Actividad evaluada 5

AE 5 Grupo 9

Tomás Salbach
Diego Ramirez
Martina Matamala
Vicente Avilés

a) O(1):

Un algoritmo que te muestre el elemento 0 de una lista A de largo n > 0. Al acceder únicamente a ese elemento 0, de modo A[0], el algoritmo siempre tendrá el mismo tiempo de ejecución, ya que solo te mostrará un valor. Independiente del tamaño de la lista, el algoritmo siempre se puede acceder al resultado en un único paso. Eso justifica la complejidad de O(1).

b) O(log(n)):

El algoritmo de "Búsqueda binaria", este algoritmo encuentra la posición de un elemento de una lista ordenada. Funciona al dividir repetidamente a la mitad la porción de la lista que podría contener al elemento, hasta reducir las ubicaciones posibles a solo una. (Khan Academy, s.f.). El algoritmo funciona en un tiempo logarítmico, usando al número n de elementos de la lista como argumento del logaritmo. Eso justifica la complejidad de O(log(n)).

c) O(n):

El algoritmo de Kadane, este algoritmo consiste en encontrar la suma máxima de números consecutivos dentro de una matriz unidimensional, o lista (Vega, 2021). Lo que hace el algoritmo es encontrar la suma máxima de la submatriz, cuyo último índice sea la posición en la que nos encontramos. El número de pasos del algoritmo depende directamente del tamaño del problema, es decir, n. Eso significa que tiene complejidad de O(n).

d) O(nlog(n)):

Algoritmo de Merge Sort. Este algoritmo consiste en dividir un vector a ordenar en varias partes, estas partes se ordenan y, posteriormente, se mezclan entre ellas de forma ordenada. (Universidad de Granada, s.f). En cada paso, se divide el vector en dos partes. Esta división ocurre $\log_2(n)$ veces. Los problemas, ahora de tamaño 1, se combinan de nuevo en un arreglo ordenado. La combinación de dos de estos subproblemas es de tamaño n/2. Por esto, y como esas divisiones ocurren en cada paso del algoritmo, el algoritmo es de complejidad O(nlog(n)). (Siendo, en este caso, el logaritmo de base 2) (Khan Academy, s.f.)

e) O(n^2):

Algoritmo de Insert Sort. El Insert Sort u ordenamiento por inserción, itera sobre los índices de una lista. Es decir, de los índices hasta (n-1), siendo n el largo de la lista. En este ordenamiento, cuando se itera, se revisan n veces el índice hasta llegar a n - 1, esto significa que el número de pasos es una serie aritmética, de n * k, siendo k el índice. Usando la fórmula para las series aritméticas, se obtiene que la función del algoritmo puede ser expresada como (n^2)/2 + n/2. (Khan Academy, s.f) Esa función justifica el nivel de complejidad de O(n^2).

f) O(2^n):

El Algoritmo LCS, o Longest common subsequence. El algoritmo busca la subsecuencia de mayor longitud entre dos secuencias. Una de las soluciones, aunque no muy efectiva, es de complejidad O(2^n). Para averiguar una subsecuencia, para cada carácter de la secuencia mayor, se tienen dos opciones, tomar el carácter, o no tomarlo. Lo que esto implica, es que si la longitud de la secuencia es un número n, entonces existen 2^n subsecuencias para esa secuencia. Eso significa, que al iterar por cada subsecuencia de la secuencia, se ocupa ese método, lo que supone una complejidad de O(2^n). (Netlify, s.f)

g) O(n!):

Un algoritmo que genere el número de todas las posibles permutaciones de un conjunto. Usando la fórmula de la permutación, podemos ver que se requiere lo siguiente: calcular de forma n!, n siendo el largo del conjunto. Un ejemplo, sería el conjunto "ABC", que contiene 3 elementos, A, B, y C. La forma de conseguir sus permutaciones, sería de modo 3!, es decir, 6 permutaciones posibles. De este modo, el número de pasos del algoritmo es obligatoriamente de n!, justificando la complejidad de O(n!).

Fuentes:

- El primero fue de elaboración propia.
- https://es.khanacademy.org/computing/computer-science/algorithms/binary-search/a/binary-search
- https://platzi.com/discusiones/1775-poo-python/116368-cual-es-la-complejidad-algoritmica-de-la-busqueda-binaria/
- https://dev.to/anscharivs/el-algoritmo-de-kadane-explicado-4e90
- https://es.khanacademy.org/computing/computer-science/algorithms/merge-sort/a/analysis-of-merge-sort
- https://lsi2.ugr.es/jmantas/ppr/tutoriales/tutorial_mpi.php?tuto=06_mergesort#:~:text=El%20MergeSort%20es%20un%20algoritmo,entre%20ellas%20de%20forma%20ordenada.

- https://es.khanacademy.org/computing/computer-science/algorithms/insertion-sort/a/analysis-of-insertion-sort
- https://longest-common-subsequence.netlify.app/
- g) fue elaboración propia.