Memoria Práctica 2: Divide y vencerás

Alonso del Rincón de la Villa y Alberto Lardiés Getán

17 de abril de 2023

Índice

1. Decisiones de diseño

Descomponemos el problema de encontrar el nodo hoja en el que acaba la bola en k subproblemas siendo k la altura del árbol. Estos subproblemas (todos iguales) consisten en decidir dado un nodo hacia qué nodo hijo irá la bola y llamar recursivamente al algoritmo con el nodo hijo destino hasta el caso base (no hay nodos hijos) que es nuestra solución.

Para el problema de construir el árbol podemos descomponerlo en n problemas siendo n el número de nodos del árbol (y por lo tanto 2^k siendo k la altura). El subproblema es la creación de cada nodo, y el algoritmo que lo resuelve crea el nodo y se llama recursivamente para los nodos hijos del nodo.

2. Pruebas

2.1. Complejidad de los algoritmos

Como ejecutamos el algoritmo para determinar la posición de la bola para m bolas tenemos un total de $m * k$ llamadas a la función (coste $O(m * k)$)

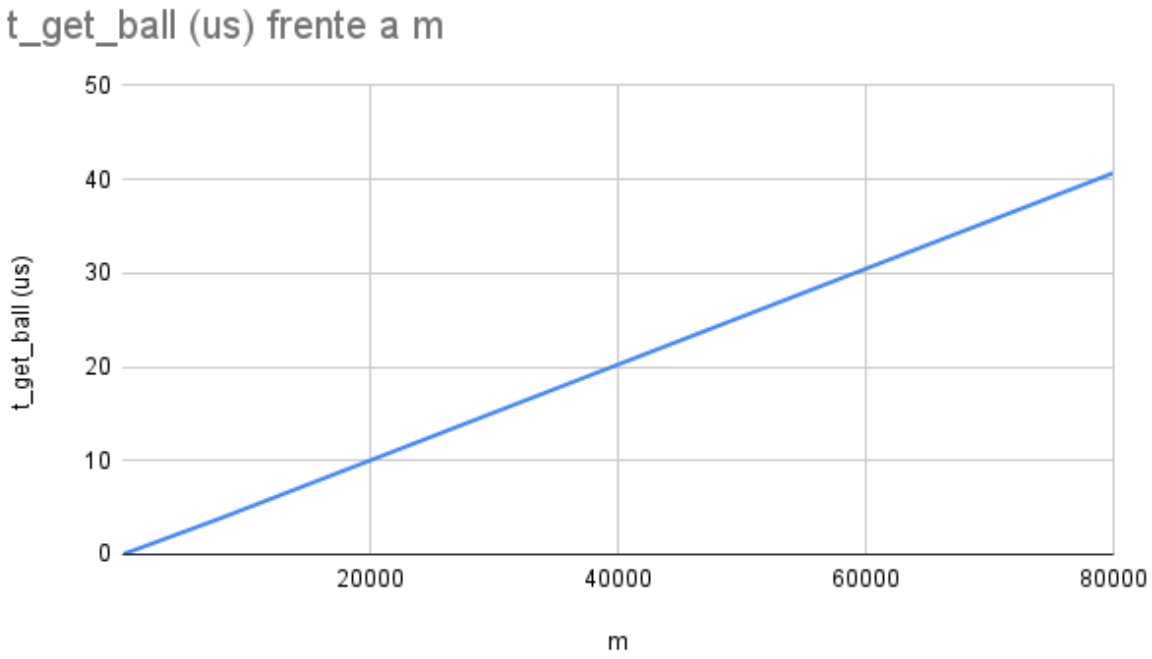
En el caso de la construcción del árbol tenemos 2^k llamadas (coste $O(2^k)$.

2.2. Mediciones y conclusiones

Hemos generado 3 sets de prueba para comprobar el impacto en el rendimiento de las distintas variables:

1. profundidad fija (21) en el que crece linealmente el número de bolas

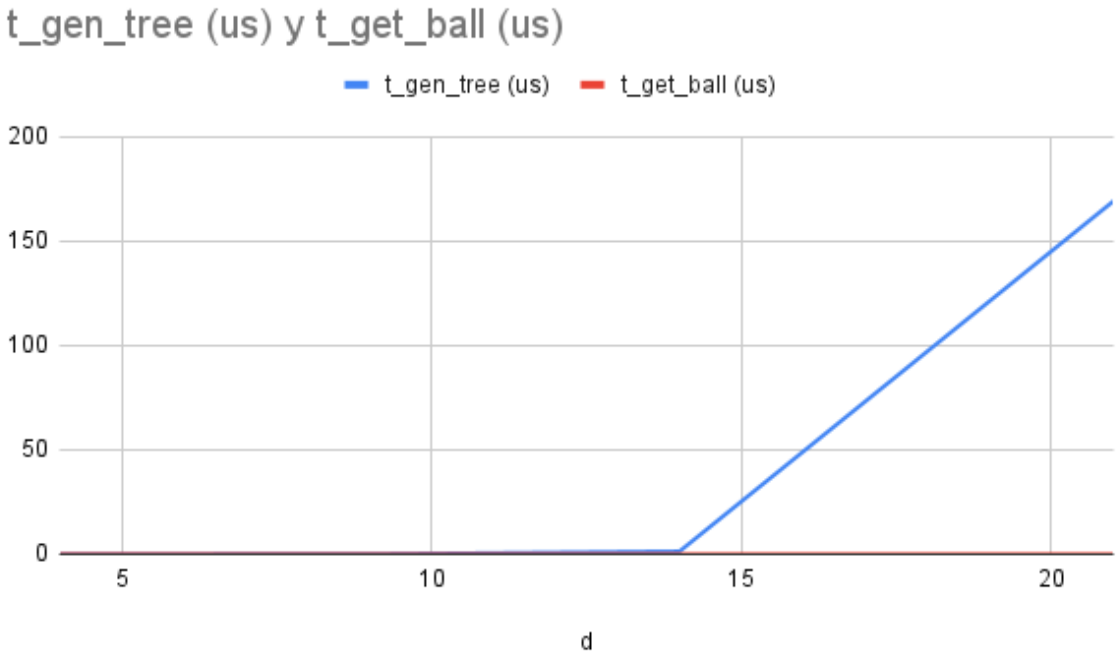
m	t get ball (us)
8	0,00504
80	0,04193
800	0,37161
8000	3,88818
80000	40,6837



No se ha representado el tiempo de generación del árbol porque no depende de m. El incremento es lineal como se esperaba.

2. número de bolas fijo (8) en el que crece linealmente la profundidad del árbol

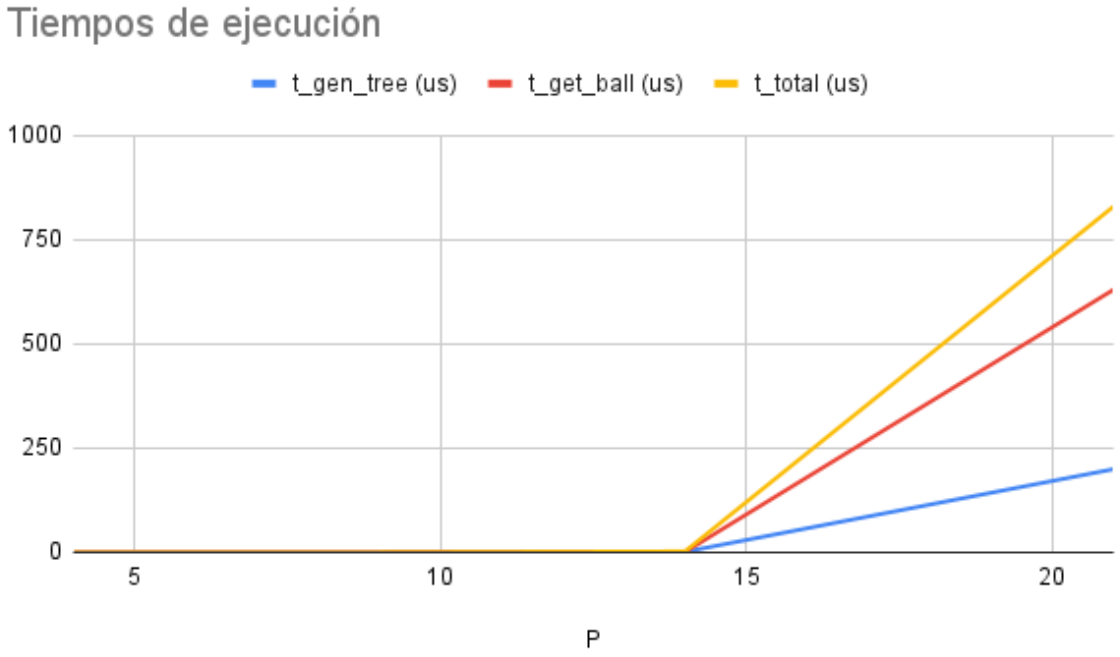
k	t gen tree (us)	t get ball (us)
4	0,002	0,00065
6	0,00571	0,00058
9	0,04305	0,00082
14	1,33063	0,00181
21	169,149	0,00494



En el tiempo de generación se aprecia el incremento exponencial esperado. En el tiempo de obtención de la posición de las bolas se esperaba un incremento lineal. El crecimiento obserbado es claramente muy inferior al de la generación del árbol pero tampoco se puede apreciar que sea lineal. Quizás se deba a que el poco tiempo de ejecución que se necesita para computar 8 bolas hace a la ejecución muy sensible a variables que influyen en la medición de tiempo como el scheduler.

3. profundidad crece linealmente y el número de bolas es el máximo que permite el árbol:

k	t gen tree (us)	t get ball (us)	t total (us)
4	0.001368	0.000571	0.001939
6	0.005402	0.001164	0.006566
9	0.041657	0.015545	0.057202
14	1.38757	1.20559	2.59316
21	199.515	630.012	829.527



En este caso $m = 2^{k-1}$ por lo tanto el coste de obtener la posición de las bolas pasa a ser $O(2^{k-1} * k)$ y por lo tanto debería apreciarse un crecimiento exponencial en ambas mediciones, lo cual reflejan nuestros datos.

El primer caso de prueba del tercer set se corresponde con el árbol del ejemplo del enunciado. Hemos cotejado los resultados de este caso con la simulación que se hace en el enunciado para comprobar la corrección de nuestra implementación.