Centro Universitario de Ciencias Exactas e Ingenierías
Universidad de Guadalajara

Figura 1: Escudo de la Universidad de Guadalajara

Seminario de Solución de Problemas de Algoritmia

Ciclo: 2025A Sección: D08

Fecha de realización: 09 / 03 / 2025

Actividad #2

Primer miniproyecto del curso "Programación y matemática" (parte 3)

Asesor y profesor: Juan José López Cisneros

Código de estudiante: 217326708
Carrera: Ingeniería Informática (INNI)
Nombre: Alan Gómez Pasillas

Horas de realización: Aproximadamente 20:36:10

Primer miniproyecto del curso "Programación y matemática" (parte 3)

La actividad consiste en utilizar los programas que habíamos creado para realizar los procedimientos en números más grandes y en un rango de numeros de 2,000,000. Unos de los requerimientos más importantes es seguir las buenas prácticas de la programación y tener un código limpio a la misma vez que utilizamos programación modular.

El objetivo de la actividad consiste en probar diferentes formas de resolver un problema, utilizando lo que se conoce como fuerza bruta y algoritmos especiales que se han encontrado para medir el tiempo que tardan en ejecutarse y comparar los resultados.

Desarrollo

Para ésta tarea primero revisé los ejercicios anteriores y los modifique para que pudiesen cumplir con los nuevos requisitos que solicitaba la actividad, después comencé a probarlos y esperar el tiempo necesario para procesar los datos.

Los resultados obtenidos son muy interesantes, vamos a revisar primero los de la actividad P03.

El rango a iterar lo definimos entre los estudiantes del salón, entre todos decidimos iterar un rango de 1000 numeros iniciando desde el C000 0000 en base 16 lo cual es un aproximado de tres millones en base 10.

Cantidad de primos (Base 10)	1. Primos por booleano (Tiempo/operaciones)	2. Primos por cantidad de divisores (Tiempo/operaciones)	3. Primos al primer divisor encontrado (Tiempo/operaciones)	4. Primos por raíz cuadrada (Tiempo/operaciones)	5. Primos por recursión (Tiempo/operaciones)	6. Criba de eratóstenes (Tiempo/Operaciones)	Procesado en: C++/Python	Descripción de la máquina (Procesador, RAM, S.O.)	Aportado por (Apellido (s), nombre (s))
47	03:36:00.8617 / 3,224,447,195,970	03:51:27.8894 / 3,224,447,195,970	00:09:25.7791 / 151,398,001,012	00:00:00.4737 / 56,810,754	00:00:00.0510 / 3,049,153	00:00:00.0000 / 1,001	C++	**CPU**: Intel Core i5 2.80 GHz x 4 **Memory**: 7.7GiB **SO**: Debian 13 x64	Gómez Pasillas Alan

Tabla 1: Total de operaciones y tiempos de ejecución P03

El P04 consiste en procesar un archivo de numeros aleatorios dado por el profesor y determinar la cantidad de numeros primos en él.

Aportado por (Apellido (s), nombre (s))	Cantidad de primos (Base 10)	1. Primos por booleano (Tiempo/operaciones)	2. Primos por cantidad de divisores (Tiempo/operaciones)	3. Primos al primer divisor encontrado (Tiempo/operaciones)	4. Primos por raíz cuadrada (Tiempo/operaciones)	5. Primos por recursión (Tiempo/operaciones)	6. Criba de eratóstenes (Tiempo/Operaciones)	Descripción de la máquina (Procesador, RAM, S.O.)
Gómez Pasillas Alan	148456	05:18:22.8275 / 1,999,367,971,807	05:19:19.7308 / 1,999,367,971,807	00:23:55.8312 / 142,721,756,241	00:00:27.5522 / 1,882,181,047	00:00:03.7508 / 178,977,979	00:00:00.0127 / 2,000,000	**CPU**: Intel Core i5 2.80 GHz x 4 **Memory**: 7.7GiB **SO**: Debian 13 x64

Tabla 2: Total de operaciones y tiempos de ejecución P04

Posteriormente hice unas graficas que hacen la comparación de resultados de una manera más visual.

```
import matplotlib.pyplot as plt
import numpy as np

plt.subplots(figsize=(12, 8))

plt.bar(Data.Solucion, Data.Operaciones, color ='r', width = 0.8)

plt.title('Comparacion de Operaciones')
plt.xlabel('Solucion')
plt.ylabel('Operaciones')
plt.show()
```

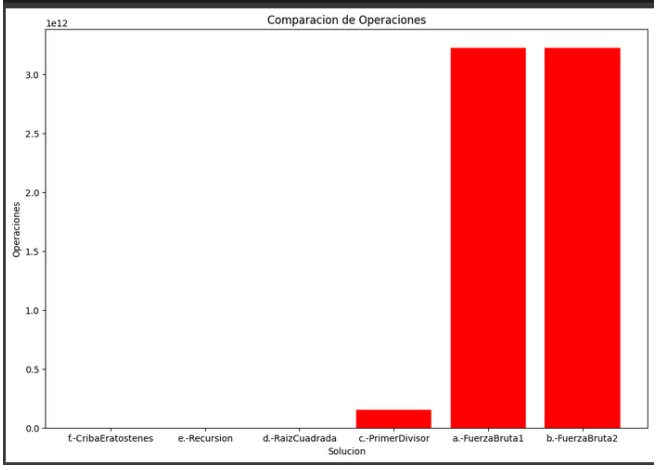

Tabla 4: Gráfica que compara las operaciones realizadas de las soluciones.

```
import matplotlib.pyplot as plt
import numpy as np

plt.subplots(figsize=(12, 8))

plt.bar(Data.Solucion, Data.Tiempo, color ='r', width = 0.8)

plt.title('Comparacion de Tiempos')
plt.xlabel('Solucion')
plt.ylabel('Tiempos')
plt.show()
```

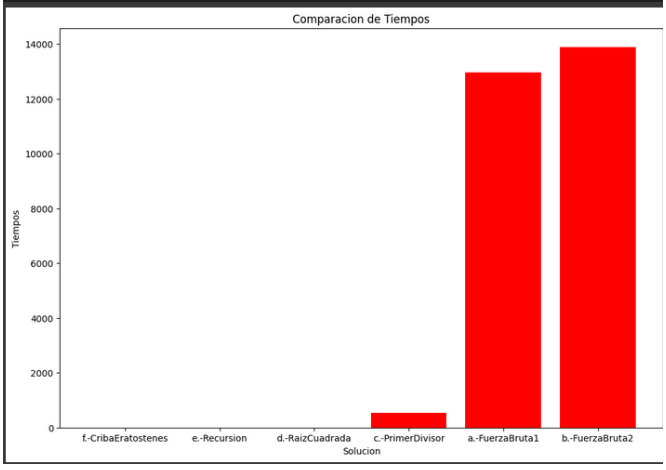

Tabla 3: Gráfica que compara los tiempos de cada solución

```
File  Edit  View  Terminal  Tabs  Help
alan@chapita:~/Universidad/SSPA/Ejercicios-de-Algoritmia/A02/C++/P04$ ls
a.exe                cout.txt              e.-Recursion.cpp
a.-FuerzaBruta1.cpp  c.-PrimerDivisor.cpp  f.-CribaEratostenes.cpp
aout.txt             d.exe                 f.exe
b.exe                dout.txt              fout.txt
b.-FuerzaBruta2.cpp  d.-RaizCuadrada.cpp   tarea2M.txt
bout.txt             e.exe
c.exe                eout.txt
alan@chapita:~/Universidad/SSPA/Ejercicios-de-Algoritmia/A02/C++/P04$ cat *t.txt
Total: 148456
Operaciones: 1999367971807
Tiempo: 05:18:22.8275
Total: 148456
Operaciones: 1999367971807
Tiempo: 05:19:19.7308
Total: 148456
Operaciones: 142721756241
Tiempo: 00:23:55.8312
Total: 148456
Operaciones: 1882181047
Tiempo: 00:00:27.5522
Total: 148456
Operations: 178977979
Tiempo: 00:00:03.7508
Total: 148456
Operaciones: 2000000
Tiempo: 00:00:00.0127
alan@chapita:~/Universidad/SSPA/Ejercicios-de-Algoritmia/A02/C++/P04$
```

```
File  Edit  View  Terminal  Tabs  Help
void criba(bool primes[], long max) {
        for(long i = 2; i <= sqrt(max+1); i++)
                if(primes[i])
                        for (long j = i; j*i <= max+1; j++)
                                primes[(i*j)]=0;
}

int main() {
        long number, max = 0;
        long* numbers = new long[N];
        int operations = 0;
        int nPrimes = 0;
        clock_t t;
        string time;

        for(int i = 0; i < N; i++){
                cin >> number;
                if (number > max) max = number;
                numbers[i] = number;
        }

        bool* primes = new bool[max];
        for(long i = 0; i < max; i++) primes[i] = 1;

        criba(primes, max);

f.-CribaEratostenes.cpp                                  57,0-1          67%
```

Conclusión

Ahora he comprobrado que hay formas distintas de atacar un mismo problema, unas soluciones pueden ser eficaces en algunos casos pero hay veces que otros algorimos pueden ser utiles tambien.