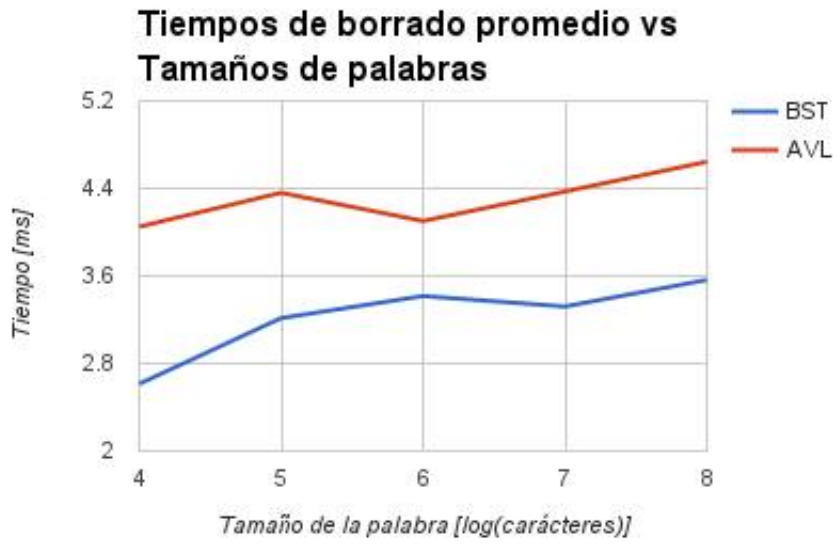
Se ejecutaron las pruebas mencionadas, obteniendo para cada operación el tiempo que tarda por tamaño de las palabras y por número de palabras, cuyos resultados se pueden observar en los siguientes gráficos

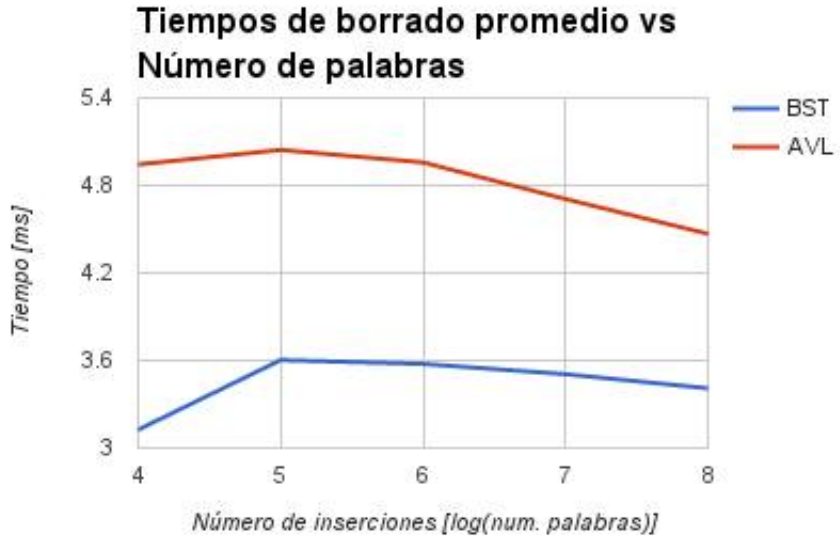

Tiempos de inserción promedio vs Tamaños de palabras
2.5
2.25
2
1.75
1.5
Tiempo [ms]
4
5
6
7
8
Tamaño de la palabra [log(carácteres)]
BST
AVL


Tiempos de insertado promedio vs Número de palabras
1.96
1.82
1.68
1.54
1.4
Tiempo [ms]
4
5
6
7
8
Número de inserciones [log(num. palabras)]
BST
AVL


Tiempos de búsqueda promedio
vs Tamaños de palabras
3
2.7
2.4
2.1
1.8
Tiempo [ms]
4
5
6
7
8
Tamaño de la palabra [log(carácteres)]
BST
AVL


Tiempos de búsqueda promedio
vs Número de palabras
3.4
3
2.6
2.2
1.8
Tiempo [ms]
4
5
6
7
8
Número de inserciones [log(num. palabras)]
BST
AVL


Tiempos de borrado promedio vs
Tamaños de palabras
5.2
4.4
3.6
2.8
2
Tiempo [ms]
4
5
6
7
8
Tamaño de la palabra [log(carácteres)]
BST
AVL


Tiempos de borrado promedio vs
Número de palabras
5.4
4.8
4.2
3.6
3
Tiempo [ms]
4
5
6
7
8
Número de inserciones [log(num. palabras)]
BST
AVL

Word Length [log(char)]	BST Size[Bytes]	BST INSRT Time[ms]	BST SRCH Time[ms]	BST DEL Time[m
4	321565.0909	2.373696091	2.202165	2.613647727
5	512122.1818	1.717207455	2.730613091	3.215207182
6	893236.3636	2.083007818	2.675012727	3.416753727
7	1655464.727	1.670363091	2.727791636	3.320845182
8	3179921.455	1.791365182	2.831529	3.564231182

Number of Words [log(N)]	BST Size[Bytes]	BST INSRT Time[ms]	BST SRCH Time[ms]	BST DEL Time
4	7052.8	1.625	3.1	3.125
5	14105.6	1.61875	2.934375	3.603125
6	28211.2	1.653125	2.8234372	3.5765624
7	56422.4	1.7382812	2.7445314	3.50625
8	112844.8	1.829297	2.6136718	3.4093752

5. Análisis y Conclusiones

Tiempos Promedio vs Largo de palabras
Tiempos Promedio vs Número de palabras
Tiempos Promedio de ejecución y Tamaño de estructura en detalle

Word Length [log(char)]	Number of Words [log(N)]	BST Size[Bytes]	BST INSRT Time[ms]	BST SRCH T
4	4	1728	2.5625	2.34375
4	5	3456	2.453125	2.25
4	6	6912	2.4375	2.445312
4	7	13824	2.621094	2.480469
4	8	27648	2.671875	2.34375
4	9	55296	2.530273	2.189453
4	10	110592	2.213379	2.081055
4	11	221184	2.077637	1.999756
4	12	442368	2.150757	2.014282
4	13	884736	2.163757	2.017883
4	14	1769472	2.22876	2.058105
5	4	2752	1.3125	2.9375
5	5	5504	1.296875	2.765625
5	6	11008	1.359375	2.703125
5	7	22016	1.441406	2.699219
5	8	44032	1.535156	2.691406
5	9	88064	1.625977	2.607422
5	10	176128	1.717285	2.992676
5	11	352256	1.874756	2.992188
5	12	704512	2.033325	2.705811
5	13	1409024	2.205994	2.519836
5	14	2818048	2.486633	2.421936
6	4	4800	1.78125	3.34375
6	5	9600	1.734375	3.140625
6	6	19200	1.75	2.945312
6	7	38400	1.796875	2.75
6	8	76800	1.947266	2.574219
6	9	153600	2.05957	2.482422
6	10	307200	2.162598	2.483398
6	11	614400	2.438721	2.505127
6	12	1228800	2.283325	2.423096
6	13	2457600	2.372253	2.378662
6	14	4915200	2.586853	2.398529
7	4	8896	1.15625	3.375
7	5	17792	1.25	3.203125
7	6	35584	1.3125	2.984375
7	7	71168	1.375	2.902344
7	8	142336	1.460938	2.701172
7	9	284672	1.552734	2.554688
7	10	569344	1.714844	2.51123
7	11	1138688	1.794678	2.4646
7	12	2277376	1.96875	2.481934
7	13	4554752	2.200989	2.429565
7	14	9109504	2.587311	2.397675
8	4	17088	1.3125	3.5
8	5	34176	1.359375	3.3125
8	6	68352	1.40625	3.039062
8	7	136704	1.457031	2.890625
8	8	273408	1.53125	2.757812