Proyecto 2: Map

Estructuras de Datos, Primer Semestre 2019

Prof. Diego Seco

Ayudantes: Diego Gatica y Alexander Irribarra

1. Descripción del Problema

En muchas aplicaciones software es necesario almacenar una colección de pares clave-valor, siendo las implementaciones del ADT Map las soluciones más frecuentemente empleadas para ello. En este proyecto se desarrollarán y evaluarán experimentalmente algunas de dichas implementaciones.

2. Implementación

Se implementará un Tipo Abstracto de Datos, que llamaremos Map, proporcionando las siguientes operaciones: insert(pair<string,int>), at(string), erase(string), empty(), size(). El comportamiento esperado de dichas operaciones es el mismo que en el Laboratorio 7.

Se implementarán tres estructuras de datos para Map:

1. Basada en un arreglo ordenado (se llamará *MapSV*). Se debe emplear la clase *vector* de la STL. La operación at se debe implementar con búsqueda binaria. La operación insert debe insertar de forma que garantice que los elementos del vector se encuentren ordenados.

2. Basada en Hashing (nombre *MapH*). Se empleará la mejor implementación de las realizadas en el Laboratorio 7.

3. Basada en AVL (nombre *MapAVL*). La implementación se basará en la descripción del AVL realizada en las clases de teoría.

El informe debe contener un análisis de peor caso de las operaciones insert, at y erase de todas las implementaciones. Nota: los análisis deben considerar que los strings que se utilizan como clave tienen una longitud máxima de m y que el número máximo de claves es n. Pista: la complejidad de la operación at en esta implementación del AVL NO es $O(\log n)$ ya que no está considerando el coste de comparar las claves entre ellas.

Además, se compararán experimentalmente todas las implementaciones de insert y at. Es decir, el informe debe incluir varias gráficas: una para la operación insert y 2 para la operación at; una de ellas para búsquedas de elementos que retornan resultados y otra para búsquedas que no retornan resultados. En todas ellas el eje X representará número de elementos y el eje Y representará el tiempo promedio de la operación. Utilice la siguiente manera de medir tiempo:

```
#include < time.h >
...
```

```
int main()
{
  clock_t start, stop;

  start = clock();
  // Realizar X operaciones
  stop = clock();

 cout << (stop-start)/CLOCKS_PER_SEC/X << endl;

}
```

Es importante observar que se mide el tiempo de X operaciones (nunca de una sola) y el tiempo obtenido se divide entre el número de operaciones realizadas.

Durante la etapa de implementación se utilizarán conjuntos de prueba pequeños. Sin embargo, para presentar la experimentación los datasets deben ser lo suficientemente grandes como para mostrar las diferencias en la implementación.

En el informe también se debe incorporar una discusión acerca de los resultados obtenidos. La discusión debe incluir un párrafo analizando las implicaciones que tiene el que las claves sean strings con respecto a que sean enteros (en particular, por qué el que las claves sean strings perjudica más al AVL).

3. Evaluación

El proyecto se realizará en parejas. Enviar en piazza (mensaje privado a *Instructors*) lo siguiente:

1. Un informe que:
   a) Incluya portada, descripción de la tarea, descripción de las soluciones propuestas, detalles de implementación, análisis teórico y análisis experimental.
   b) Sea claro y esté bien escrito. Un informe difícil de entender es un informe que será mal evaluado aunque todo esté bien implementado. La persona que revise el documento debe poder entender su solución sólo mirando el informe.
   c) Esté en formato pdf.
2. Un archivo comprimido con todos los ficheros fuente implementados para solucionar la tarea. El informe debe hacer referencia a ellos y explicar en qué consiste cada uno.

Fecha de Entrega: viernes 21 de junio 11:59PM