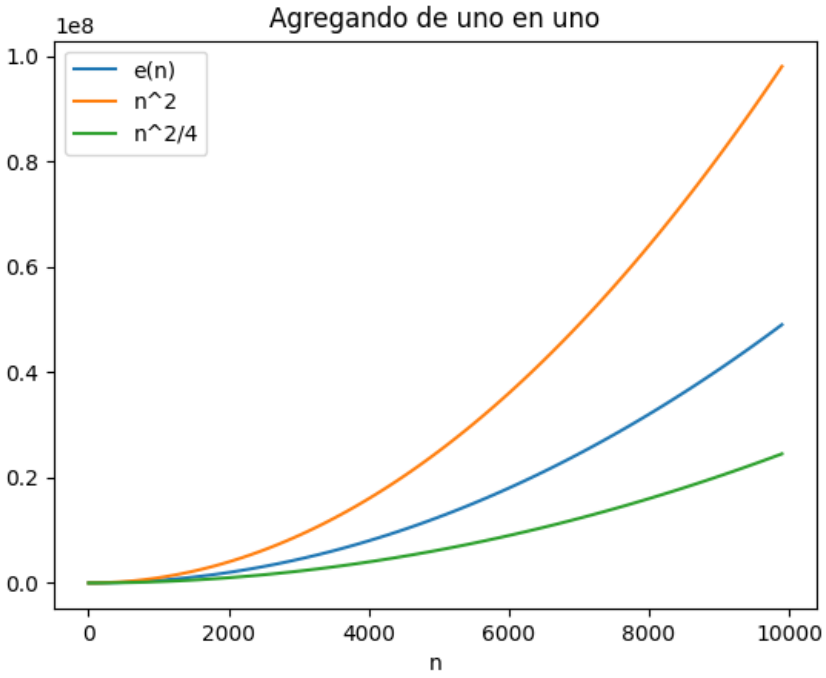
Análisis de Algoritmos e introducción a Matemáticas Discretas
Tarea 1

Rubén Pérez Palacios Lic. Computación Matemática
Profesor: Dr. Carlos Segura González

18 de agosto de 2021

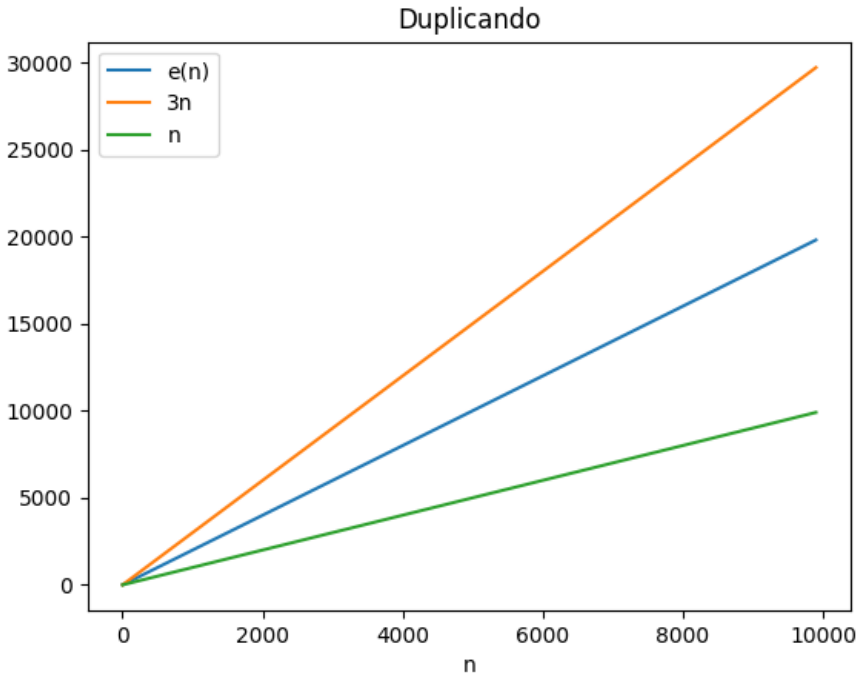
Problema 1 (2 puntos). En las sesiones de clases, se explicó que cuando en el vector de la stl se hace un push_back, y no hay memoria suficiente para albergar otro elemento, se pide memoria adicional, y se procede a copiar todos los elementos que ya se tenían en el vector a las nuevas posiciones de memoria. Supongamos que inicialmente tenemos un vector vacío, con espacio reservado para un elemento, y que se van a realizar $n = 2^p$ inserciones.

a) Determine el número de escrituras en memoria que se realizarán si cada vez que el vector se queda sin memoria, se pide memoria para un espacio adicional. Especifique el número de escrituras $e(n)$ de forma exacta, grafíquelo, y encuentre una función $g(n)$ lo más simple posible, de forma que $e(n) = \Theta(g(n))$.
Debido a que cada vez que se agrega un elemento al vector se copia todo el vector a otro vector cuyo tamaño crece en 1, entonces al insertar el $i - esimo$ elemento se harán i escrituras, por lo tanto el número de escrituras en memoria que se realizarán son $e(n) = \sum_{i=1}^{n} i = \frac{(n)(n+1)}{2} = \frac{(2^p)(2^p+1)}{2}$, la cual es $e(n) = \Theta(n^2) = \Theta(4^p)$ y cuya gráfica es:



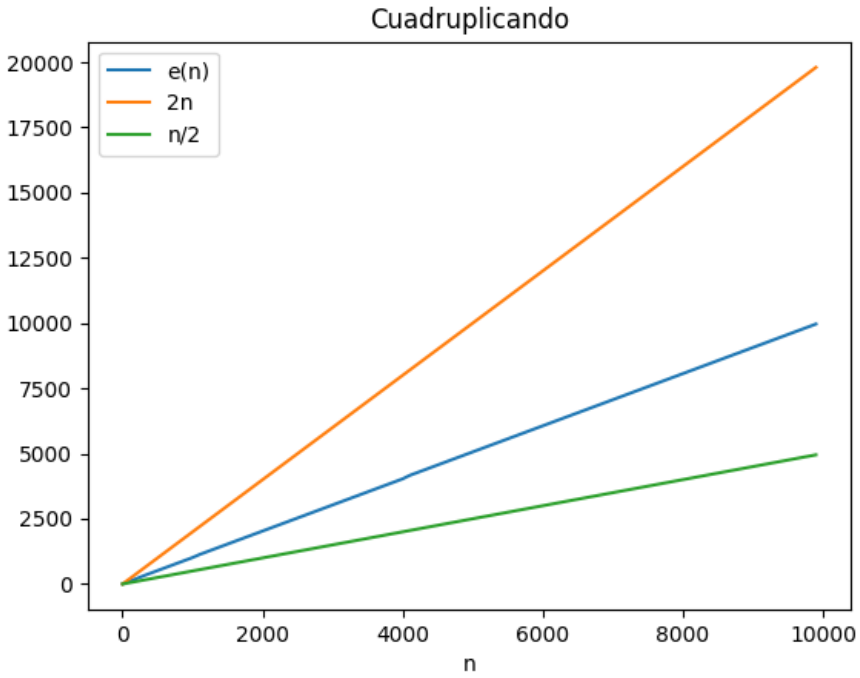
b) Determine el número de escrituras en memoria que se realizarán si cada vez que el vector se queda sin memoria, se duplica la cantidad de espacio reservado. Especifique el número de escrituras $e(n)$ de forma exacta, grafíquelo, y encuentre una función $g(n)$ lo más simple posible, de forma que $e(n) = \theta(g(n))$.

Primero para cada elemento que se inserte se tenga que aumentar el tamaño o no del vector tendras que escribir este, por lo que se harán al menos n acciones. Ahora el vector necesitara aumentar su tamaño cada que se inserte un $i - esimo$ elemento donde i es de la forma $2^k + 1$ para algún k entero, y en cada una de estas se harán 2^k escrituras por lo que el total de escrituras para las copias es de $\sum_{i=0}^{p-1} 2^i = 2^p - 1 = n - 1$. Por lo tanto el número de escrituras en memoria que se realizarán son $e(n) = 2n - 1$, la cual es $e(n) = \Theta(n)$ y cuya gráfica es:



c) Determine el número de escrituras en memoria que se realizarán si cada vez que el vector se queda sin memoria, se multiplica por 4 la cantidad de espacio. Especifique el número de escrituras $e(n)$ de forma exacta, grafíquelo, y encuentre una función $g(n)$ lo más simple posible, de forma que $e(n) = \theta(g(n))$.

Primero para cada elemento se tenga que aumentar el tamaño o no del vector se tendra que escribir este, por lo que se harán al menos n acciones. Ahora el vector necesitara aumentar su tamaño cada que se inserte un $i - esimo$ elemento donde i es de la forma $2^{2k} + 1$ para algún k entero, y en cada una de estas se harán 2^{2k} escrituras por lo que el total de escrituras para las copias es de $\sum_{i=0}^{\lfloor \frac{p}{2} \rfloor - 1} 2^i = 2^{\lfloor \frac{p}{2} \rfloor} - 1 \sim \sqrt{n} - 1$. Por lo tanto el número de escrituras en memoria que se realizarán son $e(n) = n + 2^{\lfloor \frac{p}{2} \rfloor} - 1 \sim n + \sqrt{n} - 1$, la cual es $e(n) = \Theta(n)$ y cuya gráfica es:

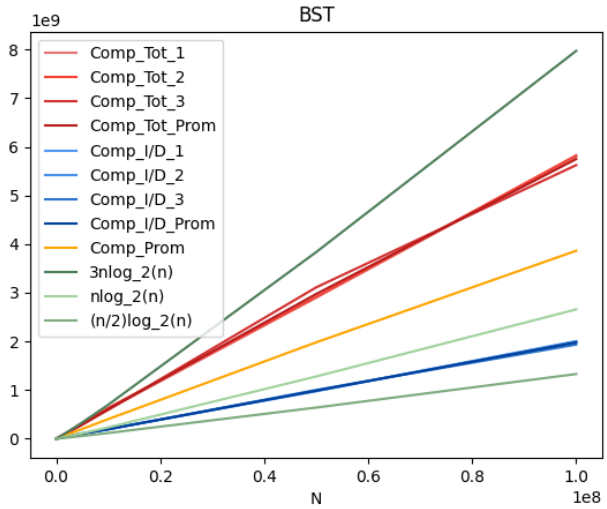


Cabe mencionar que el aumentar la cantidad de espacio que se reserva cada vez que el vector se quede sin memoria para almacenar apartir del doble del tamaño de este no será optimo en cuanto a memoria, a pesar de que esto reduce la cantidad de copias que tendremos que hacer de todas formas tenemos que escribir en memoria cada uno de los elementos ingresados al vector por lo que tenemos que hacer al menos N escrituras, pero si incrementa la cantidad de memoria reservada.

Problema 2 (6 puntos: 3 de implementación, 3 de análisis). Implemente un árbol binario de búsqueda, sin balancear, que permita insertar números enteros. Realice N operaciones de inserción, insertando números en el rango $[1, 10^{18}]$ generados de forma aleatoria. Si el número ya estaba insertado, no lo reinserte.

a) Determine, para el caso peor, cuál es el número de comparaciones entre números llevadas a cabo y expréselo usando notación θ. El número de comparaciones se refiere al total después de la N inserciones. No hace falta demostrar.
El peor caso es cuando ingresas los datos ordenados de mayor a menor o de menor a mayor, ya que así para cada inserción se aumentará en uno el alto del arbol, y el número de comparaciones esta dado por $\theta(N^2)$.

b) De forma experimental, cuente cuántas comparaciones se hace y grafíquelo para valores de N hasta 10^8. Nótese que dependerá de los valores aleatorios generados, así que ejecútelo varias veces y analice los datos en la forma que considere más oportuna. Encuentre una función que se ajuste de forma adecuada a los datos generados, y trate de explicar qué es lo que está ocurriendo, apoyándose de medidas experimentales sobre el árbol que se está generando, describiendo su intuición sobre lo que está ocurriendo e incluyendo fórmulas que apoyen a su intuición.
Notemos que la altura mínima del BST puede tener es $\lceil \log_2(n) \rceil$ cuando este es un árbol completo, y su máximo es cuando todo nodo tiene a lo más un hijo donde su altura $h = n$, es decir $\lceil \log_2(n) \rceil \leq h \leq n$. Son poco los casos que necesitaran hacer un número de comparaciones muy cercano al del peor caso, puesto que la cantidad de vectores de entrada cuyos datos son ordenados (de mayor a menor, o de menor a mayor) son pocos con respecto a la cantidad de vectores posibles, ya que las entradas fueron generadas con *grand*() (que trata de simular una distribución uniforme entre tmin y tmax). Ahora nos interesaría saber como están dsitribuidas la cantidad de vectores que les toman k comparaciones generar el BST, y ver en donde están mas condensadas para saber que es lo más probable que tome en construir nuestro BST. Por lo datos obtenidos podemos ver que los vectores generados hacen cerca de $nlog(n)$ comparaciones, esta medida se uso puesto que por como se generaron los datos es muy posible que la altura del arbol este más cerca de $\lceil \log_2(n) \rceil$ que de n. Nota para las Comparaciones I/D solo se tomo en cuenta la comparación para encontrar de que lado se inserta el nuevo nodo, en cambio en totales se conto cuando se regunta si ya estaba el valor y/o si ya llegaste al destino de este.



Problema 3 (2 puntos). Demuestre por reducción al absurdo que para todo entero $k \geq 1$, $n^{k-1} = o\left(n^k\right)$.

Demostración. Supongamos que para todo $c > 0$ existe un N tal que para todo $n \geq N$ se cumple que $n^{k-1} \leq cn^k$, en especial $\frac{1}{N} \leq c$, lo cual es una contradicción puesto que c es un Real y para propiedad arquimediana o completitud de estos siempre habra un número d tal que $0 \leq d \leq \frac{1}{N}$. □