Albero binario di ricerca

Un albero binario si dice albero binario di ricerca (***ABR***) se:

1. Presa la radice, il dato contenuto nella radice è maggiore o uguale dei dati contenuti nel sottoalbero sinistro e minore o uguale dei dati contenuti nel sottoalbero destro;

$$T \to sx \to key \leq T \to key \leq T \to dx \to key$$

2. A loro volta, i sottoalberi sono alberi binari di ricerca.

Caso base: $T = \emptyset$
Caso induttivo: $T \neq \emptyset$, in cui T_{sx} e T_{dx} sono a loro volta ABR.

Ricerca in un ABR

La ricerca di un elemento k in un ABR è più semplice di una ricerca su un albero binario, perché sfruttando le proprietà degli ABR riduciamo notevolmente il numero di nodi da controllare.

> **Important**
>
> La complessità della ricerca di un elemento in un ABR è lineare sull'altezza dell'albero; La ricerca impiega quindi $O(h)$.

Ricorsiva

```
def SearchABR(T, k):
        ret = T
        if T != NULL:
                        if T->key > k
                                ret = SearchABR(T->sx, k)
                        else if T->key < k:
                                ret = SearchABR(T->dx, k)
        return ret
```

Iterativa

```
def SearchABR_I(T, k):
        tmp = T
                while tmp != NULL && tmp->key != k:
                        if tmp->key < k:
                                tmp = tmp-> dx
                        else
```

```
                                tmp = tmp ->sx
        return tmp
```

Elemento minimo di un ABR

L'elemento minimo in un ABR è contenuto nel nodo più a sinistra senza figli a sinistra.

```
def Min(T):
        ret = T
        if T->sx != NULL:
                T = T->sx
        return ret
```

```
def MinI(T):
        if T == NULL
                return NULL
        ret = T
        while ret->sx != NULL
                ret = ret->sx
        return ret
```

Elemento massimo di un ABR

L'elemento massimo in un ABR è contenuto nel nodo più a destra senza figli a destra.

```
def Max(T):
        ret = T
        if T->dx != NULL:
                T = T->dx
        return ret
```

```
def MaxI(T):
        if T == NULL
                return NULL
        ret = T
        while ret->dx != NULL
                ret = ret->dx
        return ret
```

Successore di un nodo

Per trovare il successore di un nodo k in un ABR bisogna analizzare vari casi:
Caso base: $T = \emptyset$
Caso induttivo: $T \neq \emptyset$, allora si possono verificare 3 casi:

1. $T \rightarrow key = k$, quindi il successore sarà il minimo del sottoalbero destro: $Min(T \rightarrow dx)$;
2. $T \rightarrow key < k$, quindi devo cercare il successore nel sottoalbero destro: $Succ(T \rightarrow dx, k)$;
3. $T \rightarrow key > k$, quindi posso avere due casi:
    1. il successore è nel sottoalbero sinistro: $Succ(T \rightarrow sx, k)$;
    2. il successore è la radice dell'albero.

```
def Succ(T, k):
        ret = NULL
        if T != NULL:
                if T-> key == k:
                        ret = Min(T->dx)
                else if T->key < k:
                        ret = Succ(T->dx, k)
                else
                        ret = Succ(T->sx, k)
                        if ret = NULL:
                                ret = T
        return ret
```

```
def SuccI(T, k):
        tmp = T
        ret = NULL
        while tmp != NULL && tmp->key != k
                if tmp->key < k
                        tmp = tmp->dx
                else
                        ret = tmp
                        tmp = tmp ->sx
        if tmp != NULL && tmp->dx != NULL
                ret = Min(T->dx)
        return ret
```

Cancellazione di un nodo

Quando eliminiamo un nodo da un ABR dobbiamo assicurarci che la proprietà di ABR valga anche dopo l'eliminazione, quindi abbiamo bisogno di distinguere vari casi:

- Il nodo da eliminare è una foglia, $T \to sx = T \to dx = NULL$, quindi si può eliminare il nodo senza problemi.
- Il nodo da eliminare ha figli a sinistra, $T \to sx \neq NULL$, $T \to dx = NULL$, quindi dobbiamo tenere traccia del figlio sinistro e collegarlo all'albero dopo l'eliminazione.
- Il nodo da eliminare ha figli a destra, $T \to sx = NULL$, $T \to dx \neq NULL$,quindi dobbiamo tenere traccia del figlio destro e collegarlo all'albero dopo l'eliminazione.
- Il nodo da eliminare ha figli a destra e sinistra, $T \to sx = T \to dx \neq NULL$, quindi dobbiamo eliminare la radice; per eliminarla dobbiamo prima trovare il minimo del sottoalbero destro da sostituire in radice, scambiare le chiavi dei nodi, ed eliminare il nodo foglia.

Inoltre, abbiamo bisogno di 3 funzioni:

1. $Delete(T, k)$: trova il nodo da eliminare e richiama $DeleteRoot(T)$ sul sottoalbero radicato in k; restituisce la radice dell'albero.
2. $DeleteRoot(T, k)$: controlla se il nodo ha figli (casi precedenti), elimina il nodo e restituisce la radice del sottoalbero rimanente dopo l'eliminazione.
3. $StaccaMin(T, P)$: trova il minimo del sottoalbero destro e lo stacca dall'albero.

```
def Delete(T, k):
        ret = T
        if T != NULL:
                if T->key < k:
                        T->dx = Delete(T->dx, k) # collego il padre al
nuovo figlio dx
                else if T->key > k:
                        T->sx = Delete(T->sx, k) # collego il padre al
nuovo figlio sx
                else
                        ret = DeleteRoot(T, k)   # elimino il nodo
        return T
```

```
def DeleteRoot(T):
        ret = T
        if T != NULL:
                tmp = T    # creo un puntatore al nodo da eliminare
                if T->sx == NULL:
                        ret = T->dx   # salvo il figlio destro in ret
                else if T->dx == NULL:
                        ret = T->sx   # salvo il figlio sinistro in ret
                else
                        tmp = StaccaMin(T->dx, T)  # salvo in tmp il
minimo di T->dx
                        T->key = tmp->Key          # scambio le chiavi
                Dealloca(tmp)                  # elimino tmp
```

```
        return ret    # ritorno il sottoalbero sottostante al nodo
eliminato
```

```
def StaccaMin(T, P):
        ret = T
        if T != NULL:
                ret = StaccaMin(T->sx, T) # trovo il minimo
                if ret == NULL: # se richiamo staccaMin su una foglia ret
= nodo foglia
                        ret = T
                        if P != NULL:
                                if P->sx == T: # se il minimo si trova a
sx, salvo i suoi figli
                                        P->sx = T->dx
                                else:  # se il minimo si trova a dx, salvo
i suoi figli
                                        P->dx = T->dx
        return ret  # ritorno il minimo
```
