Eficiencia empírica de algoritmos

Códigos

```cpp
/**
   @file C@lculo de la sucesi@n de Fibonacci
*/
#include <iostream>
using namespace std;
#include <ctime>
#include <chrono>
/**
   @brief Calcula el t@rmino n-@simo de la sucesi@n de Fibonacci.

   @param n: n@mero de orden del t@rmino buscado. n >= 1.

   @return: t@rmino n-@simo de la sucesi@n de Fibonacci.
*/

using namespace std::chrono;

int fibo(int n)
{
  if (n < 2)
    return 1;
  else
    return fibo(n-1) + fibo(n-2);
}

int main(int argc, char* argv[])
{
  high_resolution_clock::time_point t_antes, t_despues;
  int n;
  int f;

  //cout << "Numero del termino: ";
  //cin >> n;
  n=atoi(argv[1]);

  t_antes = high_resolution_clock::now();
  f = fibo(n);
  t_despues = high_resolution_clock::now();

  //cout << "El t@rmino " << n << "-esimo es: " << f << endl;
  duration<double> transcurrido = duration_cast<duration<double> >(t_despues-t_antes);
  cout << n << " " << transcurrido.count() << "\n";

  return 0;
}
```

```
################################################################################
# Makefile
################################################################################

STUDDATA = $(wildcard data/*/*.dat)
STUDGRPH = $(wildcard grphx/*/*.{jpg,png})

BIN = bin

SRC = $(wildcard src/*.cpp)
EXE = $(basename $(SRC))
ALG = $(wildcard *.cpp)

CXXFLAGS = -std=gnu++0x

default: $(EXE)

%: %.cpp
  $(CXX) $(CXXFLAGS) $< -o $(patsubst src%,bin%,$(basename $< ))

clean:
  $(RM) -v $(BIN)/*

mrproper: clean
  $(RM) -v $(BIN)/* $(STUDDATA) $(STUDGRPH)

.PHONY: clean
.PRECIOUS: $(LOG)
.NOEXPORT:

################################################################################
```

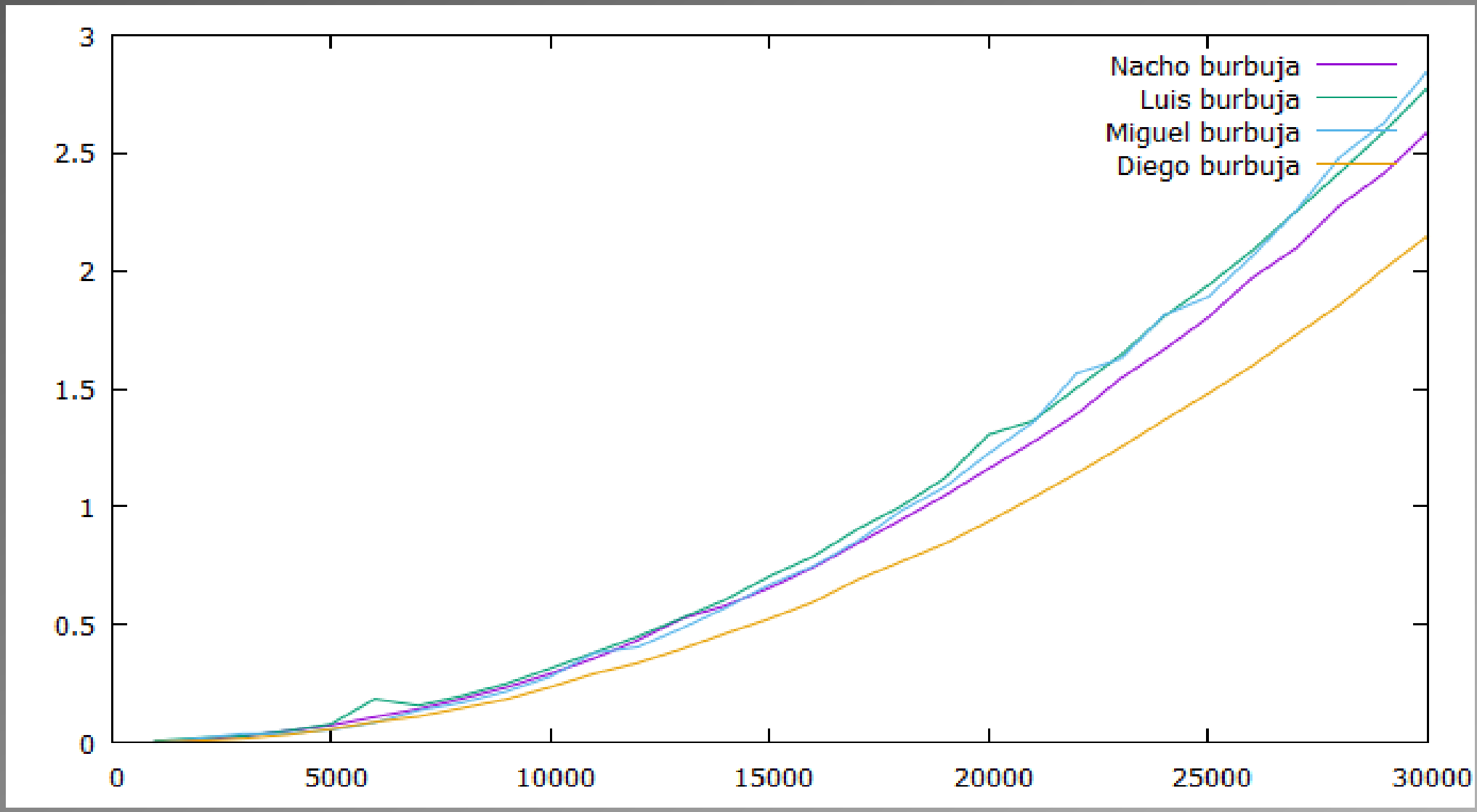
Códigos

```bash
#!/bin/bash
#Burbuja

i=10000
echo -e "Ejecutando burbuja con tamaño maximo 30.000 a saltos de 1.000"
sleep 3
while [ $i -le 30000 ]
do
  ./bin/burbuja $i >> ./data/$1/burbuja.dat
  echo -e "Iteración $i"
   ((i+=1000))
done

#Fibonacci

i=1
echo -e "Ejecutando fibonacci con tamaño maximo 45 a saltos de 1"
sleep 3
while [ $i -le 45 ]
do
  ./bin/fibonacci $i >> ./data/$1/fibonacci.dat
  echo -e "Iteración $i"
  ((i+=1))
done

#heapsort

i=10000
echo -e "Ejecutando heapsort con tamaño maximo 1.000.000 a saltos de 10.000"
sleep 3
while [ $i -le 1000000 ]
do
  ./bin/heapsort $i >> ./data/$1/heapsort.dat
  echo -e "Iteración $i"
  ((i+=10000))
done

#insercion

i=1000
echo -e "Ejecutando insercion con tamaño maximo 30.000 a saltos de 1.000"
sleep 3
while [ $i -le 30000 ]
do
  ./bin/insercion $i >> ./data/$1/insercion.dat
  echo -e "Iteración $i"
  ((i+=1000))
```

```python
#!/usr/bin/python

import os
import sys
import re
import numpy as np
import matplotlib.pyplot as plt

path= './data/' + str(sys.argv[1]) + '/'

for filename in os.listdir(path):

    with open(path+filename) as f:
        data = f.read()

    data = data.split('\n')

    data = [row for row in data if row != ""]
    x = [row.split(' ')[0] for row in data]
    y = [row.split(' ')[1] for row in data]

    fig = plt.figure()

    ax1 = fig.add_subplot(111)


    title = filename.replace(".dat","")

    ax1.set_title(title)
    ax1.set_xlabel('Tiempo')
    ax1.set_ylabel('Ejecuciones')

    ax1.plot(x,y, c='r', label=title)


    savepath = path.replace("data","grphx")

    d = os.path.dirname(savepath+title+".png")
    if not os.path.exists(d):
        os.makedirs(d)

    plt.savefig(savepath+title+".png")
```

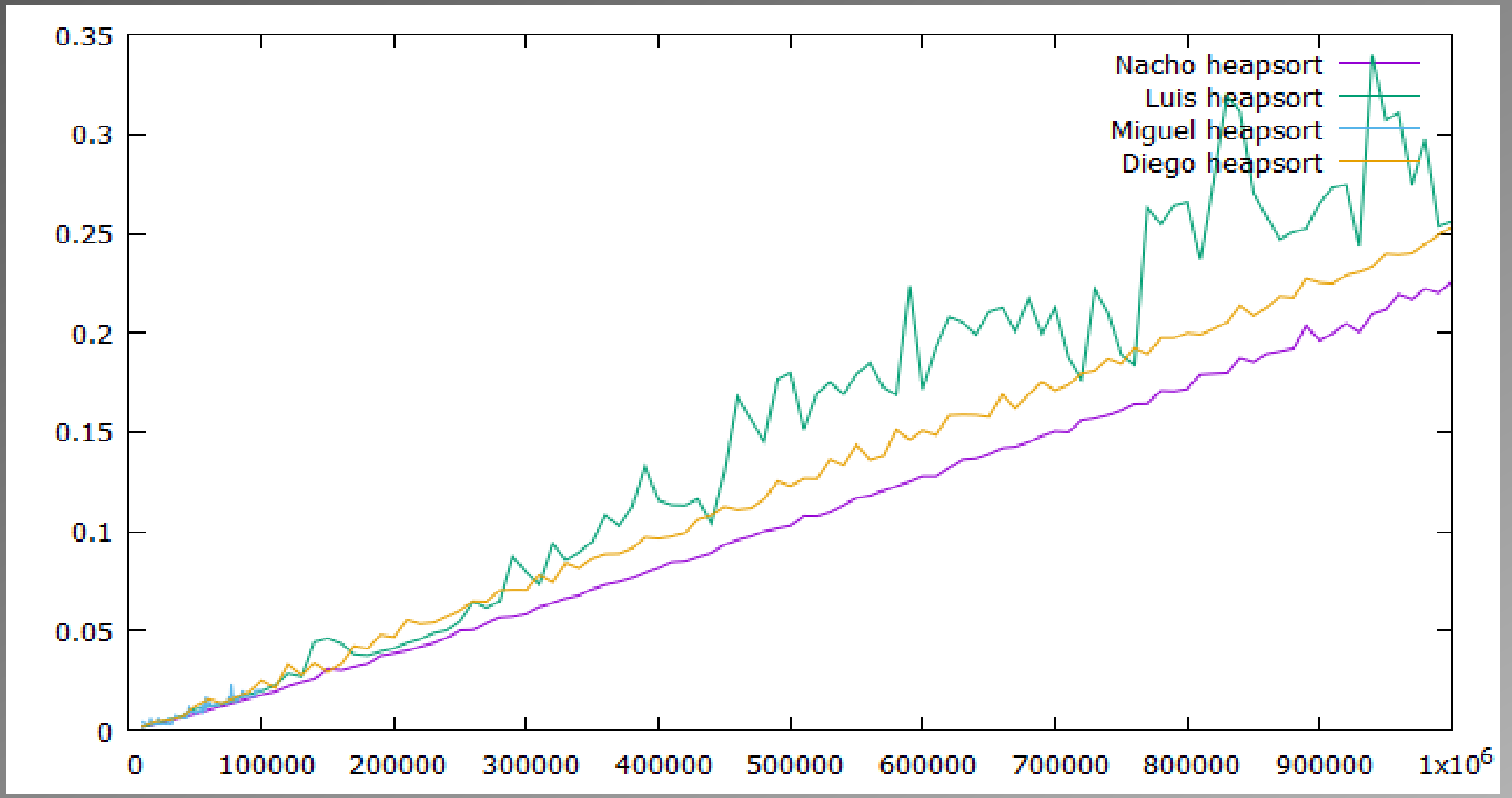
Algoritmos de ordenación

Burbuja



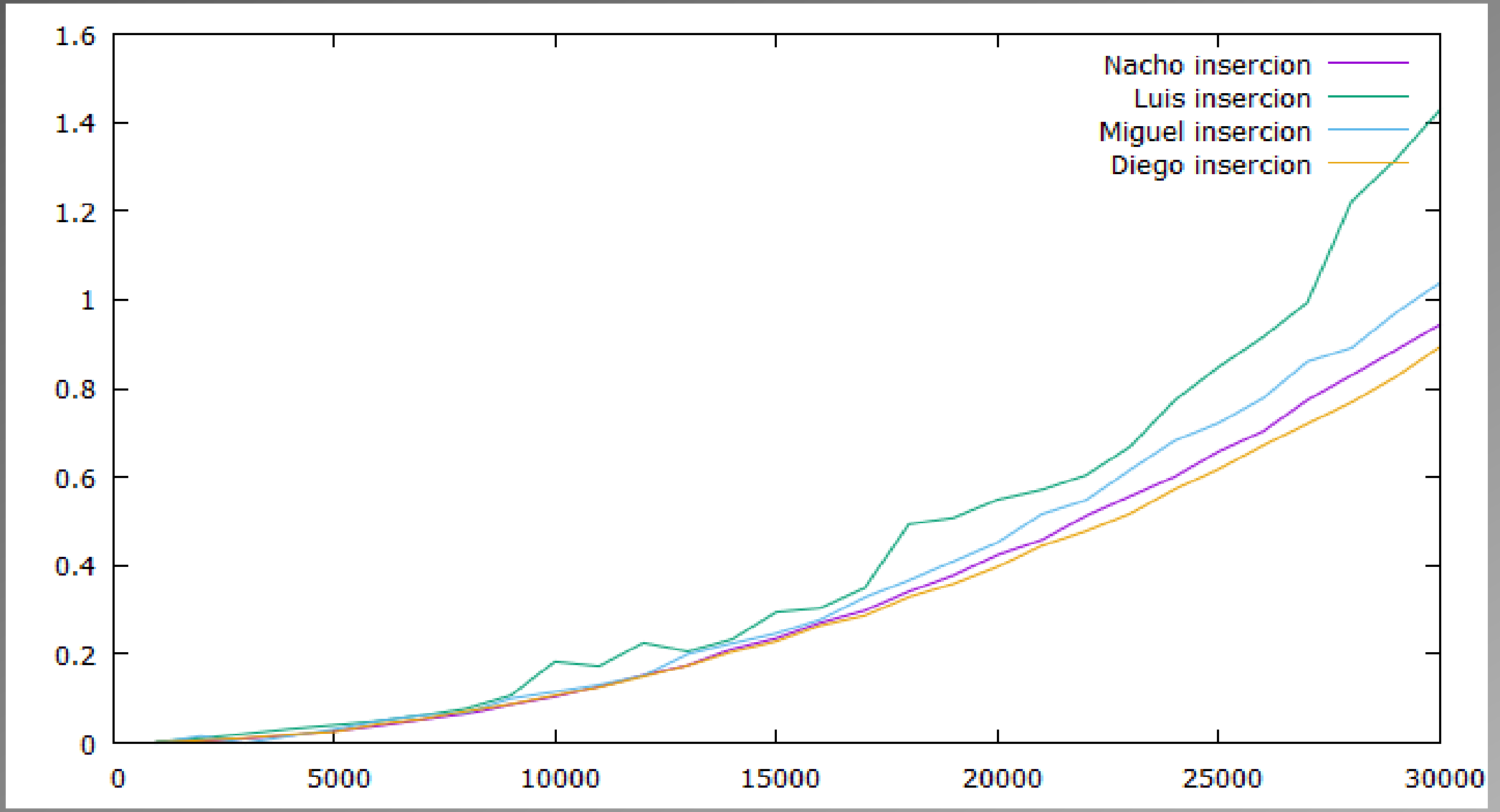
Algoritmos de ordenación

Heapsort



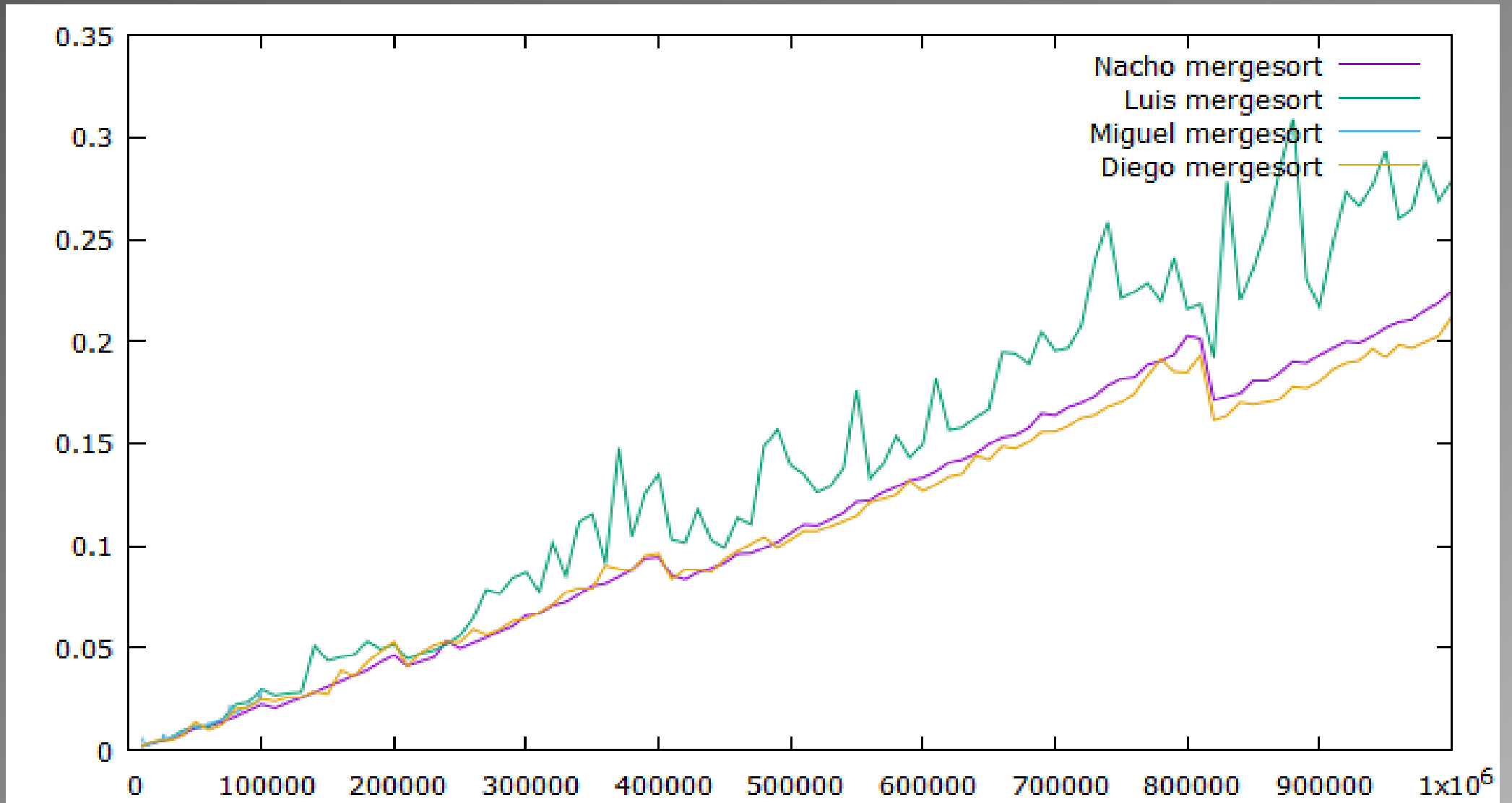
Algoritmos de ordenación

Inserción



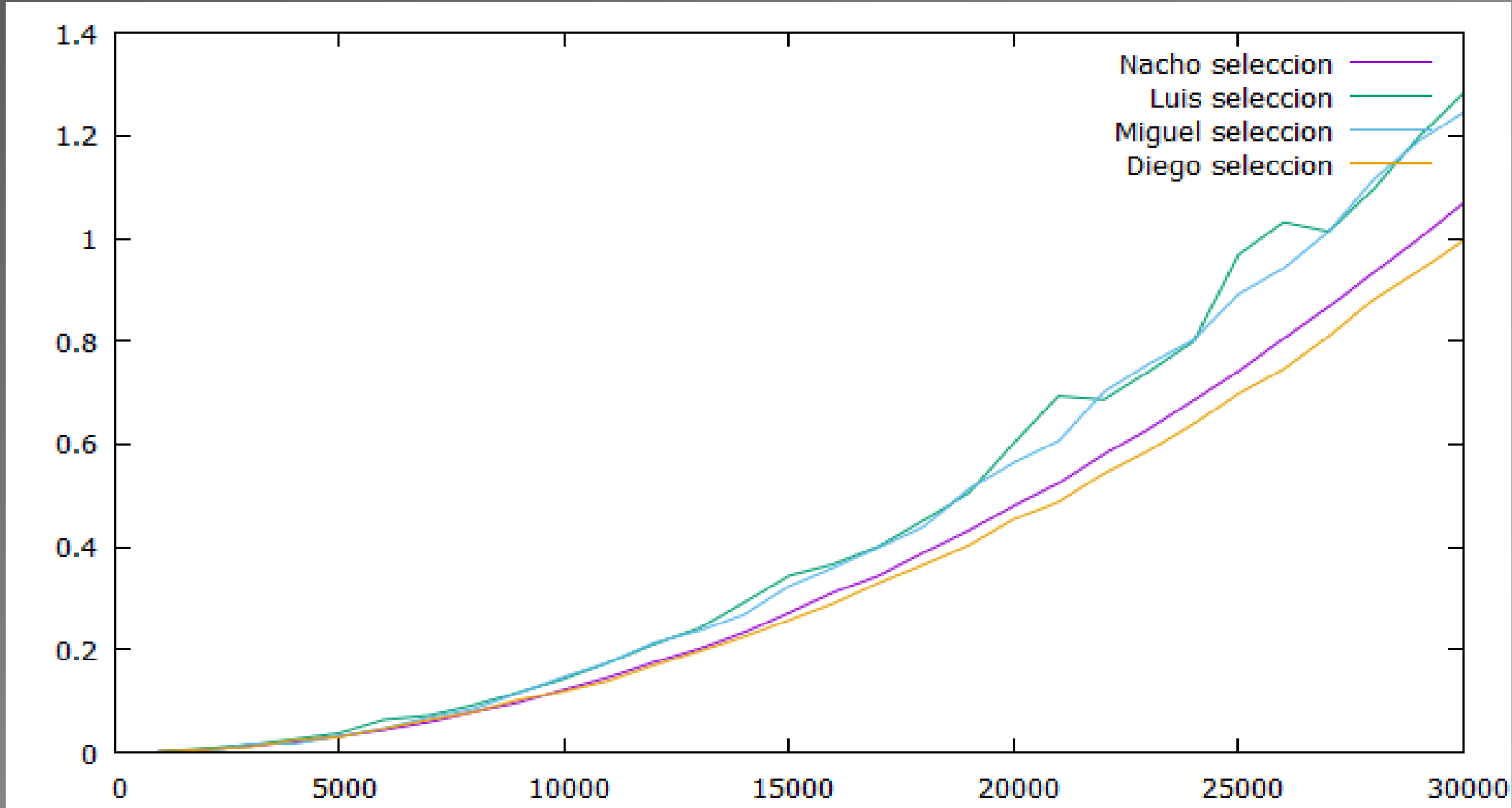
Algoritmos de ordenación

Mergesort

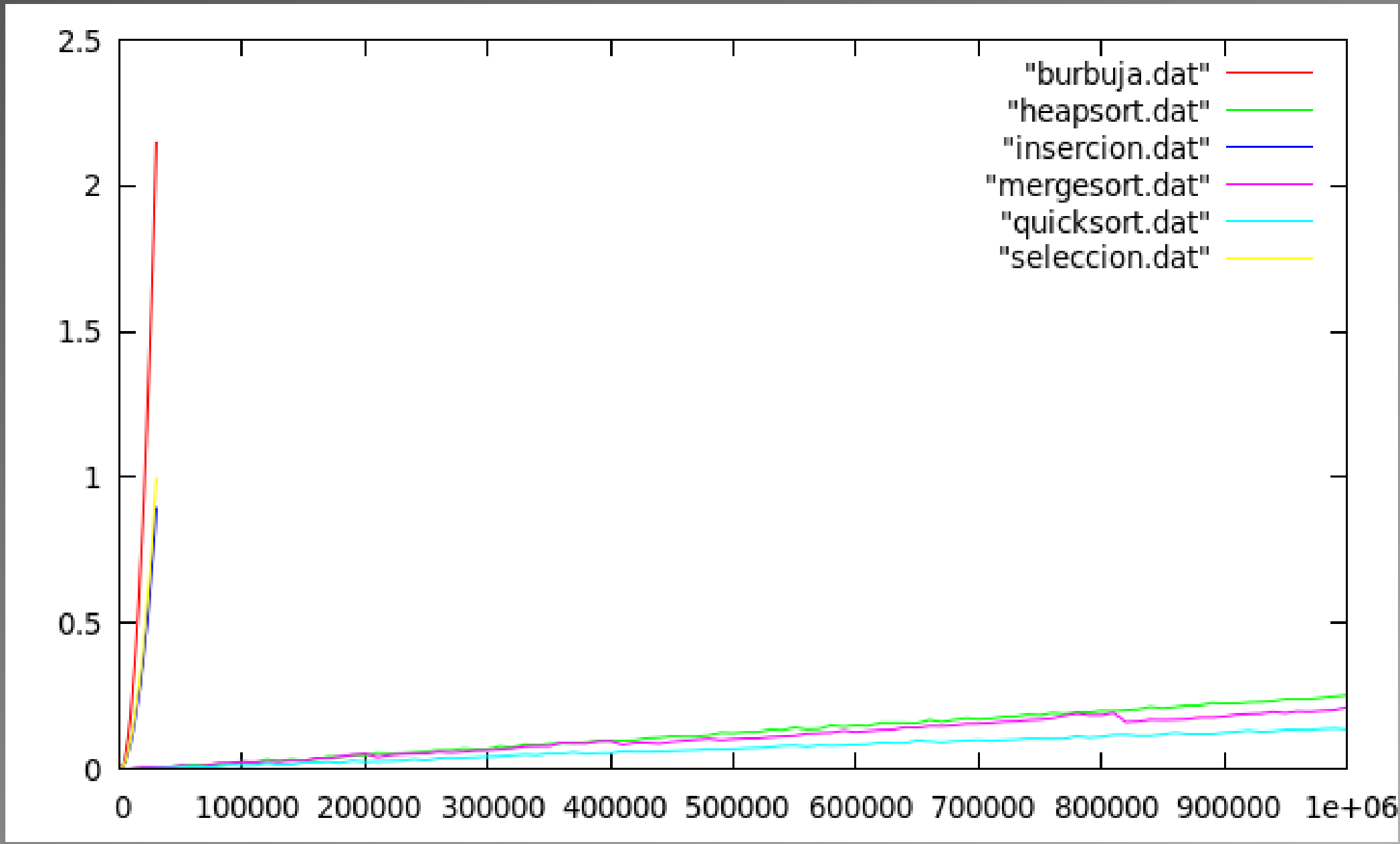


Algoritmos de ordenación

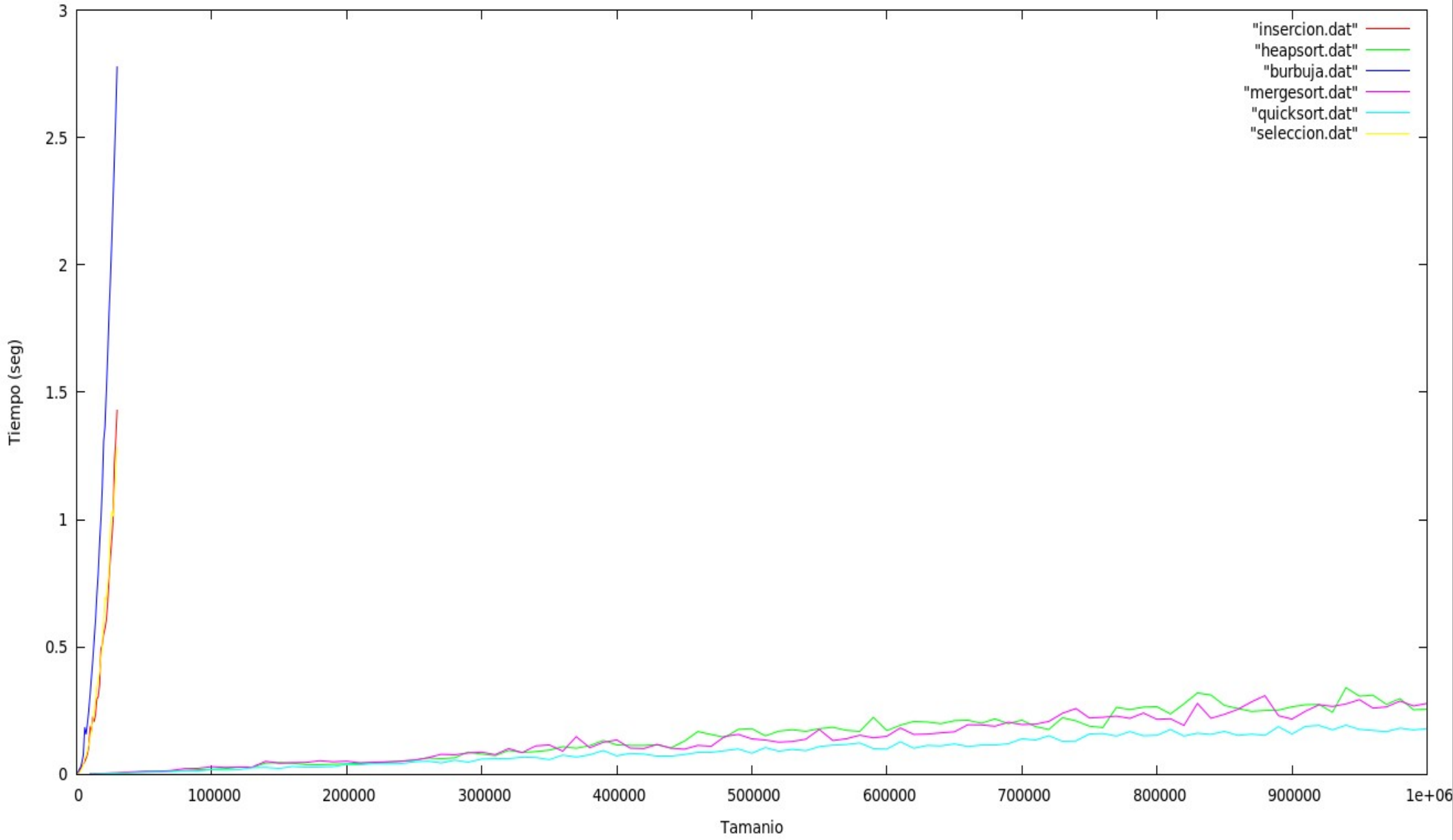
Selección



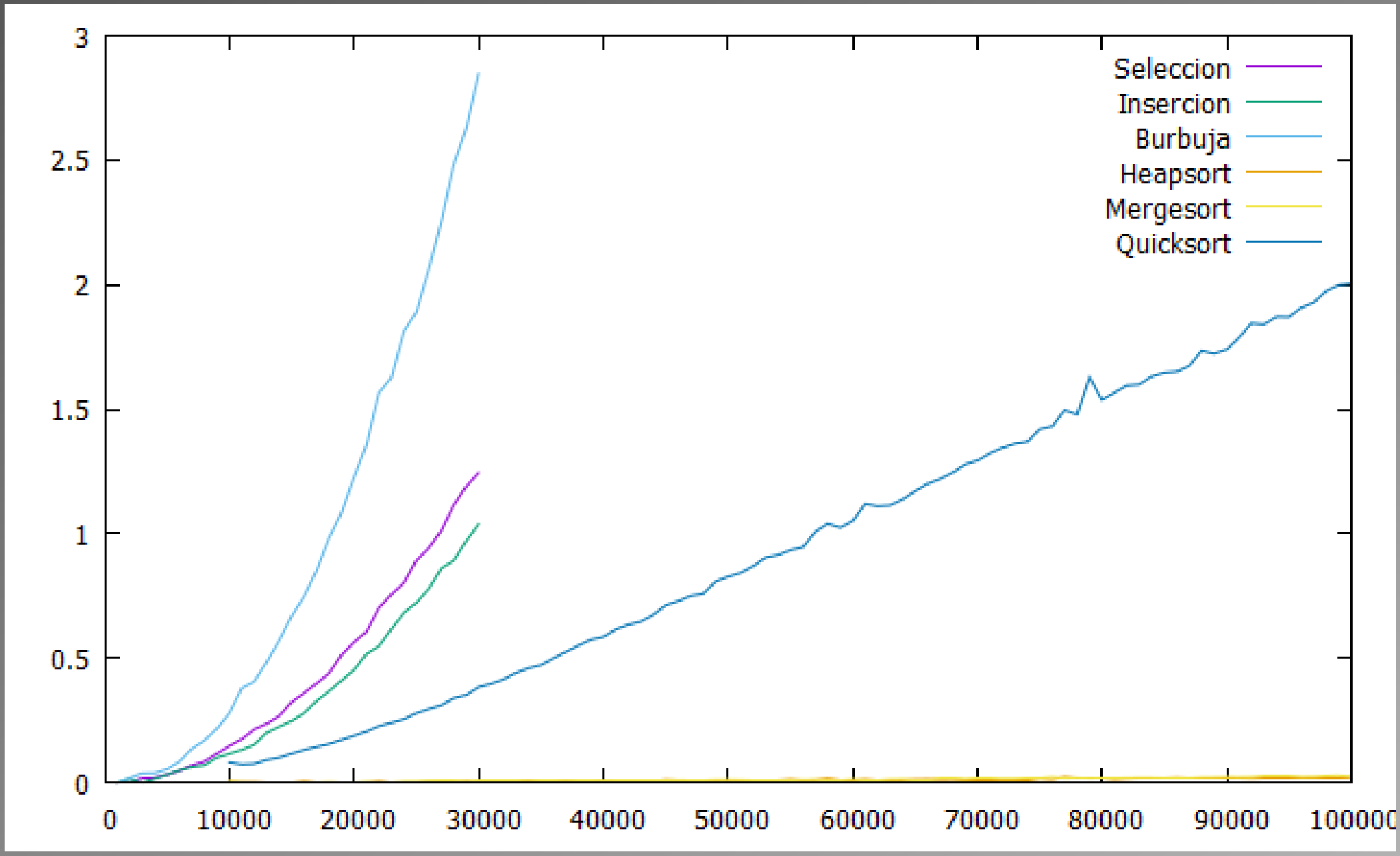
Algoritmos de ordenación



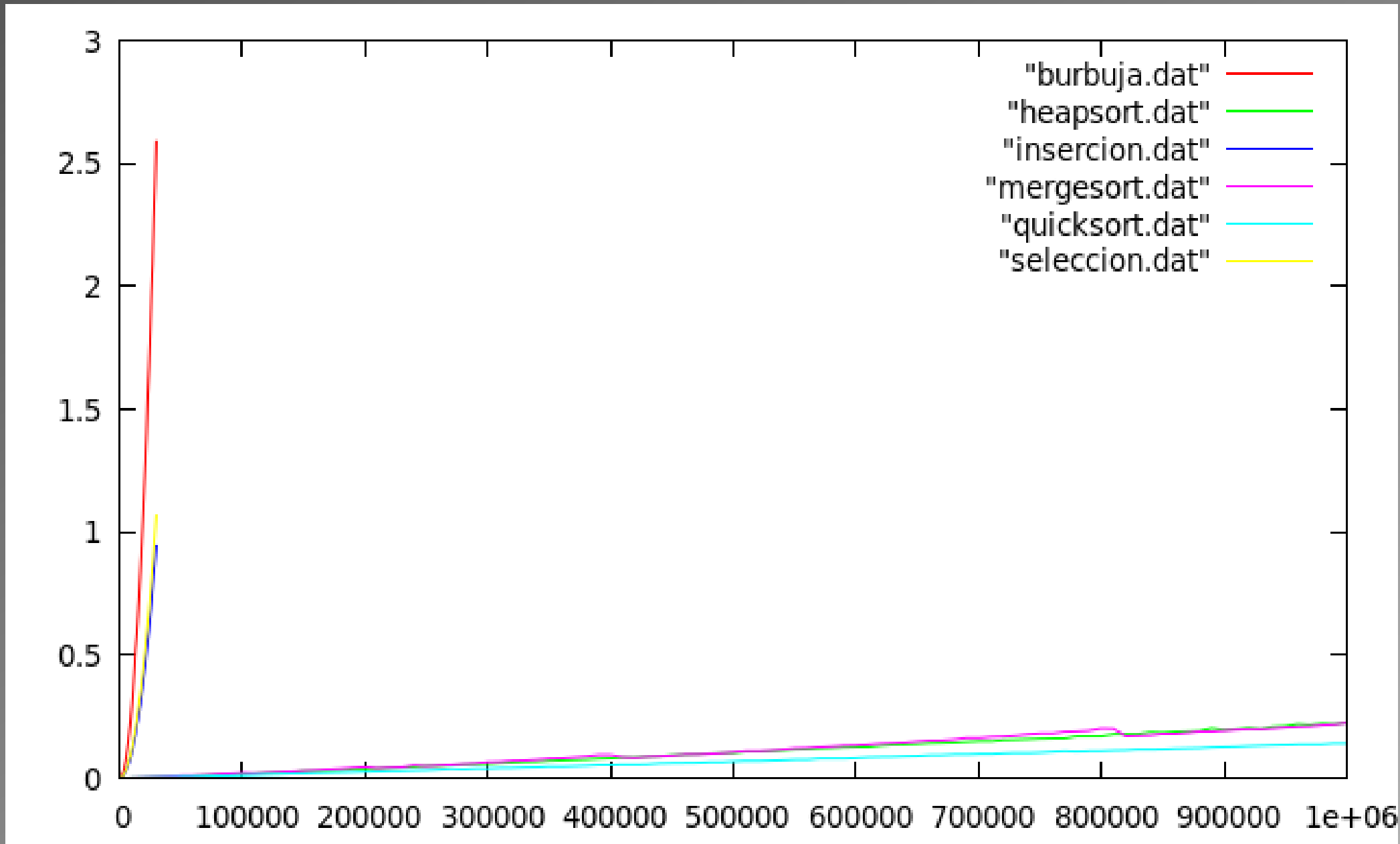
Algoritmos de ordenación



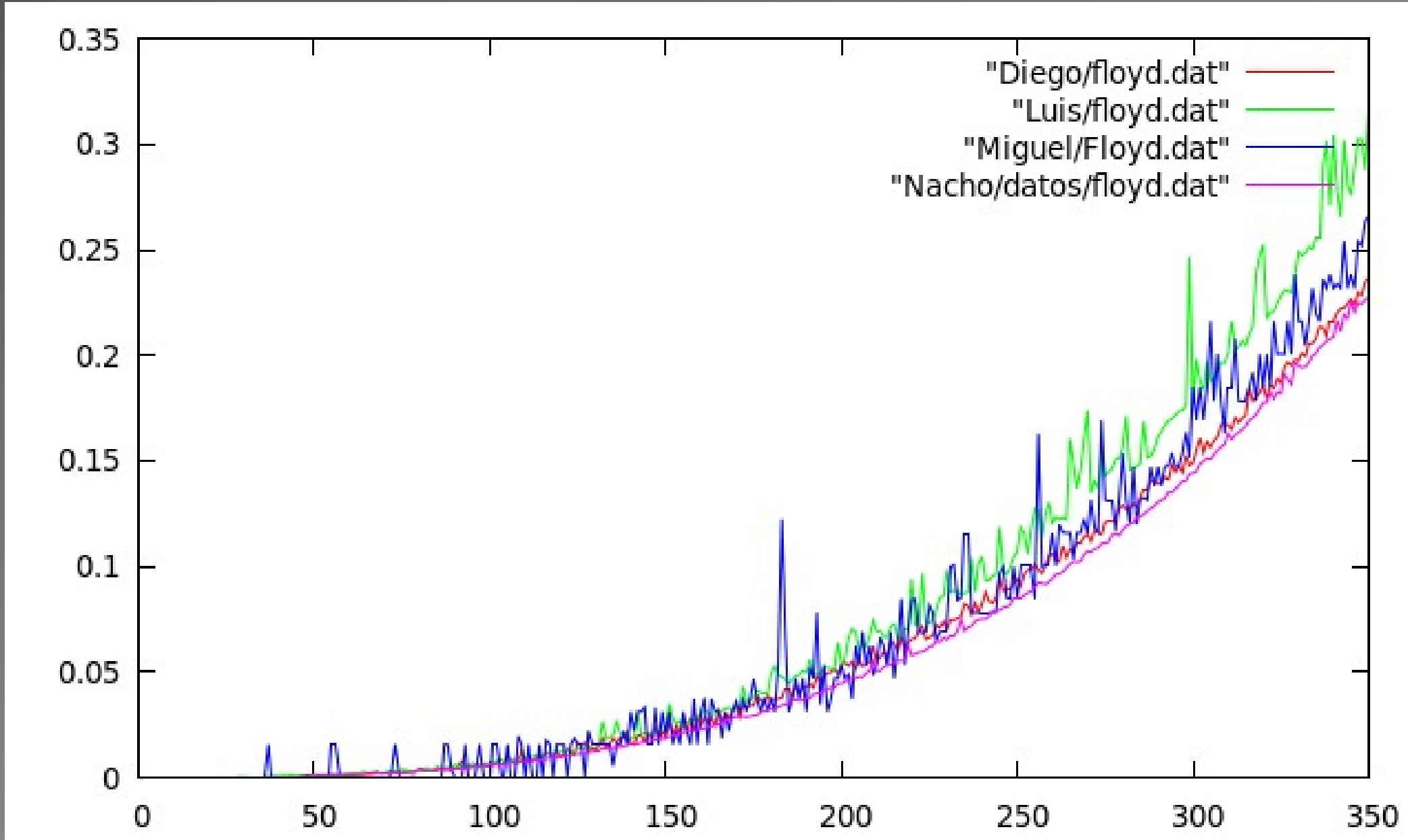
Algoritmos de ordenación



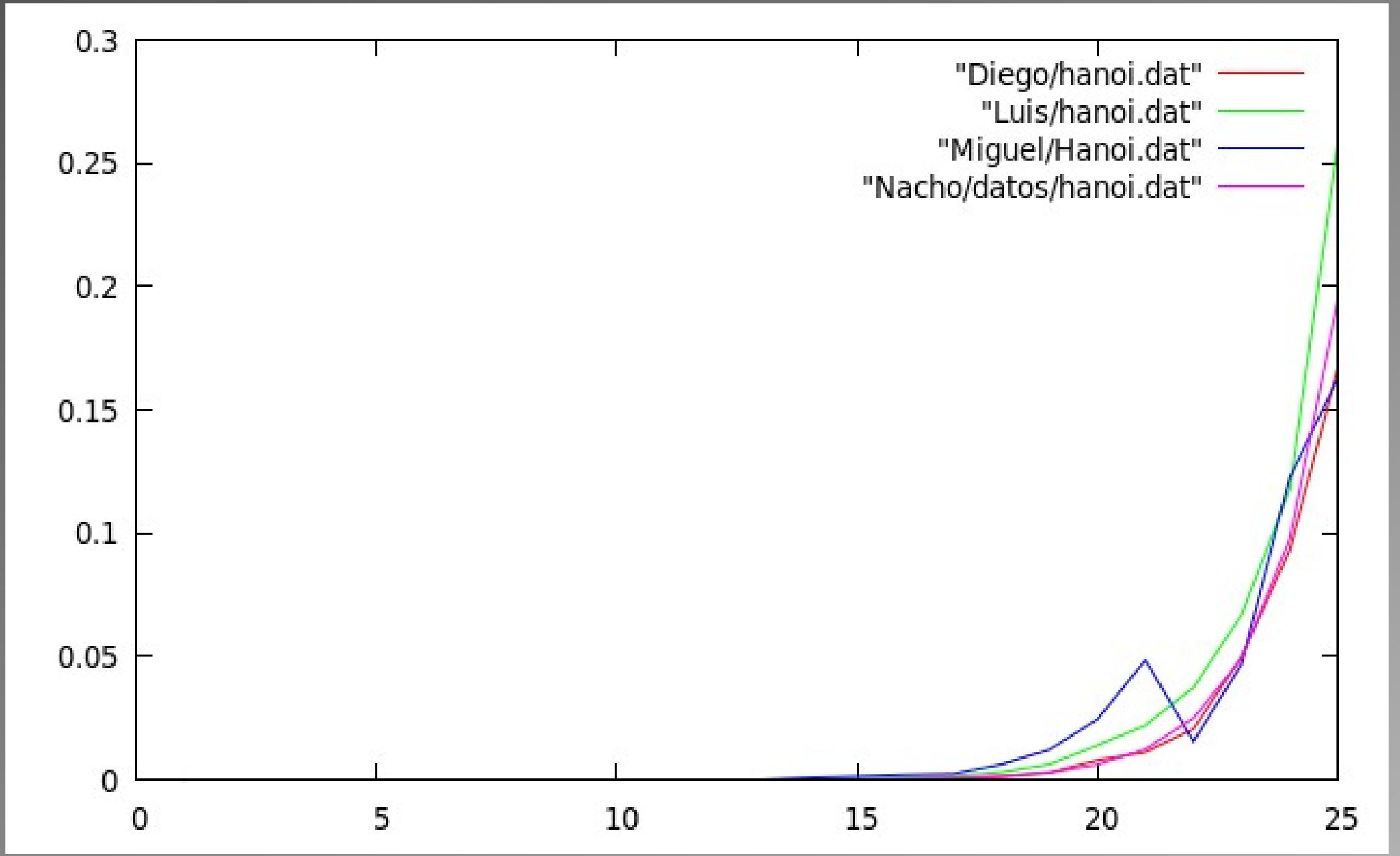
Algoritmos de ordenación



Algoritmo de Floyd



Algoritmo de Hanoi



Algoritmo de Fibonacci