**Ingegneria degli Algoritmi – AA 2017 2018 – Prima Prova in Itinere**

**Studente:** Foderaro Salvatore

**Numero di Matricola:** 0214381

**Problema - Binary search tree con lazy deletion**

### Descrizione algoritmo

Per l’implementazione del dizionario, ho dapprima definito la funzione **lazyInsert**, identica alla funzione **insert** già presente nel dizionario degli alberi binari, con l’unica differenza nell’aggiunta del terzo campo nella lista delle informazioni del nodo, impostato di default a 0. Ho definito questo terzo elemento della lista come **Valore di eliminazione del nodo**; se uguale a **1**, il nodo risulta eliminato, se uguale a **0** no.

Ho inoltre definito le seguenti funzioni aggiuntive:

* **deletionValue(nodo):** restituisce il valore di eliminazione del nodo
* **setDeleted(nodo):** segna il nodo come eliminato, andando ad impostare ad 1 il valore di eliminazione
* **setUnDeleted(nodo):** segna il nodo come non eliminato, andando ad impostare a 0 il valore di eliminazione
* **isDeletedOr(nodo):** restituisce True se il nodo risulta eliminato, altrimenti False
* **setValue(nodo,valore**): permette di modificare il valore del nodo

Per la risoluzione del problema ho definito le funzioni richieste:

1. **lazySearch(chiave):** cerco il nodo associato alla chiave e controllo se il suo valore di eliminazione è 0; in caso positivo, restituisco il nodo, altrimenti restituisco **Nome**
2. **boolDeleteLazy(chiave):** tramite **lazySearch** cerco il nodo associato alla chiave. Se la funzione chiamata restituisce il nodo, quindi quest’ultimo è presente nell’albero e non è segnato come eliminato, allora segno il nodo come eliminato e restituisco True. Altrimenti, nel caso in cui **lazySearch** restituisce None, dunque il nodo non è presente nell’albero o è presente ma risulta come eliminato, restituisco False
3. **insertLazyControl(chiave,valore):** prima di inserire la coppia **(chiave, valore)** nell’albero, controllo se esiste già un nodo associato alla chiave che voglio inserire ed il suo valore di eliminazione. Se il nodo è già presente ed il suo valore di eliminazione è **1**, imposto il valore passato alla funzione come nuovo valore del nodo ed il valore di eliminazione a **0**. Altrimenti se nell’albero non è presente nessun nodo associato alla chiave, o presente ma con valore di eliminazione **0**, effettuo il normale inserimento tramite **lazyInsert**

### Utilizzo Lazy Deletion

L’uso della **Lazy Deletion** è preferibile nei casi in cui dopo l’eliminazione di un nodo è molto probabile il suo reinserimento, oppure quando si vuole indicare un nodo come “disabilitato”. Utilizzando la Lazy Deletion, *invece di rimuovere il nodo e doverlo successivamente reinserire*, basta segnarlo come eliminato. Al suo reinserimento l’operazione mi costerà solo l’aggiornamento dell’informazione relativa all’eliminazione, ed al massimo quella relativa al suo valore.

### Tempo di esecuzione teorico