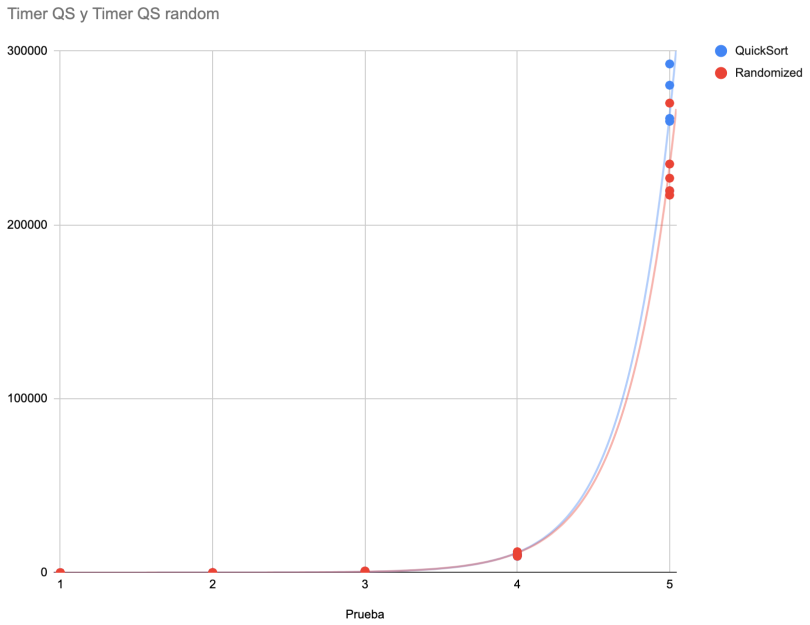
Actividad 4.4 Implementación y análisis de Randomized Quicksort

Jesus Sebastian Jaime Oviedo A01412442 - 18 nov 2022

Análisis de tiempo

	Grupo de prueba	Grupo de prueba	Grupo de prueba
Grupo diez	1	1	0
	1	1	0
	1	1	0
	1	1	0
	1	1	0
Grupo mil	2	68	53
	2	74	60
	2	68	53
	2	69	50
	2	62	76
Grupo mil	3	868	927
	3	822	692
	3	871	828
	3	916	951
	3	866	823
Grupo cien mil	4	10126	10333
	4	10276	9501
	4	9425	9658
	4	10543	12063
	4	11293	10717
Grupo millon	5	219610	270028
	5	261220	234996
	5	259572	219654
	5	292543	217118
	5	280313	226858



Aunque la tendencia no tiene una correlación todavía por el tamaño del muestreo. El tiempo dependerá la función integrada de random, como también que tan balanceado sea nuestro input.

Reporte

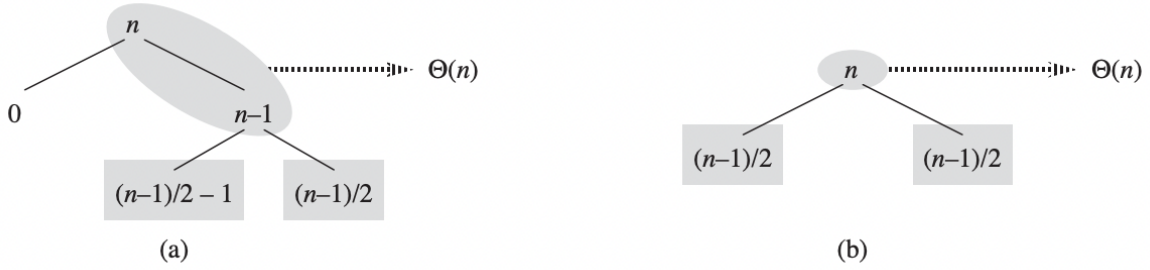
Después de haber entendido la esencia del análisis aleatorio, escribe un reporte donde expliques detalladamente cómo los autores realizaron el análisis y por qué se llegó a la complejidad . En tu reporte responde las siguientes preguntas:

¿Qué diferencia hay con respecto al análisis tradicional del peor caso para Quicksort?

Los autores al analizar el peor de los casos de los quicksort's por medio de la substitución, toman la partición como lo más complejo de la operación. Con el random, O(n log n) es el peor de los casos indistinto si es balanceado o desbalanceado. El quicksort genera un O(n^2), dándonos a entender que la partición que es la función que mayor toma tiempo, cual el random tiene mayor eficiencia en ejecutar.

¿Qué está midiendo el análisis de complejidad?

Que el randomized quicksort se considera el 'caso promedio" al particionar. La combinación el elegir aleatoriamente p o r de una partición buena más una mala, y así sucesivamente, hace que el "divide-and-conquer" se vuelva balanceado. Esto evita el peor caso del quicksort normal, de un subproblema de n-1 elementos y el otro de 0, que incrementa al máximo el running time.



¿Qué ventajas tiene el Randomized Quicksort con respecto al Quicksort?

Que al hacer el divide-and-conquer de manera aleatoria, considera la naturaleza real que los datos pueden estar desbalanceados y sean distintos entre sí. También que gesta un análisis más simple. El running time de ambos quicksort es dominado por la partición.

¿En qué aplicaciones es más práctico cada uno de los dos algoritmos?

Es práctico el uso del quicksort normal es para pocos datos y que conozcamos que por su naturaleza tienden a no estar balanceados.

Por la naturaleza de la realidad: la mayoría de los casos y al tener masivas cantidades de información que ordenar, se prefiere el random quicksort porque es igual o mejor el tiempo de procesamiento.