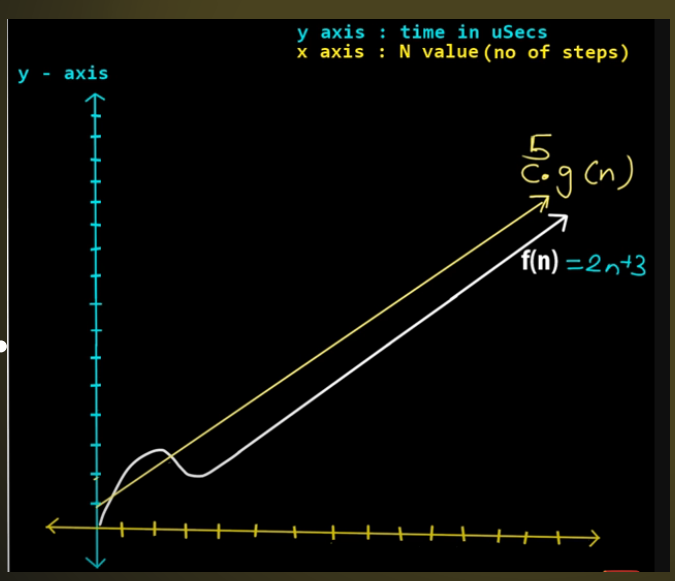
Algoritmos: Notacion Big O, Big Ω & Big θ

Algoritmos

Estudiamos estructuras de datos para aprender a escribir programas mas eficientes, pero por que deben ser mas eficientes cuando las nuevas computadoras cada vez son mas rapidas? la respuesta es que nuestra ambicion crece con nuestras capacidades.
Cada estructura de datos, y cada algoritmo tiene costos y beneficios, los practicantes necesitan entender ampliamente como aprovechar los costos y beneficios para adaptarse a nuevos desafios de diseño. Esto requiere un gran conocimiento en los principios de el analisis de algoritmos.
Relacionado a los costos y beneficios deben de tener en cuenta los intercambios, un ejemplo es, reducur tiempo de requerimientos al costo de mas requerimientos de espacio o vice versa.

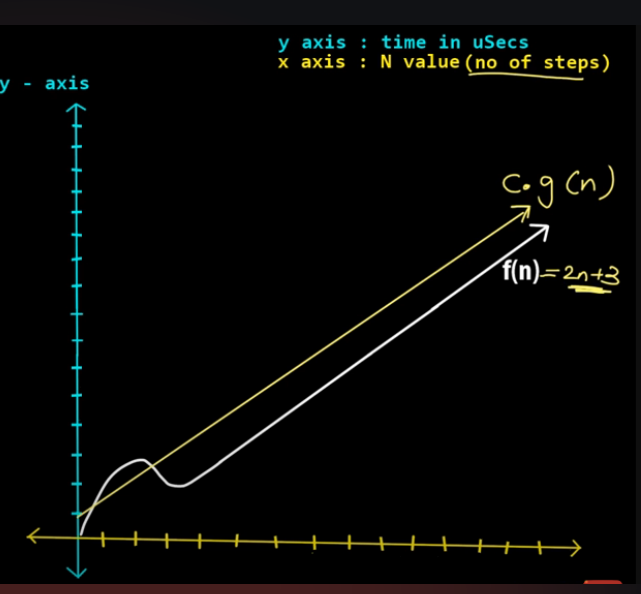
Big Ω Omega

Introducida por Hardy & Littlewood's 1914 memoir.
Este Describe el mejor caso, lo menos que puede tardar un algoritmo en completarse
f(n) = Ω(g(n))
donde n>=n0, c>0, n0>=1



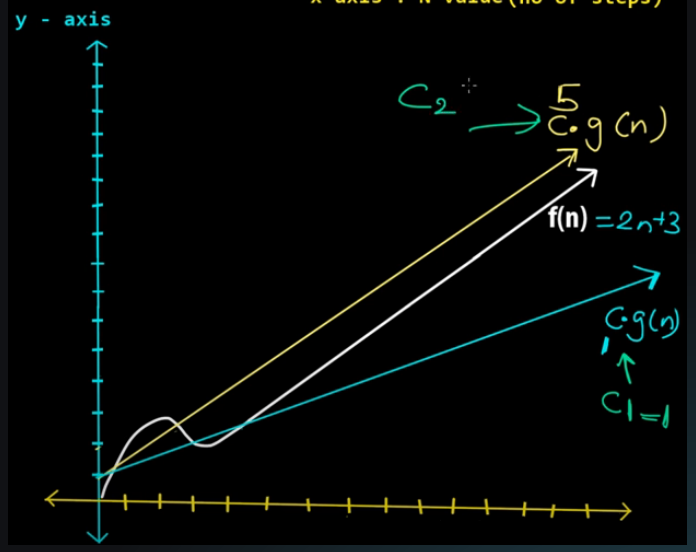
Big O

Creada por Bachmann 1984 Book.
Describe especificamente el peor escenario en tiempo, y que el algoritmo no vaya a pasar de ese tiempo
f(n) = O(g(n))
Condicion: f(n) <= c.g(n)
donde n>=n0, c>0, n0>=1



Big θ Theta

Sugerida por Bob Tarjan and by Mike Paterson.
Describe el escenario promedio nos da el tiempo mas realista de un algoritmo, por que no siempre el algoritmo se va a comportar de la mejor manera ni de la peor
f(n) = θ(g(n))
donde n>=n0, c1, c2>0, n>=n0, n0>=1



Para que sirven?

Todos teienen un uso especifico diferente, pero en general son para que podamos calcular los diferentes tiempos de un algoritmo ya que siempre se comportan diferente debido a diferentes factoeres como la RAM, el CPU y DIsco duro, y entre mas complejo el algoritmo mas recursos va a consumir y mas tiempo va a llevar, como ejemplo un algoritmo para un cohete que va al espacio, estos tienen tiempos limites para hacer cosas y no se puede tardar mas de eso por que pueden terminar en tragedias.
Estos algoritmos nunca dan numeros exactos solo son aproximaciones ya que para un numero exacto se necesesitan calculos mas complejos

Referencias

Shaffer, C. A. (1997). A practical introduction to data structures and algorithm analysis. Upper Saddle River, NJ: Prentice Hall.
Singh, N., & Tiwari, R. G. (2015). Basics of algorithm selection: a review. Int. J. Comput. Sci. Trends Technol, 3, 139-142.
Knuth, D. E. (1976). Big omicron and big omega and big theta. ACM Sigact News, 8(2), 18-24.
Hidary, J. D., & Hidary, J. D. (2021). A brief history of quantum computing. Quantum Computing: An Applied Approach, 15-21.
Curry, D. M., & Dagli, C. H. (2014). Computational complexity measures for many-objective optimization problems. Procedia Computer Science, 36, 185-191.
https://youtu.be/1tfdr1Iv6JA?si=0npmiSD7vM0CGsrg
Levitin, A. (2014). Introduction to the design and analysis of algorithms. Pearson Education.
Kleinberg, J., & Tardos, E. (2022). Algorithm Design (2a ed.). Pearson.
Cormen, T. H., & Leiserson, C. E. (2022). Introduction to Algorithms, fourth edition. MIT Press.
Weiss, M. A. (2005). Data Structures and Algorithm Analysis in C++ (3a ed.). Pearson.
Sedgewick, R. (1983). Algorithms. Addison Wesley Longman Publishing.