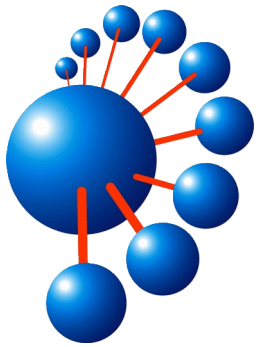


Tarea 1
Análisis de Algoritmos

Profesora: Andrea Rodríguez.
Integrantes: Diego Varas.
Jeremías Torres.

Abril 17, 2017

Introducción

Éste problema consiste en construir y recorrer árboles binarios. Dada una secuencia de árboles binarios se debe imprimir un recorrido de orden de nivel de cada árbol, para ello cada nodo de los árboles consiste en un entero positivo y su trayectoria, por ejemplo, (11,LLL), además cabe destacar que tienen menos de 256 nodos.

En un recorrido de orden de nivel de un árbol los datos se imprimen de izquierda a derecha en cada nivel, es decir, en el nivel k se imprimen antes que k+1.

Para desarrollar el problema se sabe que el nodo raíz será de la forma (n,) y se termina la entrada con un ().

Se considera árbol binario completo si cada nodo tiene un solo valor y en caso de que no sea un árbol completo se deberá imprimir “no completo”.

Finalmente se pide demostrar que el algoritmo usado para resolver el problema es correcto, también analizar el costo asintótico y hacer una evaluación experimental en base a entradas variables de tamaño.

Desarrollo

Para resolver el problema dado utilizamos el algoritmo BFS para así recorrer el árbol. Además usamos estructuras de datos como queue y vector.

Este algoritmo lo implementamos de la siguiente manera:

BFS()

Begin
cola q;
encolar nodo raíz en q;
vector ans;
Nodo Nodo;

while cola q no vacía **do**
Nodo = frente de cola;
desencolar q;
apilar por atrás en vector ans el valor de Nodo;

if hijo izquierdo de Nodo es distinto de nulo **do**
encolar hijo izquierdo en q;
end if

if hijo derecho de Nodo es distinto de nulo **do**
encolar hijo derecho en q;
end if
end while
end

Demostración:

Para demostrar que este algoritmo es correcto utilizamos el método del bucle invariante:

Begin
cola q;
encolar nodo raíz en q;
vector ans;
Nodo Nodo;

while cola q no vacía **do**

Invariante: Existe un nodo hijo del vértice padre.

Nodo = frente de cola;
desencolar q;
apilar por atrás en vector ans el valor de Nodo;

si hijo izquierdo de Nodo es distinto de nulo **do**
encolar hijo izquierdo en q;
end if

si hijo derecho de Nodo es distinto de nulo **do**
encolar hijo derecho en q;
end if
end while
end

- Asumimos que al comenzar en el bucle **while** tenemos la cola q no vacía. En ella estará el nodo raíz del árbol que será el nodo padre inicial.

- Luego, antes del primer **if**, se guarda el valor de la raíz en un Nodo y se elimina el nodo padre de la cola, para luego guardar el valor en el vector ans. Luego en los dos **if** siguientes se encolan los nodos izquierdo y derecho del nodo padre (Nodo) en la cola q. Aquí termina la primera iteración y se observa que dentro del loop se cambian datos, pero la cola q seguirá teniendo los nodos hijos de la raíz. Finalmente, antes de comenzar la siguiente iteración se chequea la condición del **while (**q no vacía**)**, por lo que la invariante sigue siendo verdadera.

** Añadiendo que en c++ queue nos da la ventaja de que es FIFO, por lo que al ir encolando los hijos se irán analizando los nodos de izquierda a derecha (como lo indica el problema).

- Luego en las siguientes iteraciones la cola q va a tener los nuevos valores padres y se seguirán guardando los nodos hijos en q hasta que ya no existan hijos.

- Notar que cada vez que se elimina un nodo padre de la cola q, el valor de éstos se irán guardando en el vector ans.

Costo Asintótico:

En BFS las operaciones de encolar y desencolar toman O(1), por lo que el tiempo total dedicado a estas operaciones es O(N) con N la cantidad de nodos existentes en el árbol.

Se tiene que cada vértice adyacente a los vértices padre (de turno) se encolan solo cuando son adyacentes a él, por lo que esto tomará O(m) donde m será la cantidad de nodos hijos de cada nodo padre.

Por lo tanto, el costo asintótico de nuestro algoritmo será de O(n+m).

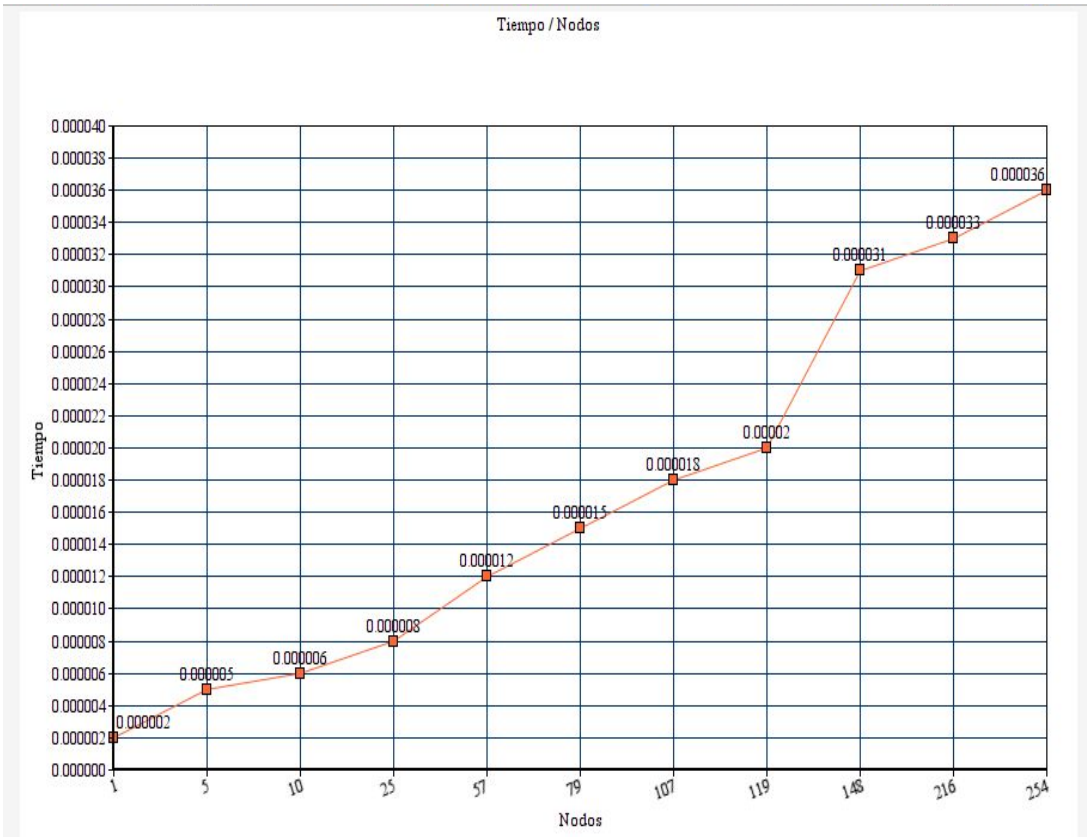
Begin
cola q;
encolar nodo raíz en q;
vector ans;
Nodo Nodo;

while cola q no vacía **do**
Nodo = frente de cola; O(1)
desencolar q; O(1)
apilar por atrás en vector ans el valor de Nodo; O(1)

if hijo izquierdo de Nodo es distinto de nulo **do**
encolar hijo izquierdo en q; O(1)
end if

if hijo derecho de Nodo es distinto de nulo **do**
encolar hijo derecho en q; O(1)
end if
end while
end

Evaluación Experimental:



Experimentalmente los datos arrojados por la medición de tiempo del algoritmo BFS implementado en nuestro código se asemejan al Análisis Teórico, ya que la curva del gráfico tiende a ser O(n), ya que el tiempo de ejecución (medido en segundos) aumentaba a razón de la cantidad de Nodos analizados por el algoritmo.

Conclusión

En este trabajo pudimos poner en práctica la demostración de que los algoritmos son correctos, investigar y aprender diferentes técnicas para lograrlo. Además pudimos ver que el coste teórico se aproximaba bastante del coste experimental y que podían variar drásticamente en algunos casos más específicos.