Universidad de Concepción

Facultad de ingeniería
Departamento de ingeniería informática y Ciencias de la computación

Tarea 1 de Análisis de Algoritmos.

Lisette Morales.

Pablo Venegas.

Profesor : Andrea Rodríguez.

Índice

1. Problema asignado:

Dado un árbol binario y un número k, elimine todos los nodos que se encuentran solo en la (s) ruta (s) de raíz a hoja de longitud menor que k. Si un nodo X se encuentra en múltiples rutas raíz a hoja y si alguna de las rutas tiene una longitud de ruta ¿= k, entonces X no se borrará del árbol binario. En otras palabras, un nodo se elimina si todas las rutas que lo atraviesan tienen longitudes menores que k.

2. Algoritmo

Algoritmo cuya función es recorrer un BST de manera recursiva, una vez que termina de recorrer un camino, vuelve a subir retornando el largo del camino de los nodos, luego guarda el mas largo entre los caminos de sus hijos

```
int marcar(int nodo,int recorrido){
int aux,guardar = recorrido;
for(int i = 0; i < arbol[nodo].size();i++){
aux = marcar(arbol[nodo][i],recorrido+1);
if(aux > guardar) guardar = aux;
}
if(guardar > distancia[nodo])distancia[nodo] = guardar;
return guardar;
}
```

```
function marcar(nodo, recorrido)
INPUT: Dos enteros que representan el nodo del un árbol a analizar y el recorrido que
Output: Retorna un entero que representa la distancia a la que esta una hoja de la rai

begin
declare aux
initialize guardar = recorrido
for i = 0 to size of arbol[nodo]
aux = marcar(arbol[nodo][i], recorrido + 1)
if aux > guardar
guardar = aux
end if
increment i in 1
end for
if guardar > distancia[nodo]
distancia[nodo] = guardar
end if
return guardar
end
```

3. Análisis del Algoritmo

3.1. Evaluación experimental

Gráfico

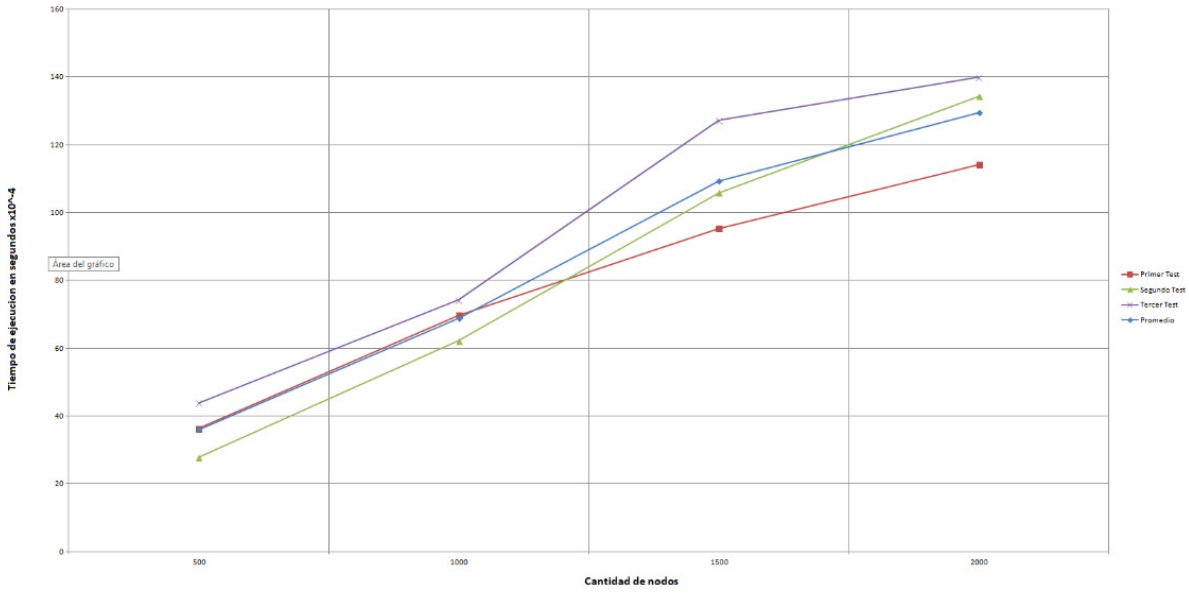

Figura 1: En el caso promedio, nuestra implementación refleja una curva logaritmica de datos vs tiempo.

Tabla de datos obtenidos

Cantidad de nodos	Primer test [x10^-4]	Segundo Test [x10^-4]	Tercer Test [x10^-4]	Promedio [x10^-4]
500	36,28	27,75	43,9	35,97666667
1000	69,75	62,12	74,23	68,7
1500	95,29	105,77	127,19	109,4166667
2000	114,15	134,35	139,9	129,4666667

Figura 2: Se realizaron tres ejecuciones de código por cantidad de nodos modificando k.

3.2. Análisis asintótico

Implementación recursiva, con Big-Oh Olog(n)
Demostración: Se deduce para un árbol de m niveles

```
Árbol de nivel m:
Altura A = m + 1
Nodos n = 2^A - 1
Hojas =  2^m
Nodos internos = n - 2^m
```

De la expresion para el número de nodos, se despeja A, en efecto A=log(n+1)=O(log n)

En un árbol binario, cada vez que se sigue una trayectoria por un determinado subárbol, se descarta la mitad de los nodos. Para árboles desbalanceados se obtiene el mismo analisis O(log n)

3.3. Análisis de correctitud

Precondición: si el BTS tiene al menos 1 elemento guardara el recorrido.
Postcondición: guardar la ruta mayor o igual a k.
Base inducción: Si de (3) se afirma que si nodo = 1, quiere decir que el árbol tiene un solo nodo y cumpliendo la condicion de igualdad del problema, el algoritmo guarda el recorrido.

hipótesis de inducción: Dado que se obtiene un BST de tamaño n+1 y un nodo de (3) se afirma que se recorre el árbol obteniendo el nodo y guardando su recorrido mas largo, en (5) se afirma que guarda el camino mas largo entre los caminos de sus hijos. En efecto, si ocurre para un n+1 tambien será para el caso de tamaño n.