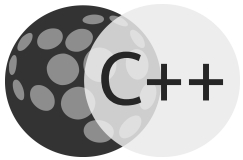
Mira esa secuencia... ¿Es un vector? ¿Es una lista?

¡No! ¡¡Es un Súper Árbol!!

Martín Knoblauch Revuelta
http://www.mkrevuelta.com @mkrevuelta mkrevuelta@gmail.com

indizen  using std::cpp

Except where otherwise noted, this work is licensed under:
http://creativecommons.org/licenses/by-nc-sa/4.0/

Universidad Carlos III de Madrid,30 de noviembre de 2017

Presentación disponible en mi blog semiabandonado:
http://www.mkrevuelta.com

Índice

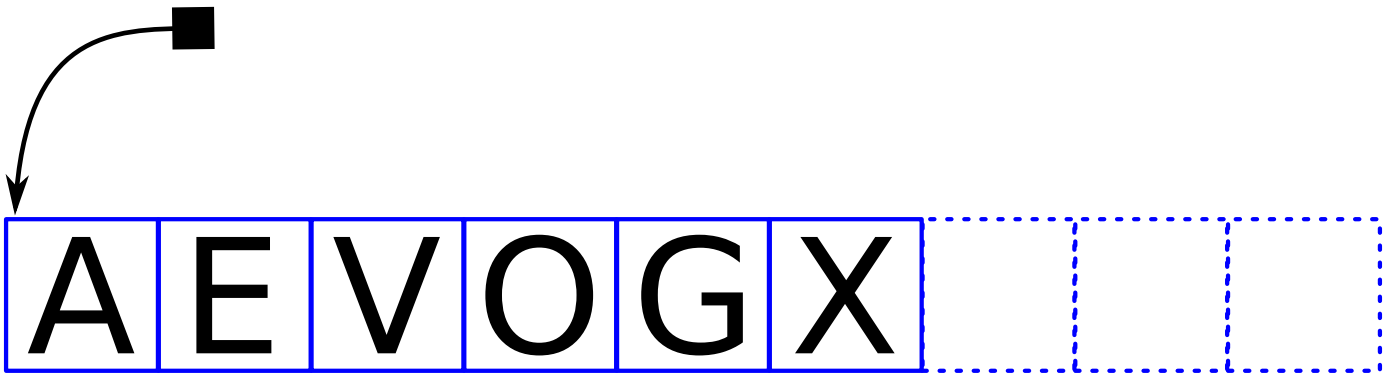
Introducción al problema

Are lists evil?–Bjarne Stroustrup

https://isocpp.org/blog/2014/06/stroustrup-lists

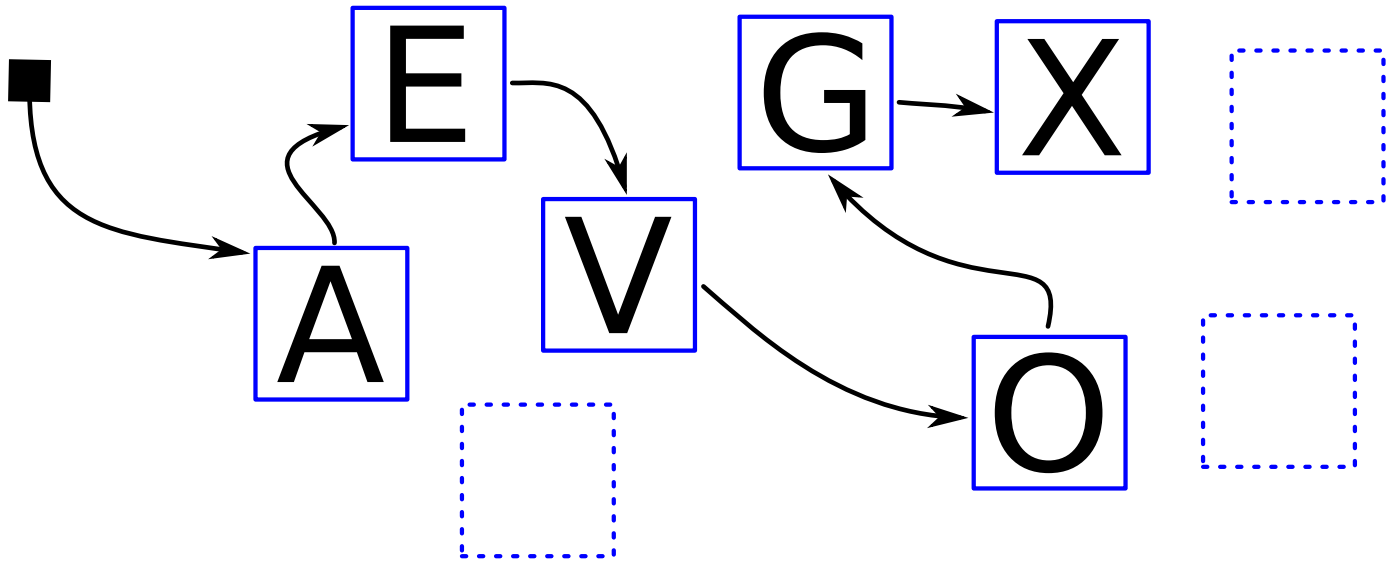
Array

- Acceso aleatorio rápido
- Inserción/extracción... lentas

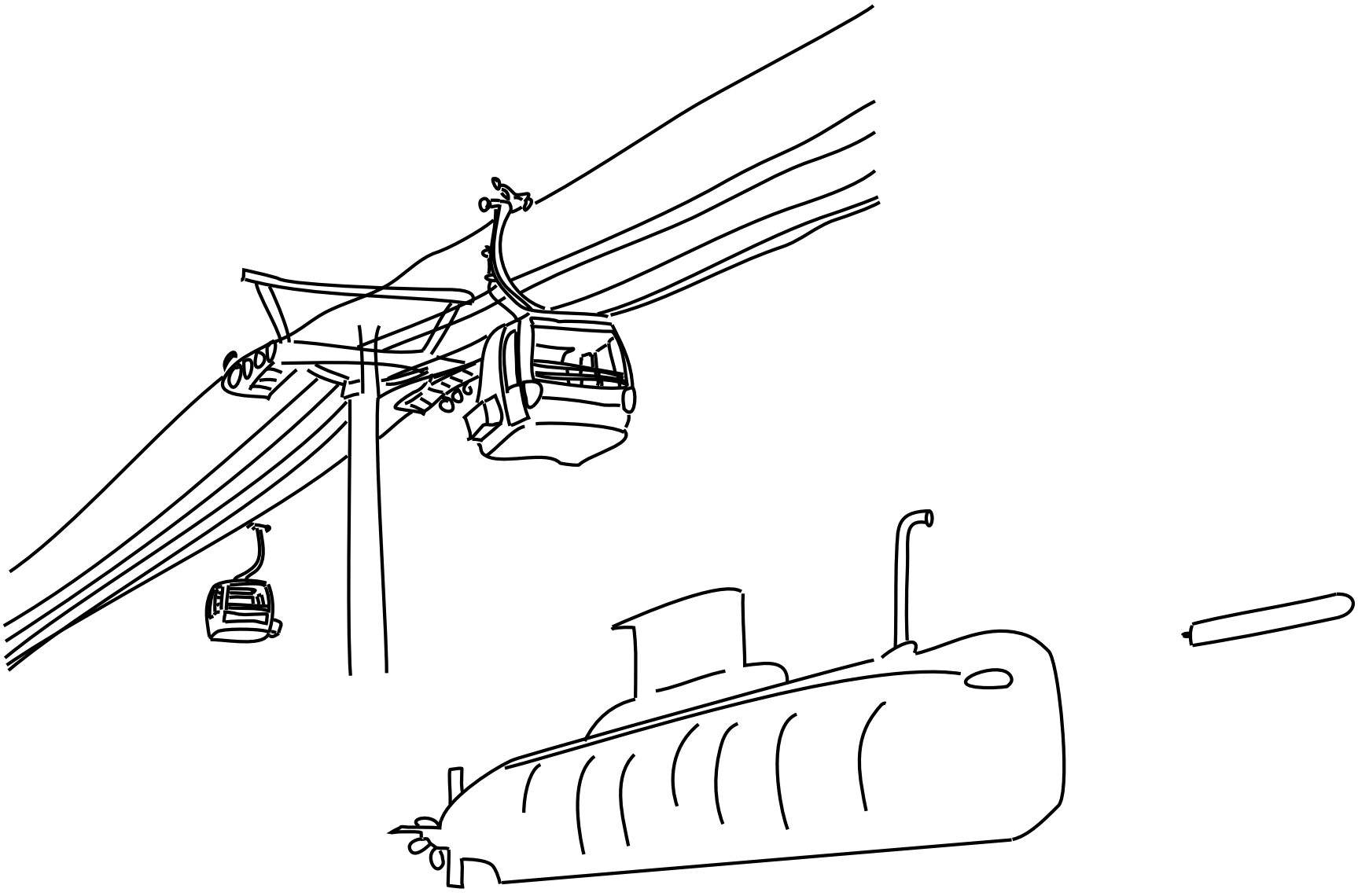


Lista enlazada

- Inserción/extracción rápidas
- Acceso aleatorio... leeeento



¿Cómo compararlos?



Sugerencia de Jon Bentley

```
for (;;)
{
```

Acceso aleatorio

Inserción / extracción

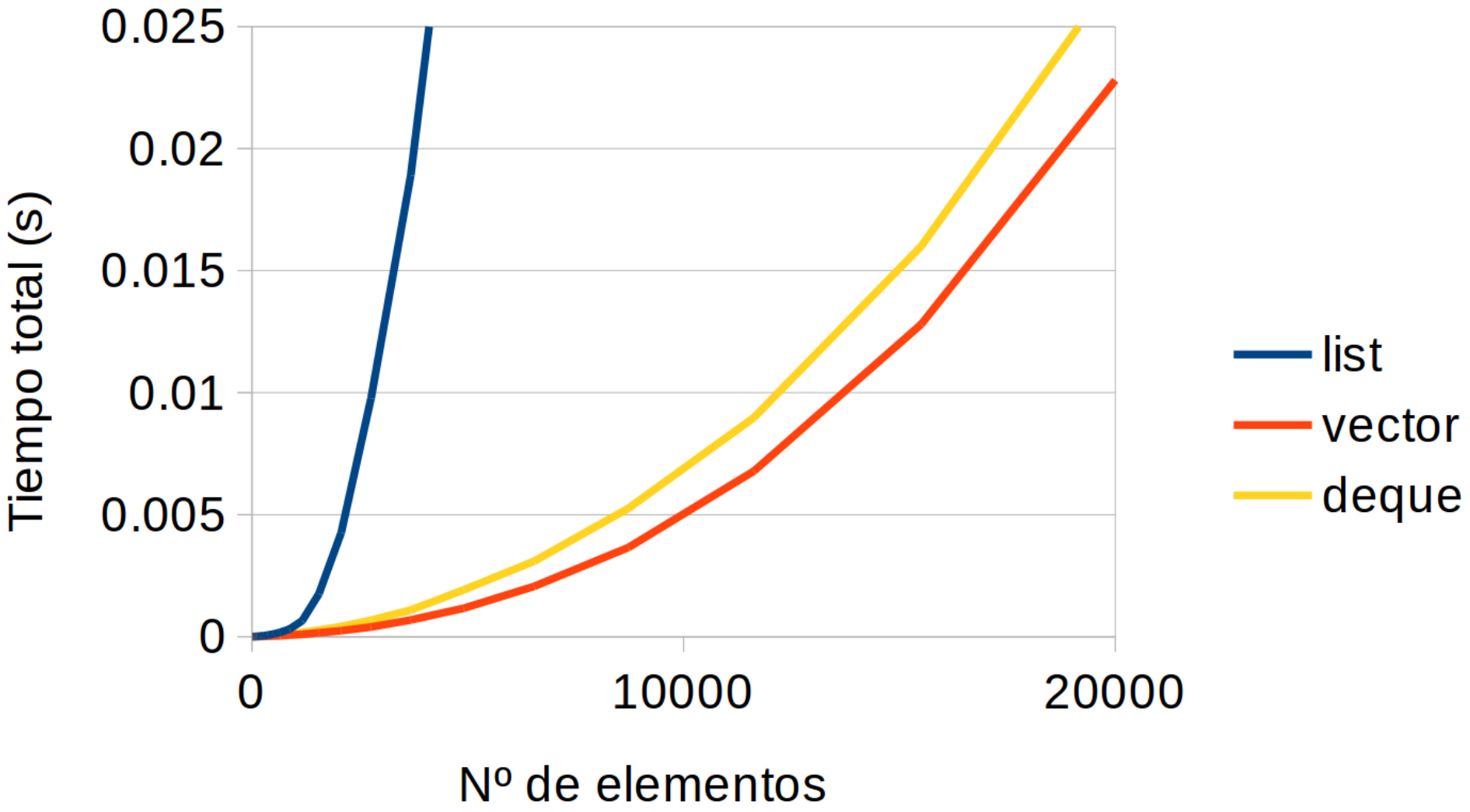
```
}
```

Sugerencia de Jon Bentley

"Insert a sequence of random integers
into a sorted sequence,

then **remove** those elements **one by one**
as determined by
a **random** sequece of **positions**"

Resultado


0.025
0.02
0.015
0.01
0.005
0
Tiempo total (s)
0
10000
20000
Nº de elementos
list
vector
deque

Conclusión

El vector es más rápido
en cierta proporción fija
(una proporción considerable)

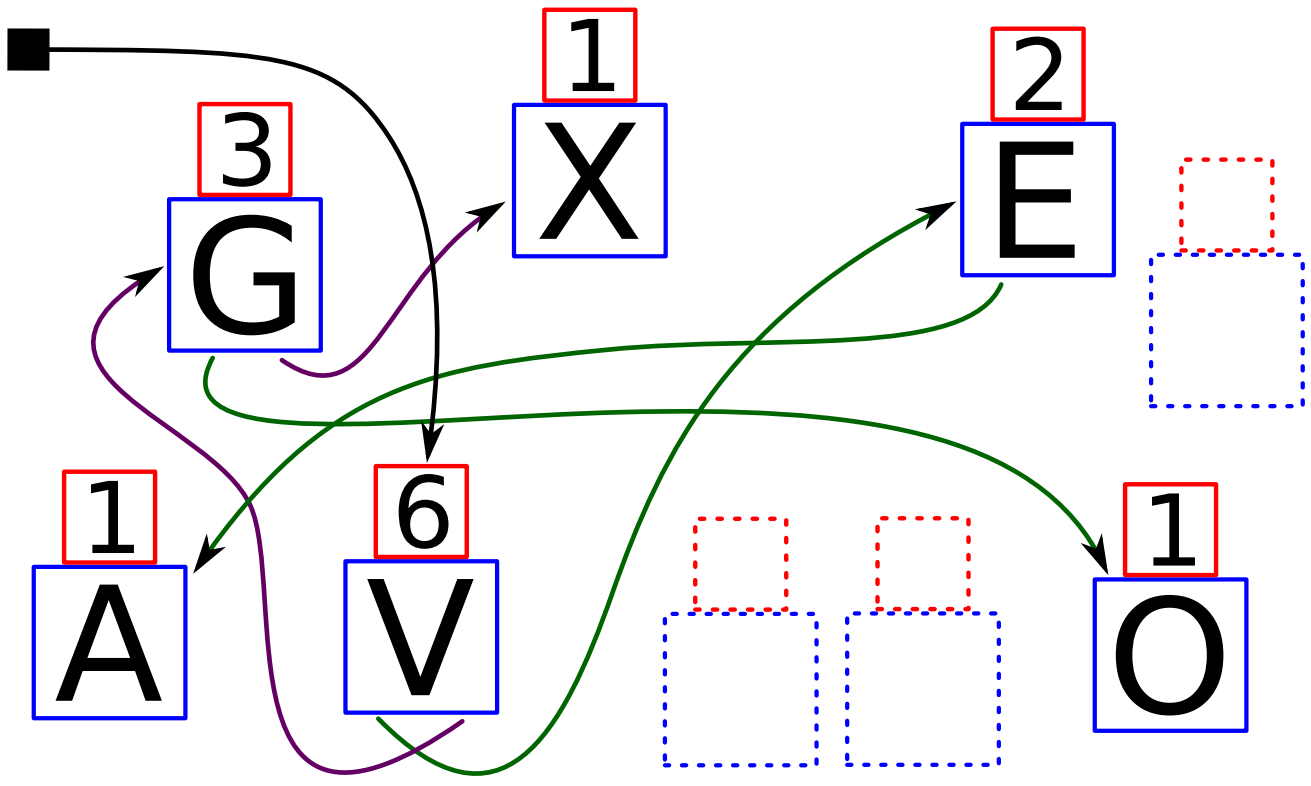
Pero...

¿Nos interesa realmente
el problema de Jon Bentley?

Super Árbol

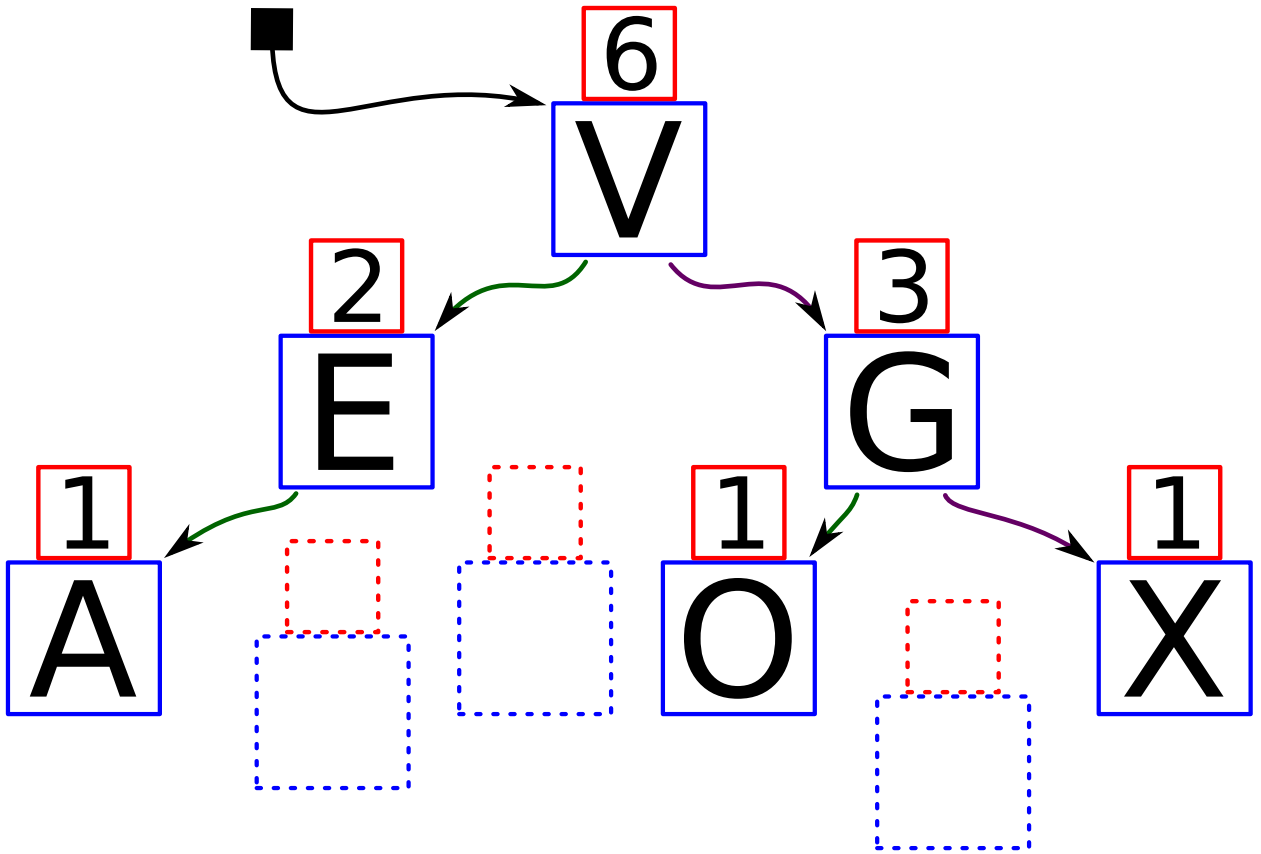
Árbol aumentado (enredado)

Como una lista, pero con dos “siguientes” (izq., y der.)



Árbol aumentado

Metadato especial: número de nodos del sub-árbol



Acceso aleatorio (1/3)

```
template <typename T>
struct node
{
  node<T> * left;       // Sub-árbol izq.
  node<T> * right;      //  "     "    der.
  std::size_t count;    // Núm. nodos
  T value;              // Carga útil
};
```

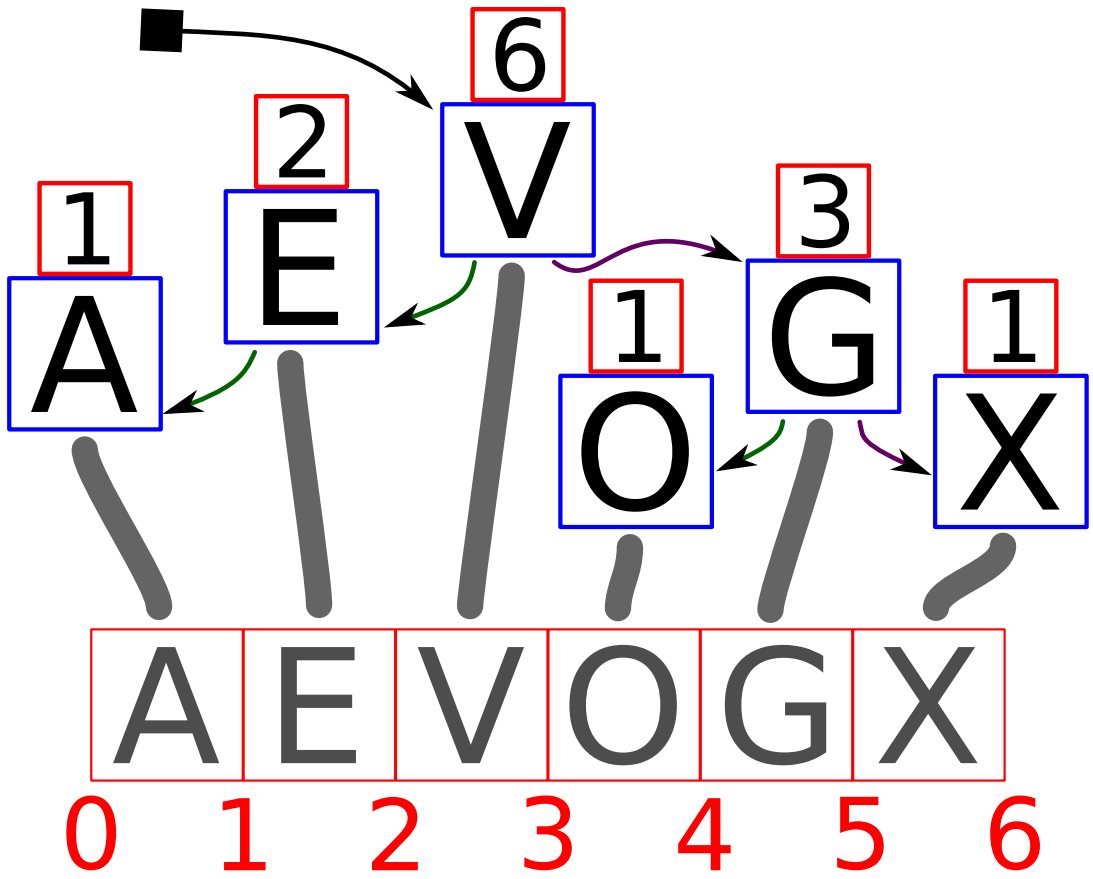
Acceso aleatorio (2/3)

```
template <typename T>
node<T> * RandomAccess (node<T> * root,
                        std::size_t pos)
{
  if (pos >= root->count)
    return nullptr;

  node<T> * p = root;
```

Acceso aleatorio (3/3)

```
for (;;)
{
  std::size_t nLeft = p->left ?
                      p->left->count : 0;

  if (pos == nLeft)          return p;
  else if (pos < nLeft)      p = p->left;
  else // (pos > nLeft)
  {
    pos -= nLeft + 1;
    p = p->right;
  }                                  } } // fin
```

Vista proporcional



Complejidad computacional

	Acceso aleatorio	Inserción/ Extracción	**Suma de ambas**
Array	$O(1)$	$O(N)$	$O(N)$
Lista	$O(N)$	$O(1)$	$O(N)$
Súper Árbol	$O(log(N))$	$O(log(N))$	$O(log(N))$

Complejidad computacional (leyenda)

- $O(1)$ = constante 
- $O(log(N))$ = logarítmica 
- $O(N)$ = lineal 
- $O(N\ log(N))$ = "linearítmica" 
- $O(N^c)$ = polinómica 
- $O(c^N)$ = exponencial 
- $O(N!)$ = factorial 

N: tamaño del problema, ***c***: constante > 1

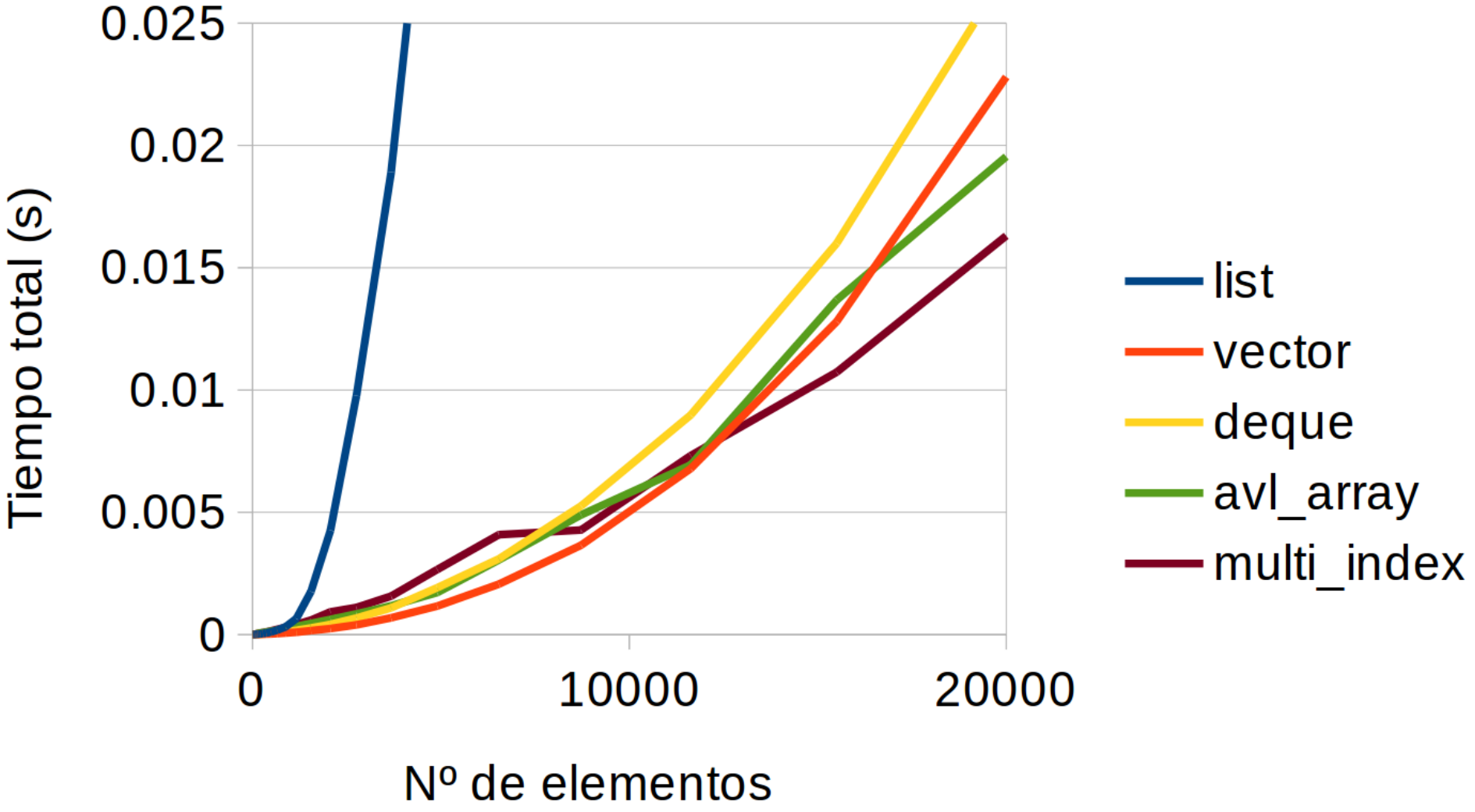
Complejidad computacional

	Acceso aleatorio	Inserción/ Extracción	**Suma de ambas**
Array			
Lista			
Súper Árbol			

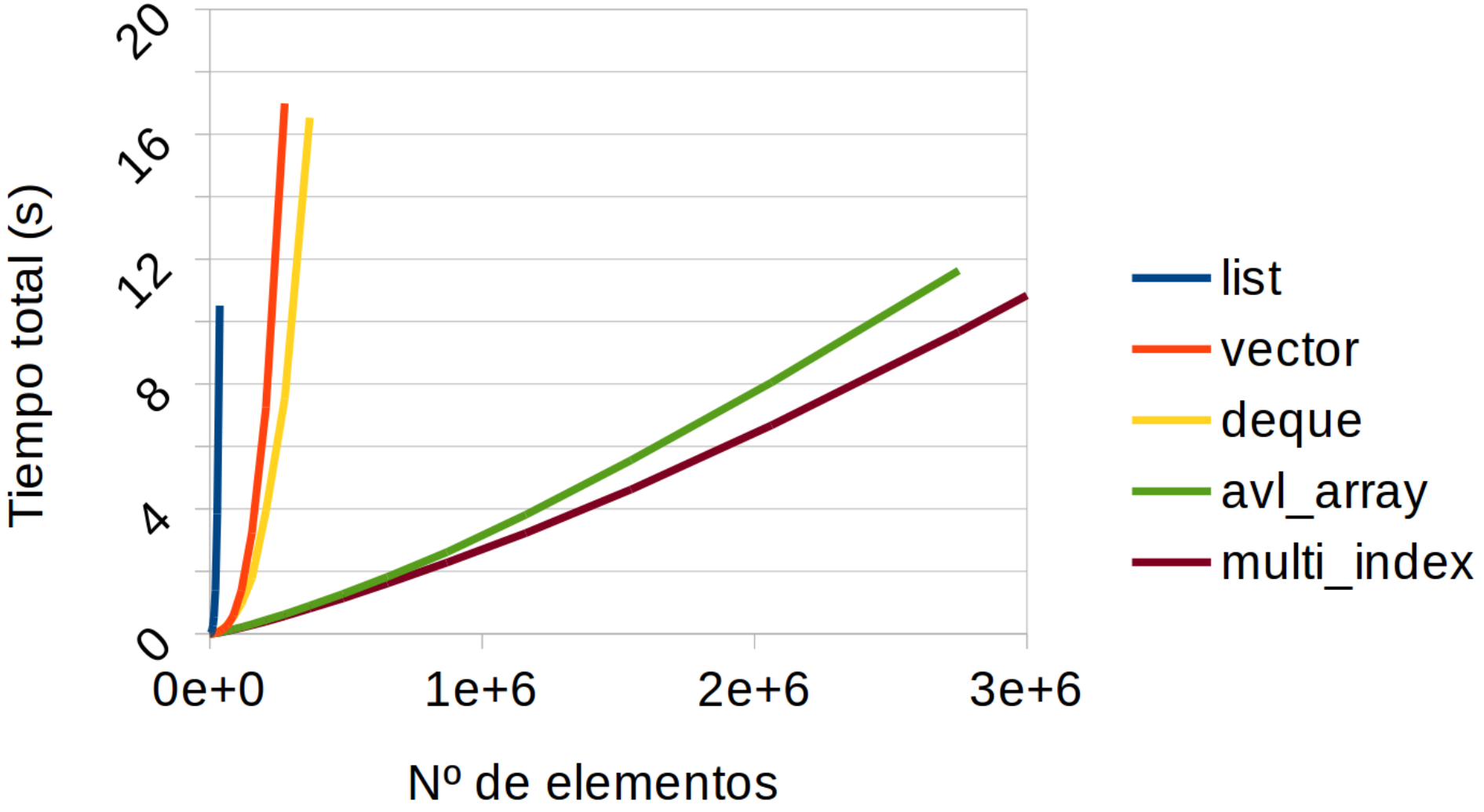
Complejidad computacional

	(1 acc. aleatorio + 1 ins./extr.)	$\times N =$ **total**
Array		
Lista		
Súper Árbol		

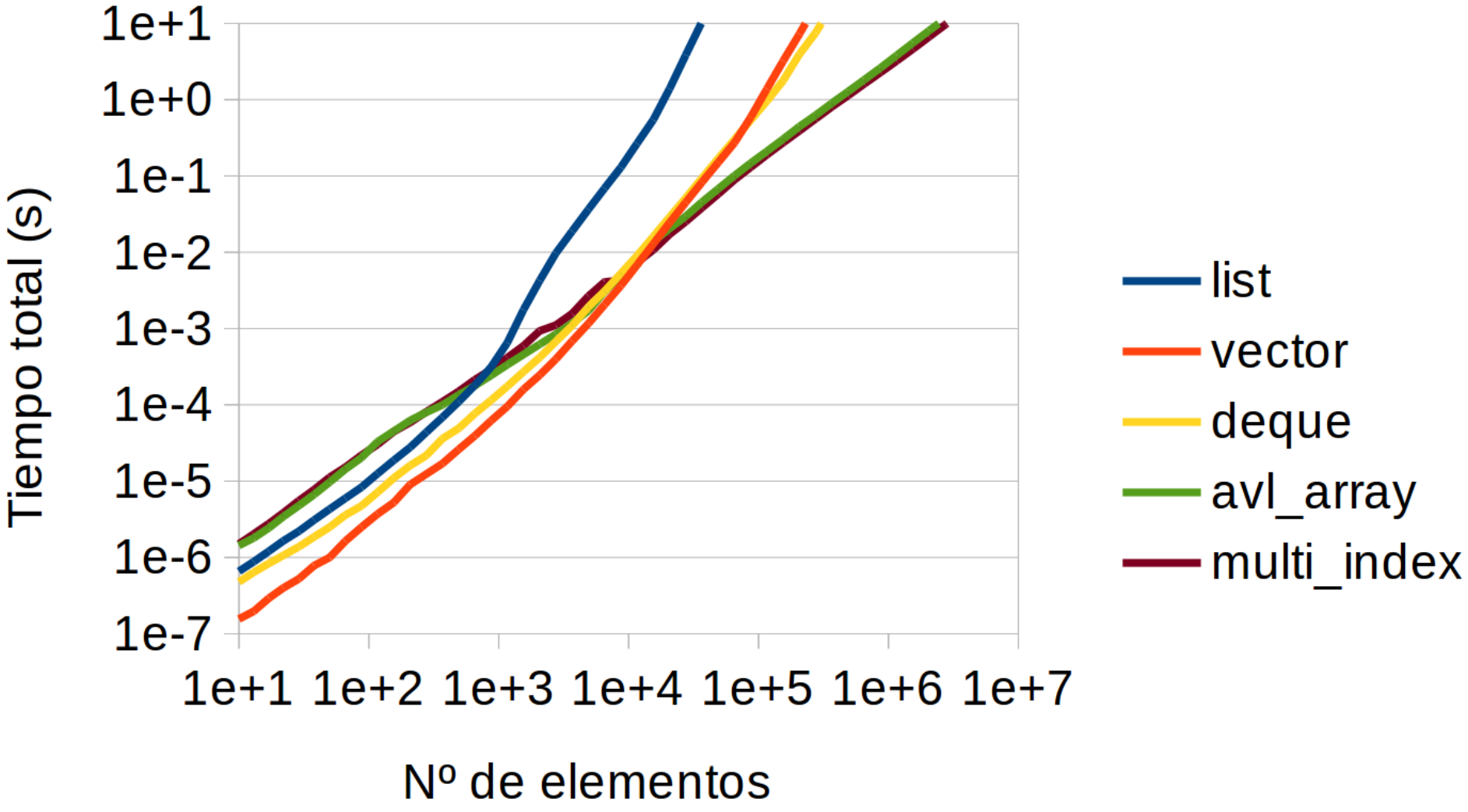
Resultado (1/3) — pocos elementos



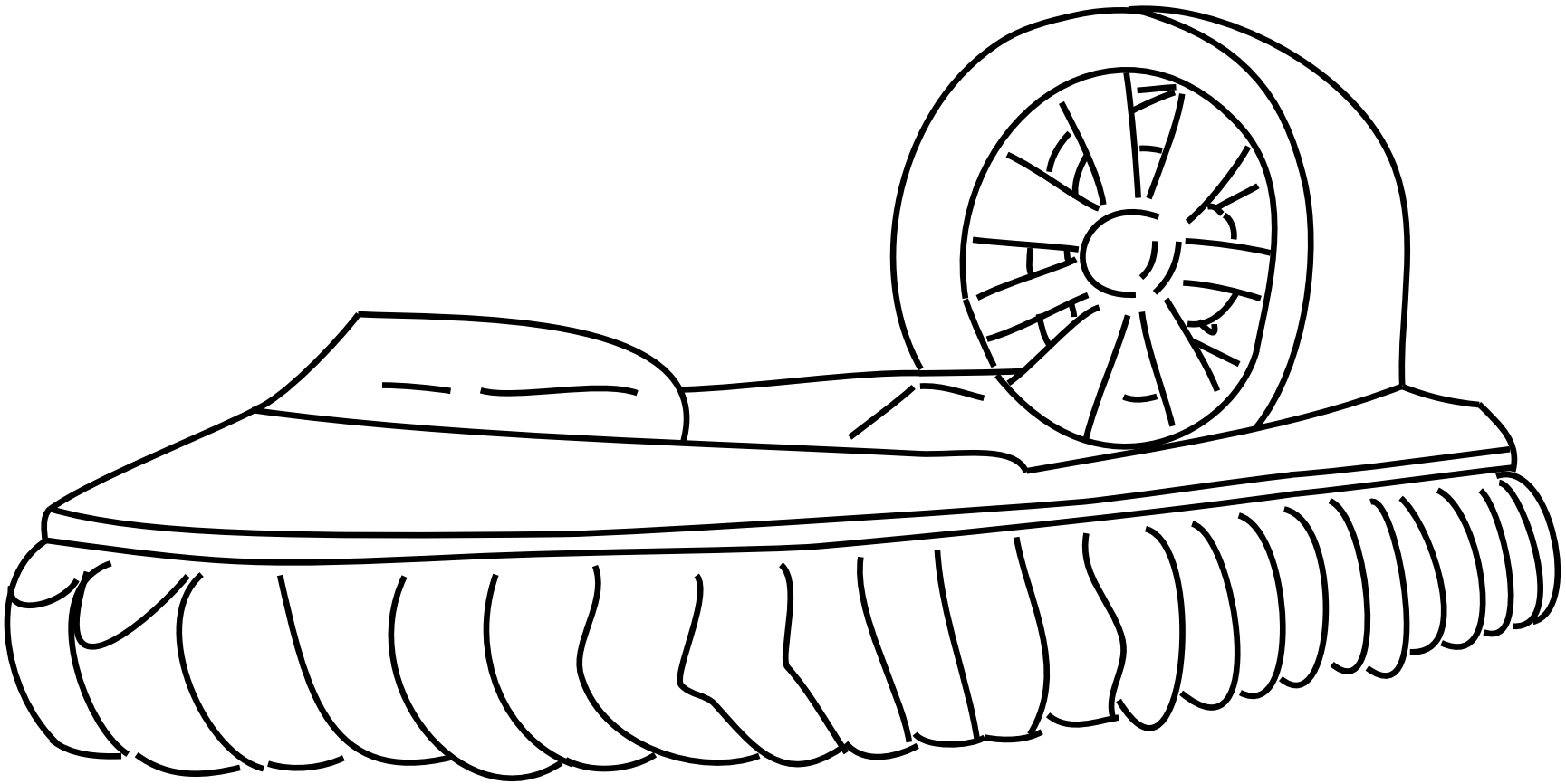
Resultado (2/3) — muchos elementos



Resultado (3/3) — escala logarítmica

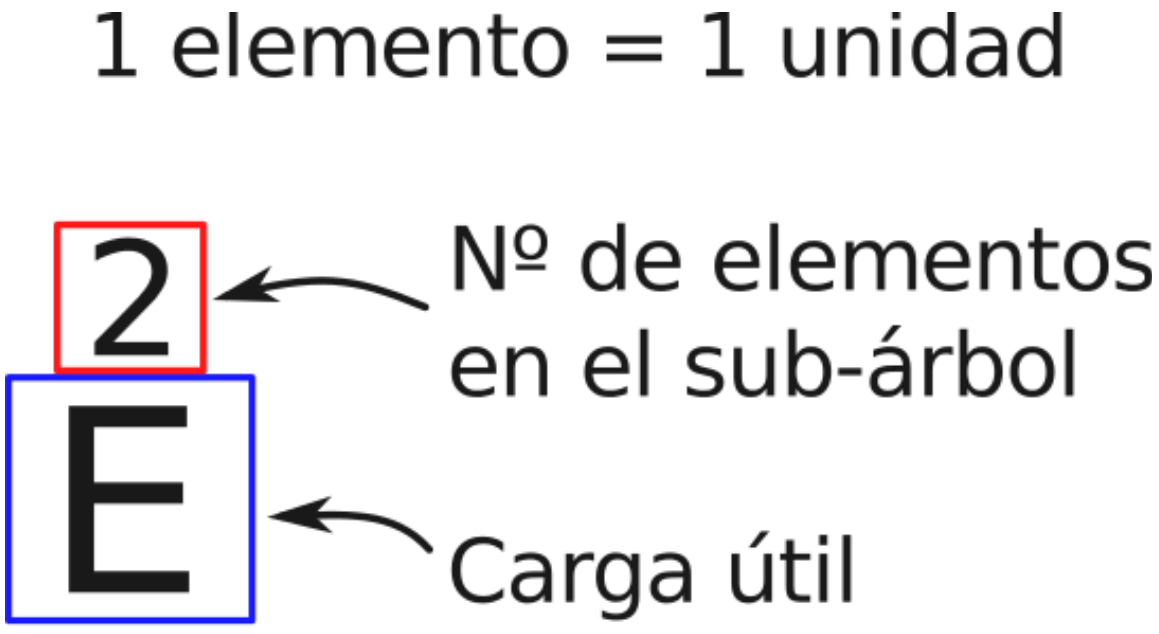


Ideal para la playa

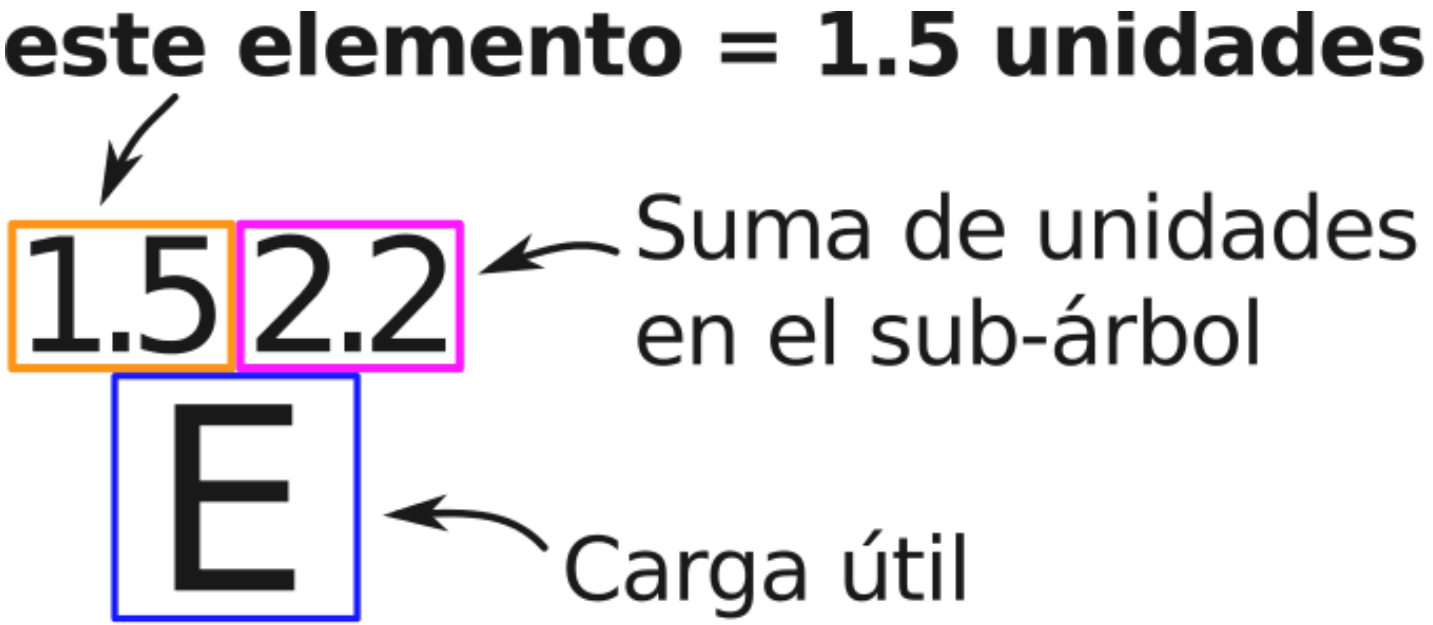


Vista **no** proporcional

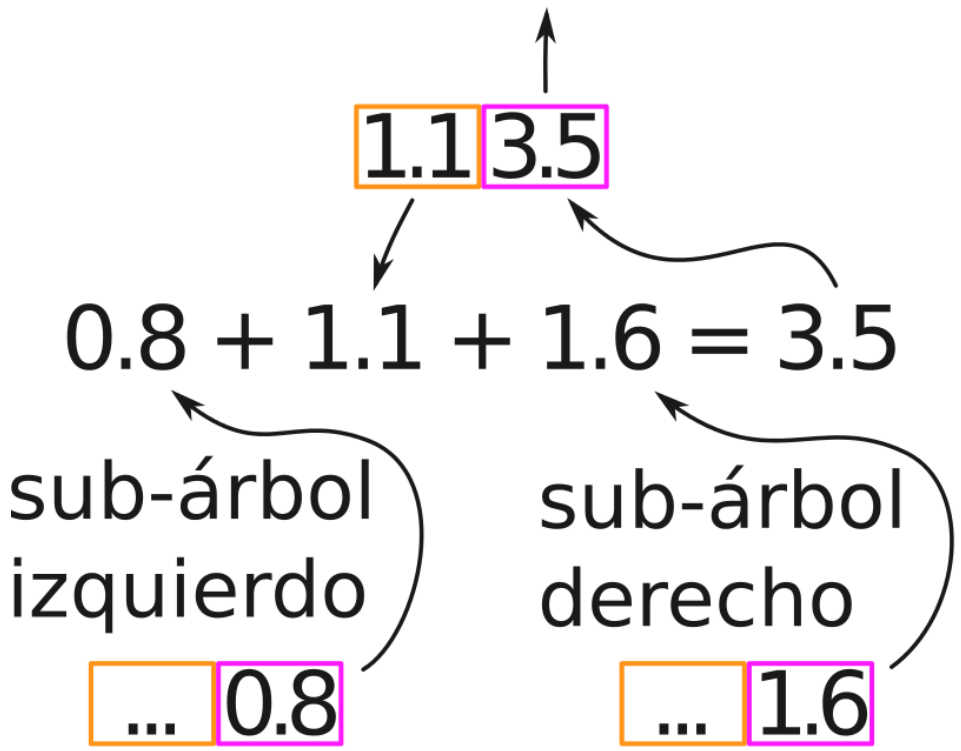
Leyenda en vista proporcional



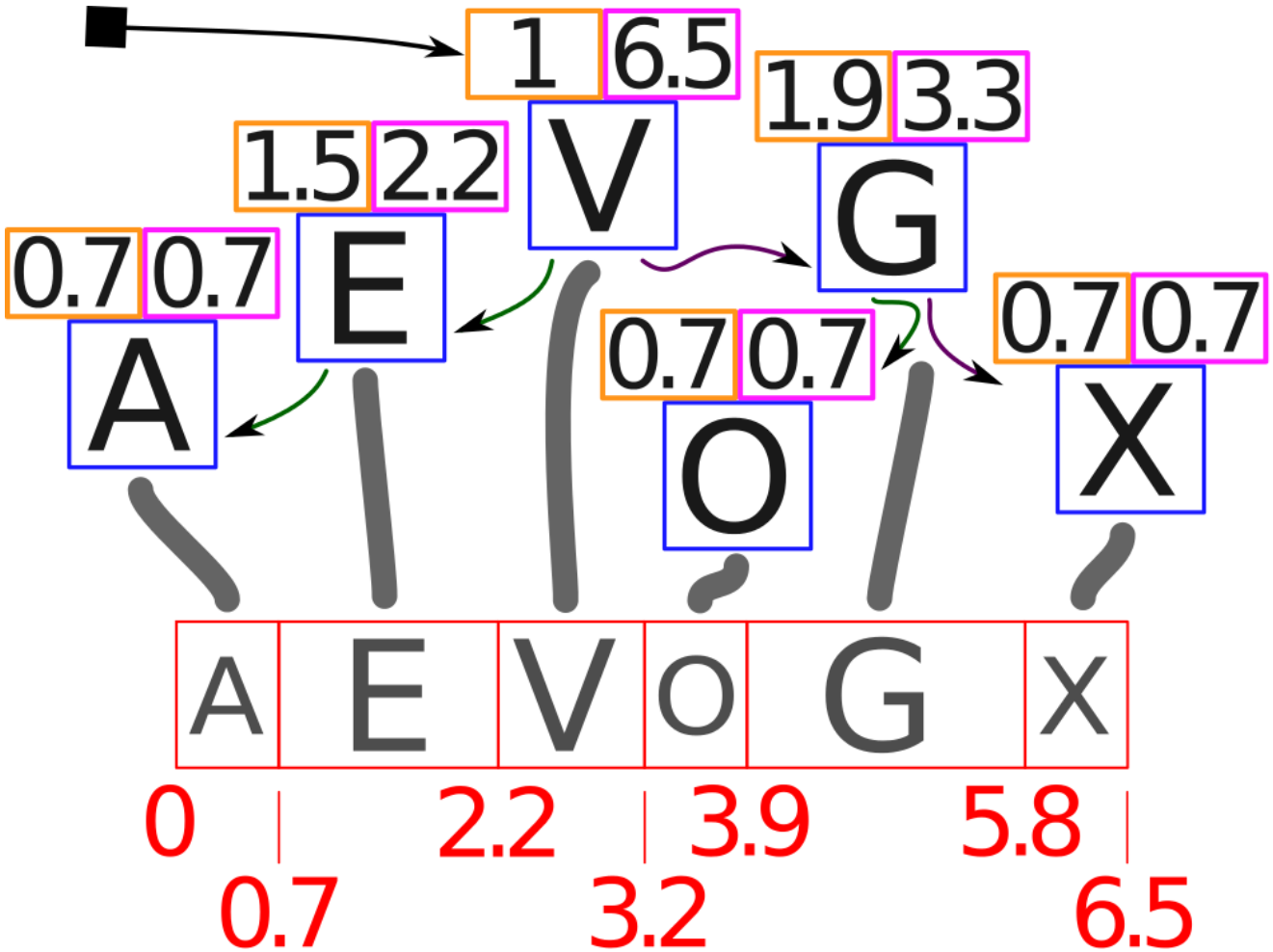
Leyenda en vista **no** proporcional



Suma en vista **no** proporcional



Vista **no** proporcional



Aplicaciones

Editor de texto

Secuencia de líneas

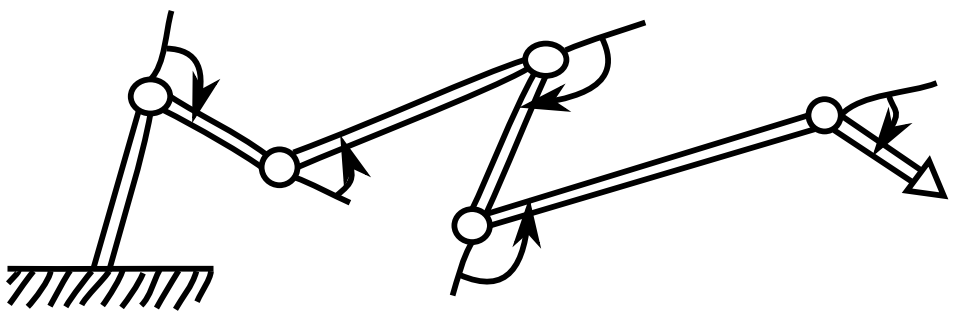
- Número de bytes
- Número de líneas tras el ajuste
- Número de caracteres
- Si no es texto plano, número de píxels

gtk

Árbol B+ “ad hoc” con número de caracteres y líneas

Brazo robot o cadena de moléculas

- Secuencia de transformaciones de traslación y rotación
- Operación de vista **no** proporcional: suma y producto de matrices



Versión en disco: `shiftable_files`

- Implementación basada en proyección de archivos en memoria
- Código horrible (¡macros!)
- Metadatos en el propio archivo
- Al cerrar se puede optar por:
  1. Recompactar el archivo, ó...
  2. dejarlo tal cuál, con los metadatos

¿Cómo seguir el rastro de las secciones?

Con una secuencia en memoria con vista **no** proporcional

Edicion de archivos XML gigantes

- Una primera pasada puede construir un índice en memoria (no necesariamente completo)
- Se puede insertar/extraer nodos sin reescribir todo el archivo
- Hay que mantener el índice al día, claro
- ¿Recompactar al cerrar?
  1. Sí: vuelve a ser un XML normal
  2. No: más rápido

Propuestas similares

Multi Index (1/2)

```
boost::multi_index_container
<
    T,
    boost::multi_index::indexed_by
    <
        boost::multi_index::ranked_non_unique
        <
            boost::multi_index::identity<T>,
            unordered_less<T>
        >
    >
>
```

Multi Index (2/2)

```
template<typename T>
struct unordered_less
{
    bool operator() (const T &,
                     const T &) const
    {
        return false;
    }
};
```

Propuestas similares en Boost (1/2)

- 2004 – La mención más antigua (no sé si implementada), por Peter Palotas

  `http://lists.boost.org/Archives/boost/2004/03/62823.php`
- 2006 – "Hierarchical Data Structures" por Bernhard Reiter y René Rivera

  `http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2006/n2101.html#tr.hierarchy.augment`
- 2006 – "AVL Array" (horrible nombre, lo sé)

  `http://sourceforge.net/projects/avl-array`

  "Rank List" tras debate en foro de Boost

Propuestas similares en Boost (2/2)

- 2012 – Countertree por Vadim Stadnik
  `http://dl.dropbox.com/u/8437476/works/countertree/doc/index.html` (enlace roto)
- 2015 – SegmentedTree por Chris Clearwater
  `https://det.github.io/segmented_tree/`

Propuestas similares fuera de Boost

- "Simon Tatham's Algorithms Page"
  `https://www.chiark.greenend.org.uk/~sgtatham/algorithms/cbtree.html`

  "Counted B-trees: An enhancement to the well known B-tree algorithms to allow you to **look up items in the tree by numeric index**, or to **find the numeric index of an item**. Useful for finding percentiles, [...]"

Propuestas similares en Python

- https://pypi.python.org/pypi/rbtree
- https://pypi.python.org/pypi/pyavl
- https://pypi.python.org/pypi/blist

Reflexionemos

Muchas gracias

¿Alguna pregunta?