

Eficiencia `gs1Set`, `set` y `unordered_set`

La medición de tiempos que se ha usado es:

```
clock_t start, end;
```

```
...
```

```
start = clock();  
//llamada a la funcion de insercion/busqueda/borrado  
end= clock();  
double dif = end-start;  
cout<< dif/(double) (CLOCKS_PER_SEC * 1000.0) << endl;
```

Inserción

Para insertar elementos hemos utilizado:

```
template <typename T>  
void load(T & contenedor, const string & s) {  
    ifstream fe;  
    string cadena;  
    cout << "Abrimos " << s << endl;  
    fe.open(s.c_str(), ifstream::in);  
    if (fe.fail())  
    {  
        cerr << "Error al abrir el fichero " << s << endl;  
    } else {  
  
        while ( !fe.eof() )  
        { getline(fe,cadena,'\n');  
            if (!fe.eof()) {  
                // cout << "leo:: " << cadena << endl;  
  
                contenedor.insert(cadena);  
            }  
        }  
  
    } // else  
    fe.close();  
}
```

Resultados obtenidos:

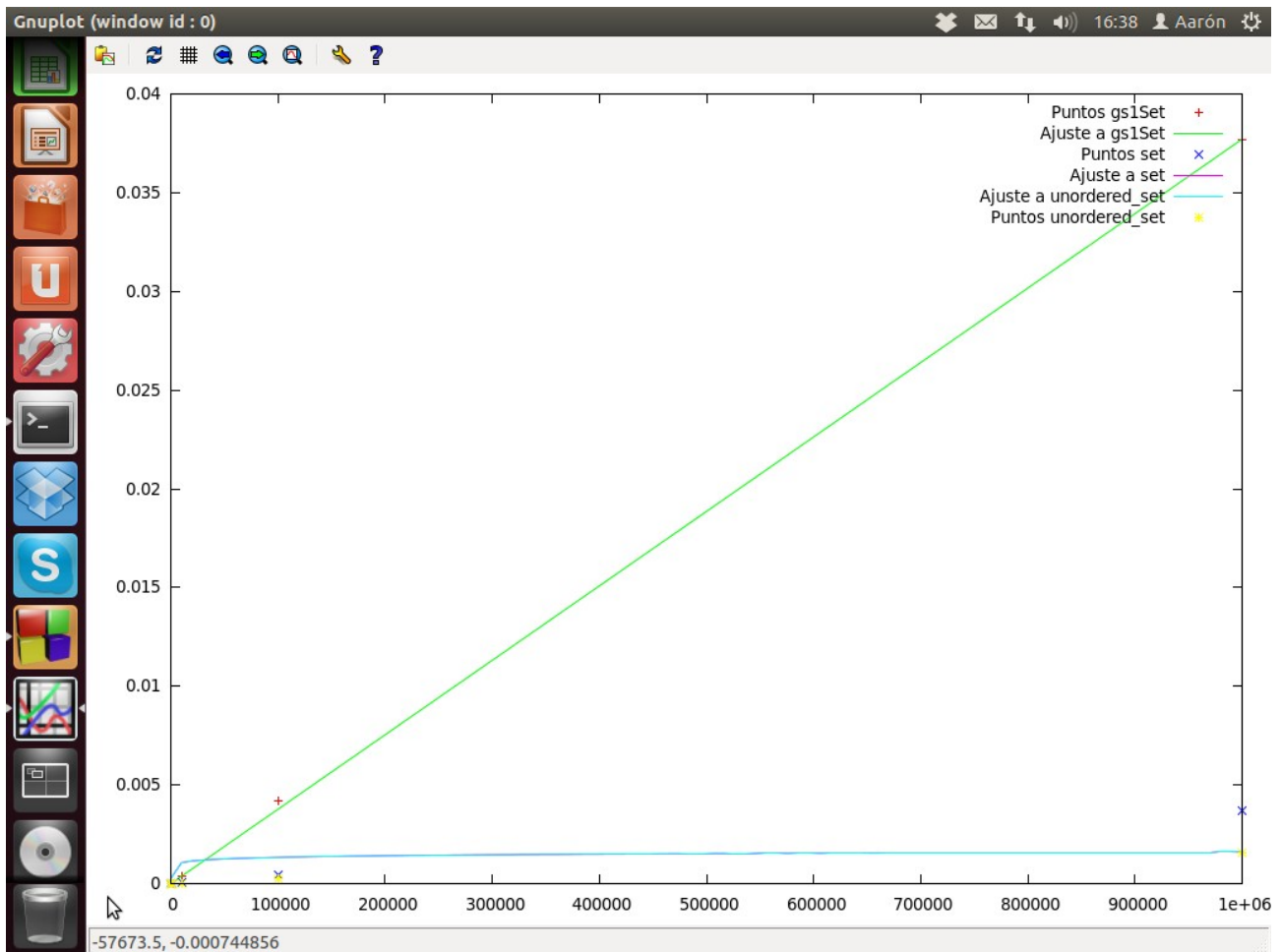
Num. elementos	<code>gs1Set</code>	<code>set</code>	<code>unordered_set</code>
10	0	0	0
100	0	0	0
1000	3e-05	0	0
10000	0,00038	4e-05	2e-05

Aarón Rodríguez Bueno
Bryan Moreno Picamán

100000	0,00417	0,00044	0,00027
1000000	0,03767	0,00367	0,00155

Eficiencia inserción gs1Set: $O(n)$
Eficiencia inserción set: $O(\log(n))$
Eficiencia inserción unordered_set: $O(\log(n))$

Gráfico eficiencia:



Búsqueda

Para buscar elementos hemos utilizado:

```
//Una vez insertado el árbol correspondiente
template <typename T>
void busc(T & contenedor, const string & s) {
    ifstream fe;
    string cadena;
    //cout << "Abrimos " << s << endl;
    fe.open(s.c_str(), ifstream::in);
    if (fe.fail())
    {
```

Aarón Rodríguez Bueno
Bryan Moreno Picamán

```
cerr << "Error al abrir el fichero " << s << endl;
} else {

    while ( !fe.eof() )
    { getline(fe,cadena,'\n');
      if (!fe.eof()) {
        // cout << "leo:: " << cadena << endl;

        contenedor.find(cadena);
      }
    }

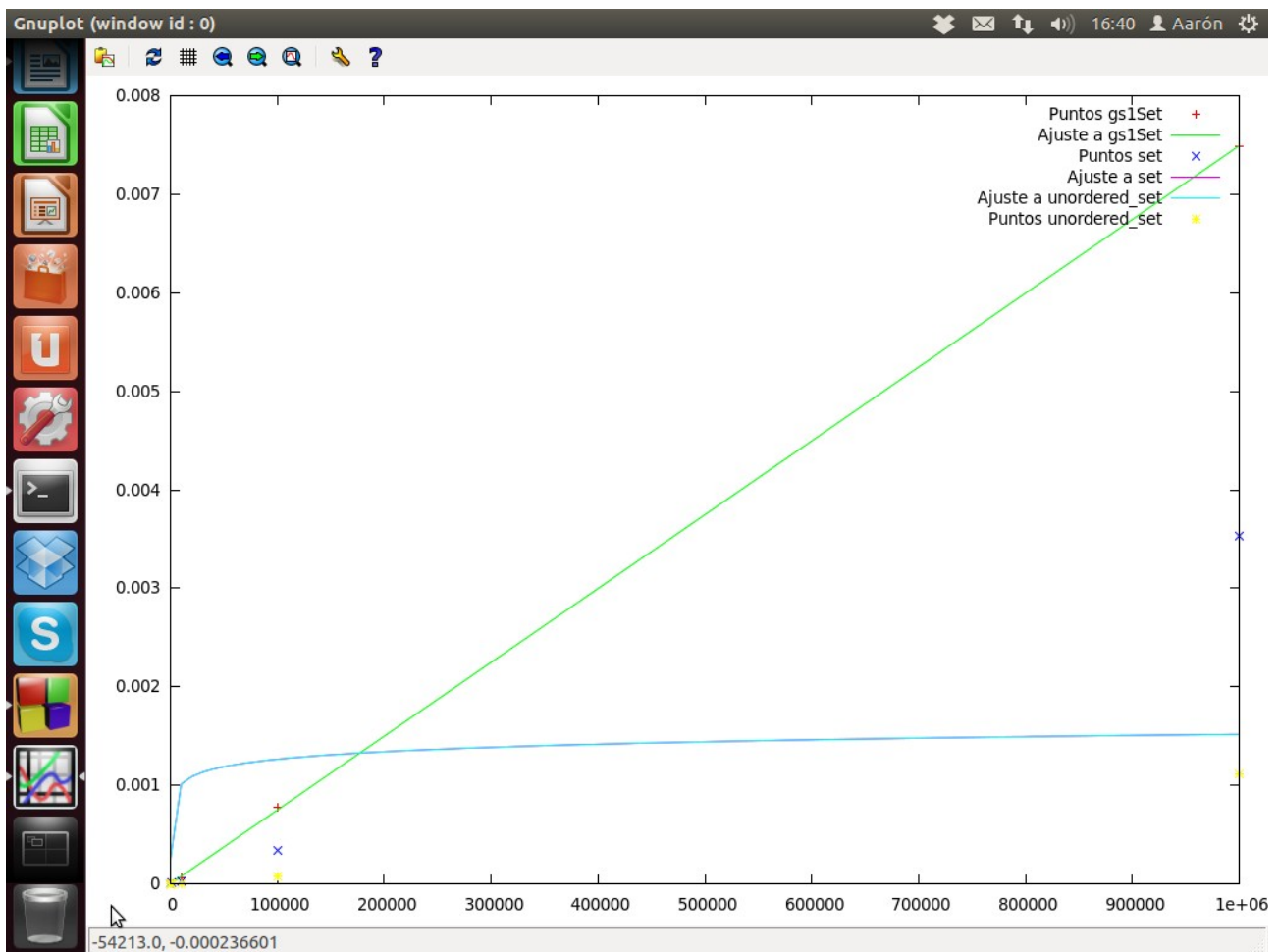
    } // else
fe.close();
}
```

Resultados obtenidos:

Num. elementos	gs1Set	set	unordered_set
10	0	0	0
100	0	0	0
1000	0	1e-05	0
10000	7e-05	3e-05	0
100000	0.00078	0.00034	8e-05
1000000	0.00749	0.00353	0.00112

Eficiencia inserción gs1Set: $O(n)$
Eficiencia inserción set: $O(\log(n))$
Eficiencia inserción unordered_set: $O(\log(n))$

Gráfica eficiencia:



Borrado

Para eliminar elementos hemos utilizado:

```
//Una vez insertado el árbol correspondiente
template <typename T>
void borrar(T & contenedor, const string & s) {
    ifstream fe;
    string cadena;
    //cout << "Abrimos " << s << endl;
    fe.open(s.c_str(), ifstream::in);
    if (fe.fail())
    {
        cerr << "Error al abrir el fichero " << s << endl;
    } else {
        while ( !fe.eof() )
        {
            getline(fe,cadena,'\n');
            if (!fe.eof()) {
                // cout << "leo:: " << cadena << endl;

                contenedor.erase(cadena);
            }
        }
    }
}
```

Aarón Rodríguez Bueno
Bryan Moreno Picamán

```
}  
  
} // else  
fe.close();  
}
```

Resultados obtenidos:

Num. elementos	gs1Set	set	unordered_set
10	0	0	0
100	0	0	0
1000	1e-05	0	0
10000	8e-05	3e-05	2e-05
100000	0.0008	0.00047	0,00013
1000000	0.00779	0.00088	0.00058

Eficiencia inserción gs1Set: $O(n)$

Eficiencia inserción set: $O(\log(\text{size}()) + \text{count}(k))$

Eficiencia inserción unordered_set: $O(\log(\text{size}()) + \text{count}(k))$

No sabemos hacer en gnuplot $O(\log(\text{size}()) + \text{count}(k))$, así que dejamos la línea con $f(x) = a \cdot \log(n)$, por lo que diferirá mucho de los puntos.

Gráfica eficiencia:

Aarón Rodríguez Bueno
Bryan Moreno Picamán

