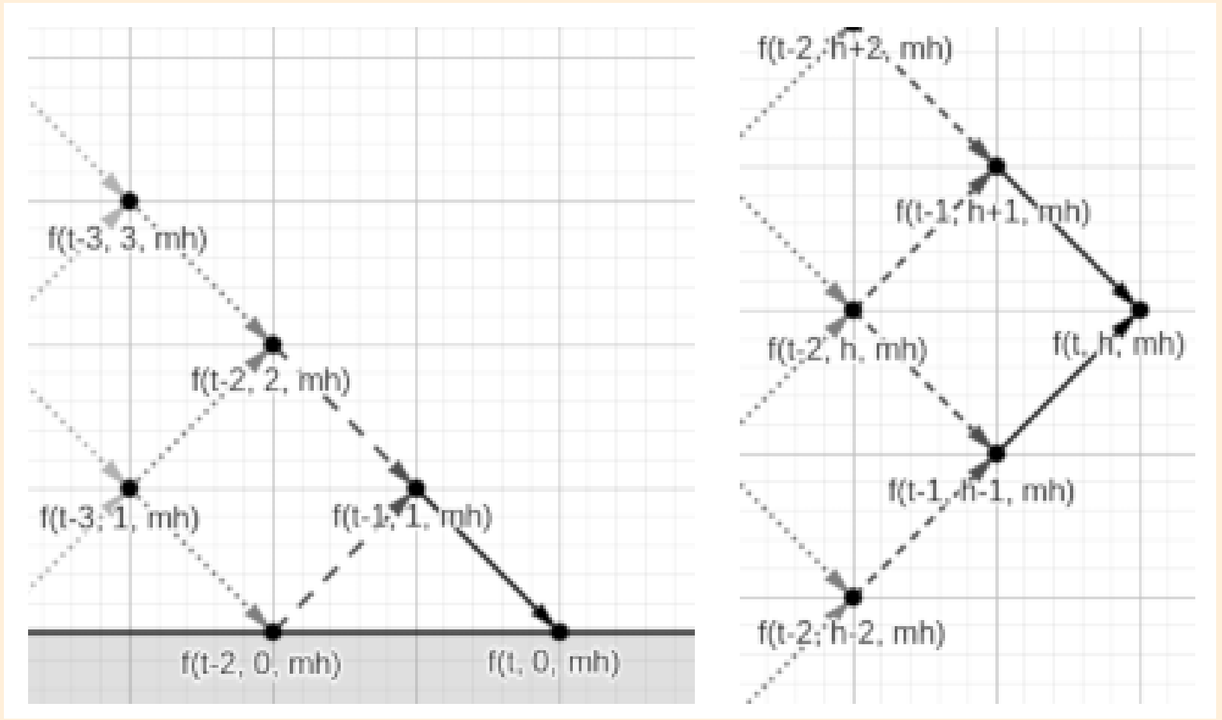
PROYECTO 2

Análisis y Diseño de Algoritmos

Nelson García | Joaquín Puente

Definición del problema

Tenemos una montaña rusa con dos tipos de movimiento en los que cada tramo avanza una unidad de longitud ascendente o descendente y una unidad de longitud horizontalmente, en función de si es un tramo ascendente o descendente.



Solución DP

Para llegar a la solucíón del problema podemos observar que cada estado de la montaña rusa depende del estado anterior donde cada estado se puede representar como (t, ch, mh)

Tenemos 3 casos

- ch = 0
- 0 < ch < mh
- ch = mh

¿Cómo se aplica DP?
Tenemos la matriz tridimensional res_montanias
En la que se almacena el estado a cada estado por lo que se utiliza el estado anterior para determinar el siguiente. De este modo llegando a la solución final en la que se retorna el resultado final el cuál es la cantidad de montañas rusas posibles para L & H
La matriz se llena con el caso base (0,0,0) = 1 iterando sobre los valores para llenar la matriz. (Buttom-up)

Solución DaC

La representación de los estados es muy similar a la implementación de DP.

Para encontrar la solución se hace una recursión hasta llegar a t = 0. Con una logitud máxima de 2L.

¿Cómo se aplica DaC?

Según el caso con el que nos encontremos de representar (t, ch, mh)

Si ch = 0
consideramos
(t-1, 1, mh)

Si 0 < ch < mh
consideramos
(t-1, ch-1, mh)
(t-1, ch+1, mh)

Si ch = mh
consideramos
(t-1, mh-1, mh)
(t-1, mh-1, mh-1)

Los resultados se almacenan para no recalcular innecesariamente algunos cálculos.

Resultados

Dynamic Programing

O(L*H*H) + O (T)

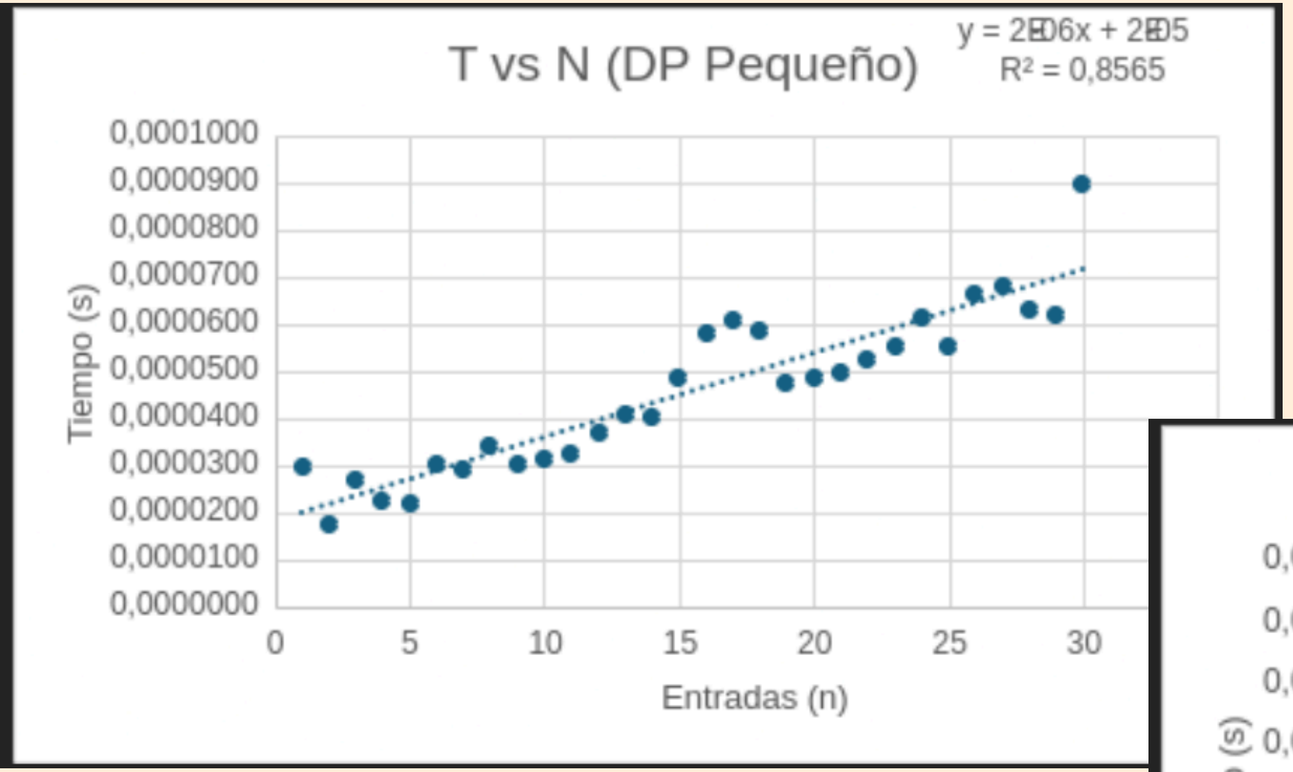
1. Por la construcción de la matriz de forma iterativa de tamaño L*H*H
2. Ya que las soluciones están en una matriz y la respuesta es un acceso a un arreglo entonces tenemos O(1) para el acceso, para T consultas entonces O(T)

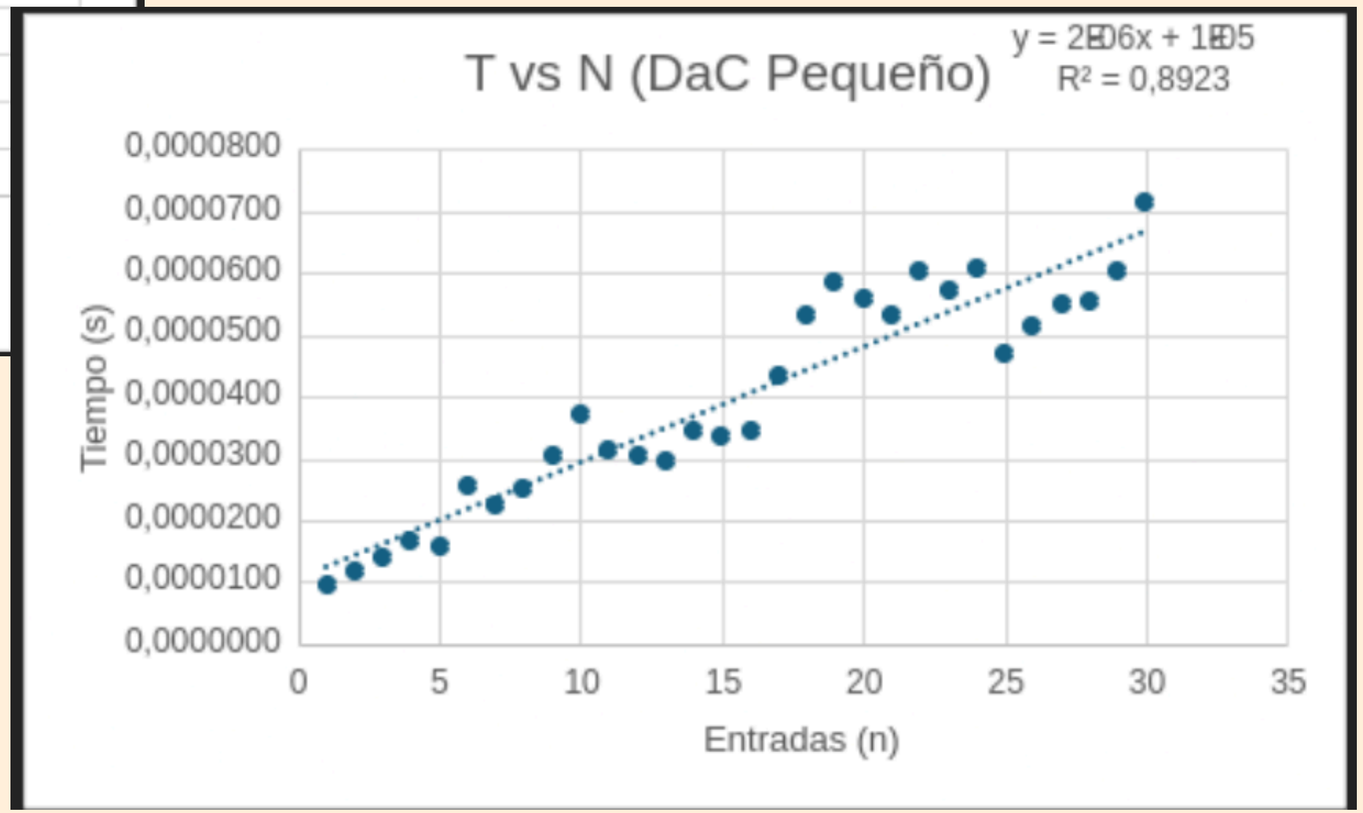
Divide and Conquer

O(L*H*H) + O (T)

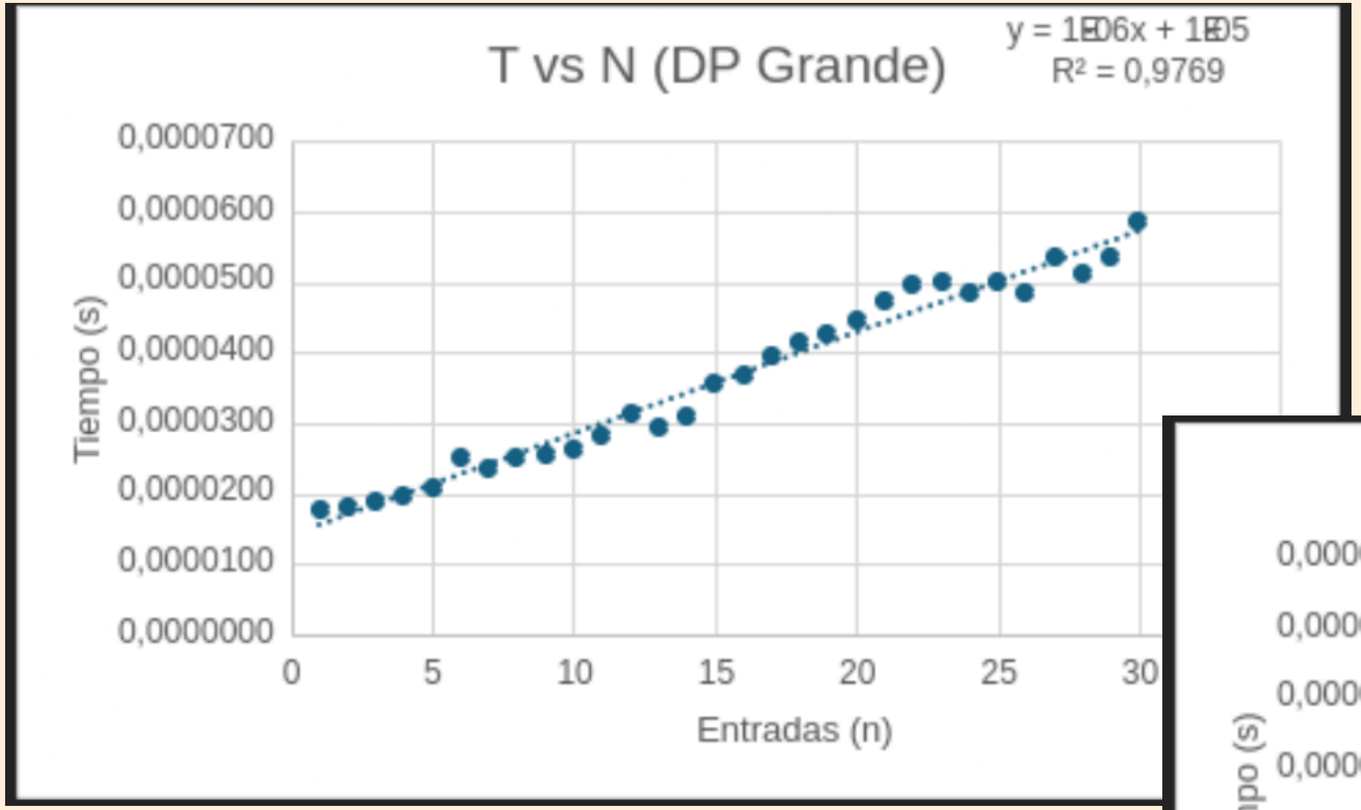
1. Ya que cada subproblema se almacena entonces la complejidad se reduce de $O(4^L)$ a O(2*L*H*H) el cual se puede reducir a O(L*H*H)
2. Ya que las soluciones están en una matriz y la respuesta es un acceso a un arreglo entonces tenemos O(1) para el acceso, para T consultas entonces O(T)

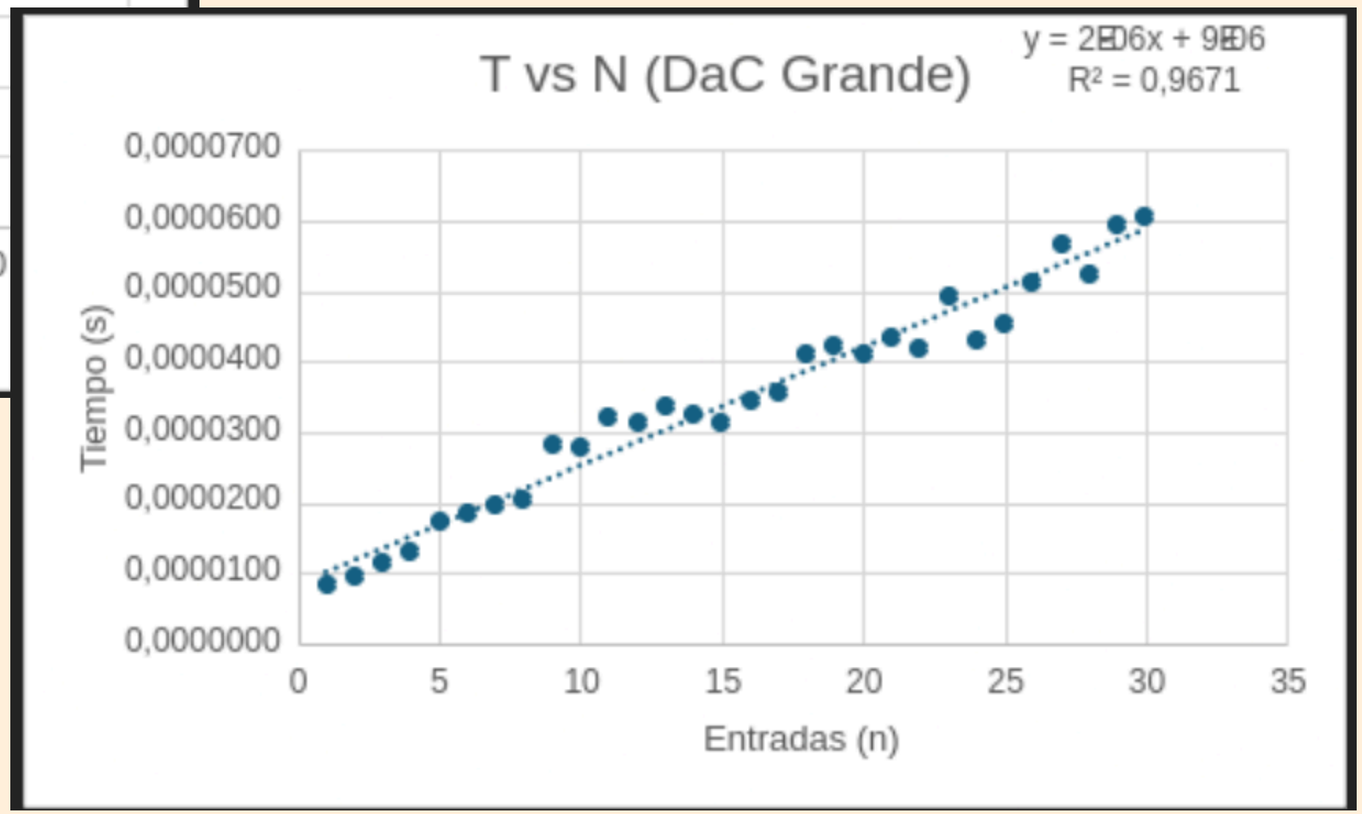
Análisis Empírico





Análisis Empírico





Bibliografía

Programación Dinámica (II): Ejemplos más avanzados | Aprende Programación Competitiva. (n.d.). https://aprende.olimpiada-informatica.org/algoritmia-dinamica-2

Soltys, M. (2018). An Introduction to the Analysis of Algorithms. World Scientific Publishing Company.