Laboratorio Nro. 2: Notación O Grande

Rafael Villegas M.
Universidad Eafit
Medellín, Colombia
rvilegasm@eafit.edu.co

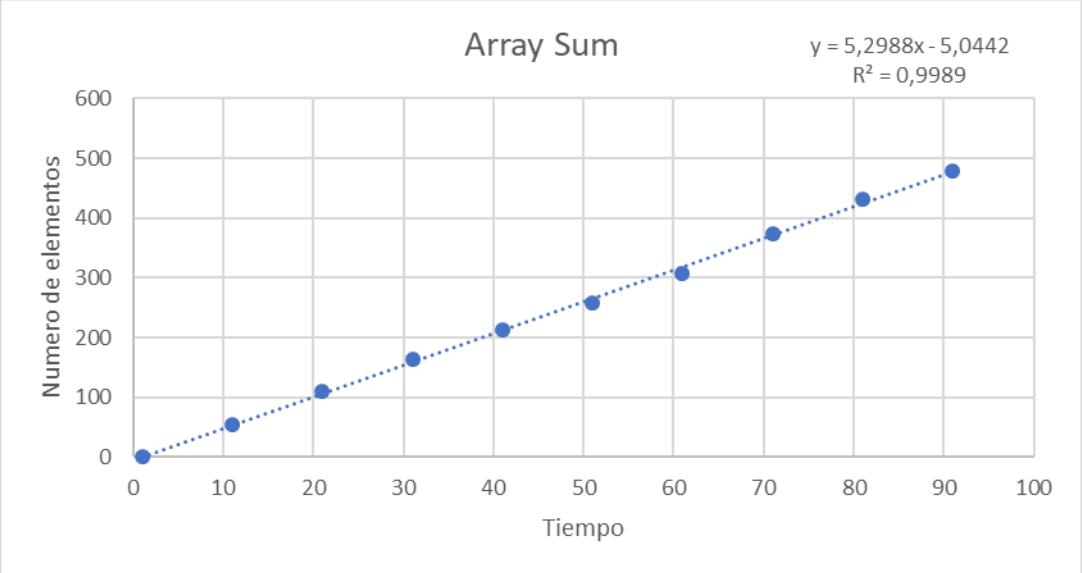
Felipe Cortés J.
Universidad Eafit
Medellín, Colombia
fcortesj@eafit.edu.co

3) Simulacro de preguntas de sustentación de Proyectos

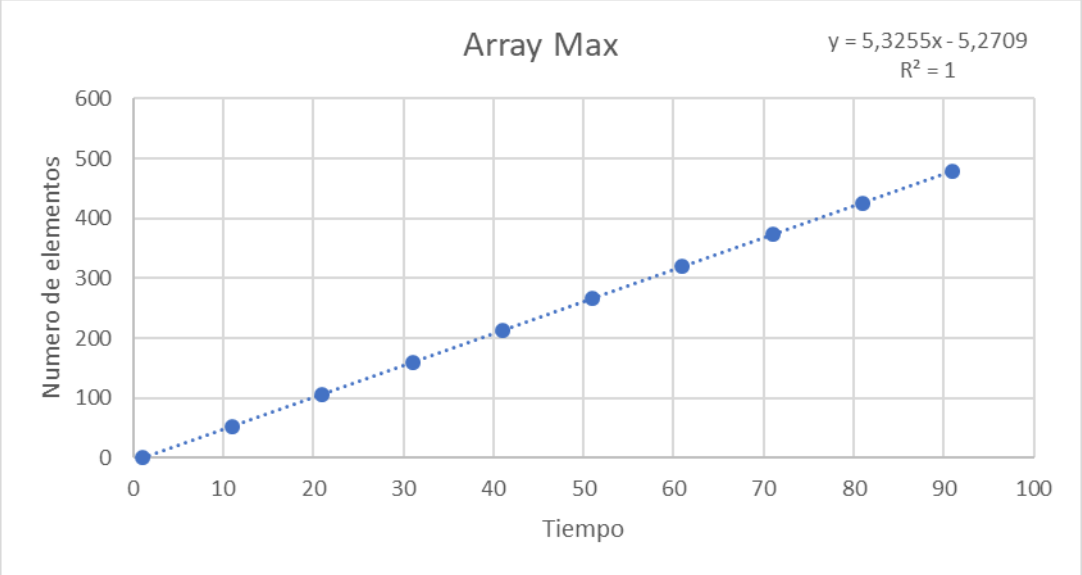
1. Tabla de Datos:

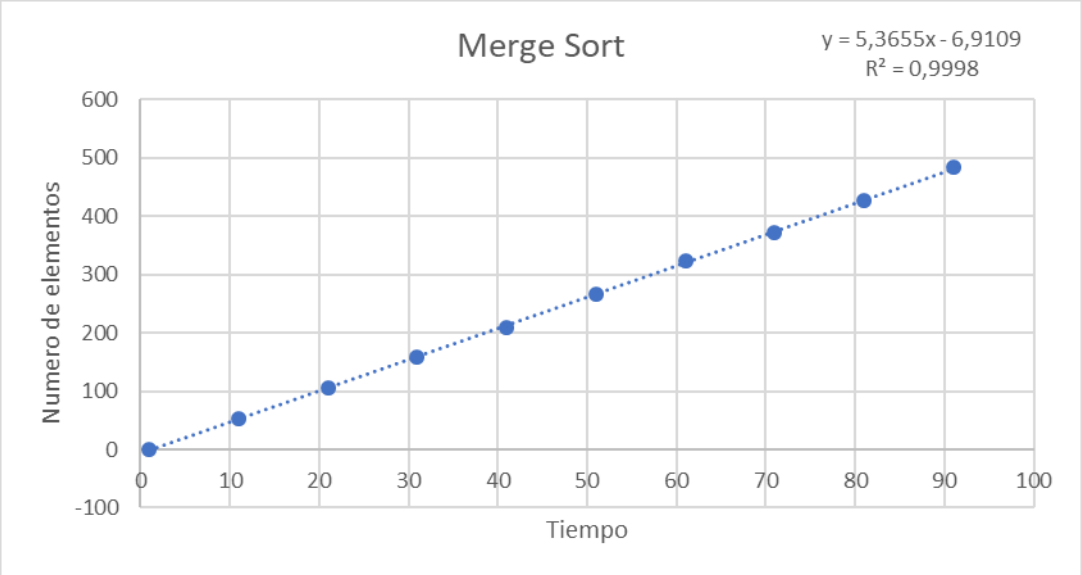
	N=11(N=10100)	N=51(N=10500)	N=71(N=11700)	N=91(N=12000)
ArraySum	54	257	373	478
ArrayMax	53	266	374	479
MergeSort	53	266	371	484
InsertionSort	54	54	68	72

2. Graficas:

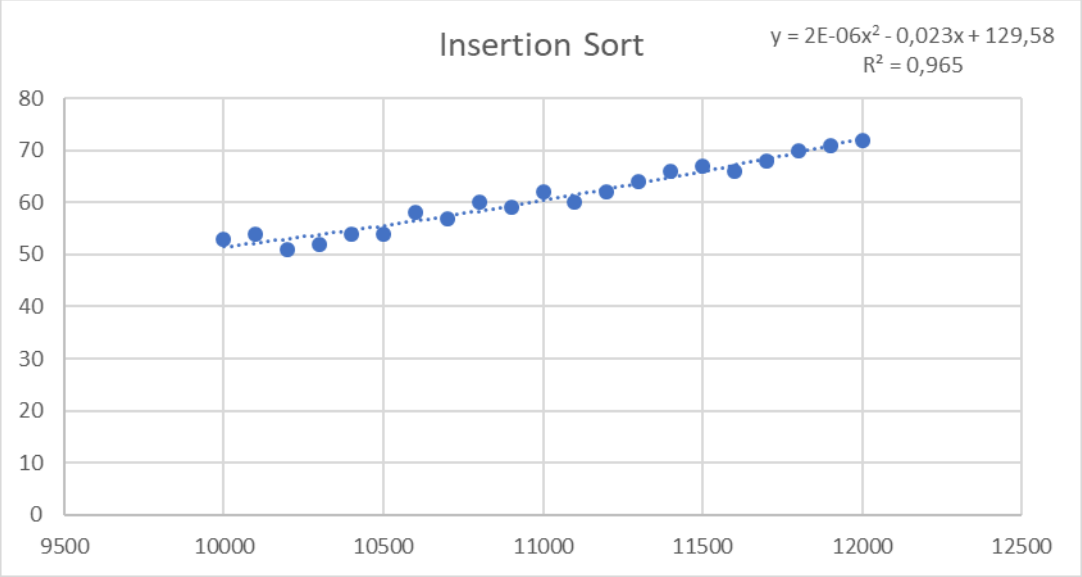


DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co





DOCENTE MAURICIO TORO BERMÚDEZ
Teléfono: (+57) (4) 261 95 00 Ext. 9473. Oficina: 19 - 627
Correo: mtorobe@eafit.edu.co



3. **R/** Los resultados teóricos y los experimentales concuerdan, ya que cada algoritmo actuó de la manera en la que se esperaría al sacarle la complejidad y la notación O correspondiente.

4. **R/** Mientras más grandes sean los valores de N, más tiempo se demorará, incluso más que merge sort, ya que tiene una complejidad de O(n^2), lo que significa que para valores muy grandes cada vez se ejecuta mucho mas lento.

5. **R/** Para valores grandes de N, Array sum sevuelve mas lento, pero de una forma lineal, lo que significa que no se demora tanto como los algoritmos de complejidad O(n^2), como lo es Insertion Sort.

6. **R/** Si lo que se quiere es ordenar un conjunto pequeño de datos, es mas recomendable usar insertion sort, ya que al principio no crece tanto. Pero si se necesita ordenar un conjunto grande de datos, es mejor usar Merge Sort, ya que, debido a su crecimiento lineal, para grandes datos se demorará menos que Insertion Sort, debido a que este último es cuadrático.

7. **R/** maxSpan se encarga de revisar el arreglo de adelante a atrás y de atrás hacia adelante, buscando un valor igual al comienzo y al final del mismo, de forma que la diferencia de posiciones que hay entre ellos, incluyendo el ultimo valor, sea la mayor.

8. **R/** Complejidades:

Array 2
CountEvens: C1 + C2*n -> O(n)
bigDiff: C1 + C2*n -> O(n)
sum13: C1 + C2*n -> O(n)
has22: C1 + C2*n -> O(n)
lucky13: C1 + C2*n -> O(n)

Array 3
maxSpan: C1 + C2*n + C3*(n^2) -> O(n^2)
fix34: C1 + C2*n + C3*(n^2) -> O(n^2)
fix45: C1 + C2*n + C3*(n^2) -> O(n^2)
canBalance: C1 + C2*n + C3*(n^2) -> O(n^2)
linearIn: C1 + C2*n + C3*(n^2) -> O(n^2)

9. **R/** n es la longitud del arreglo. En otras palabras, es la cantidad de elementos que tiene cada arreglo que se analiza.

4) Simulacro de Parcial

1. C
2. D
3. B
4. B
5. D