**Strutture dati utilizzate**

**TreeArrayList (**Aggiunta del campo **Distanza** alla classe **Node)**

**Descrizione algoritmo**

**Altro punto di forza dell’algritmo:** Per ogni nodo, non ho necessità di calcolare il numero di nodi “precedenti”. Aggiungendo il campo **distanza** alla classe Node, eseguendo una visita generica partendo dalla radice dell’albero, per ogni “sottolivello” incremento il valore del campo **Distanza**, in modo da avere in automatico il numero di nodi “all’indietro”, cioè il numero di nodi tramite i quali posso raggiungere il nodo stesso. Confrontando questo valore nella chiamata successiva, ottengo il numero di volte che il nodo risulta medio.

**(Punti di forza dell’algoritmo:** Se l’altezza del nodo dalla radice è uguale alla distanza dal sotto nodo che sto considerando, mi fermo. Questo mi permette, fino al nodo ad altezza , di controllare solamente un numero di, circa, sotto nodi, invece di doverli controllare tutti quanti.)

**Analisi tempo teorico**

Nodo radice (getRoot):

* in quanto devo considerare tutti gli archi
* in quanto devo controllare tutti i nodi nel caso peggiore

Visita generica (mediumNode):

Visita generica a partire da ogni singolo nodo (calculateMediumValue): **)**

Tempo totale:

**Analisi tempo sperimentale**

L’analisi sperimentale conferma l’analisi teorica e mostra come l’utilizzo delle Liste di incidenza e Liste di adiacenza sia una scelta migliore rispetto alle Matrice di adiacenza.