Albero binario di ricerca e operazioni di lettura

Albero in cui il valore di ogni nodo è maggiore o uguale al valore di ogni nodo nel sottoalbero sinistro e minore o uguale al valore di ogni nodo nel sottoalbero destro.

Non è richiesto che l'albero sia "quasi pieno" come per un heap, quindi la rappresentazione sotto forma di array è potenzialmente inefficiente in termini di spazio richiesto rispetto a quello effettivamente occupato.

Stampa in ordine

Visita simmetrica. Costo $\Theta(n)$.

Ricerca

TREE-SEARCH(x, k)

```
1  if x == NIL or k == x.key
2        return x
3  if k < x.key
4        return TREE-SEARCH(x.left, k)
5  else return TREE-SEARCH(x.right, k)
```

I nodi attraversati formano un cammino semplice dalla radice verso le foglie, quindi il tempo di esecuzione è $O(h)$ con h l'altezza dell'albero.

Minimo e massimo

TREE-MAXIMUM(x)

```
1  while x.left ≠ NIL
2        x = x.left
3  return x
```

Con $x.right$ si ottiene il massimo. $O(h)$.

Successore

TREE-SUCCESSOR(x)

```
1  if x.right ≠ NIL
2        return TREE-MINIMUM(x.right)
3  y = x.parent
4  while y ≠ NIL and x == y.right
5        x = y
6        y = y.parent
7  return y
```

Percorre un cammino semplice ascendente o discendente, quindi $O(h)$.