Go: verificare la simmetria di un albero binario

Un albero binario è una struttura dati comune in informatica utilizzata per organizzare dati in una struttura ad albero. Uno degli aspetti interessanti degli alberi binari è la loro simmetria. Un albero binario è considerato simmetrico quando il sottoalbero sinistro di ogni nodo è speculare al sottoalbero destro rispetto alla radice. In questo articolo, esploreremo come verificare se un albero binario è simmetrico utilizzando il linguaggio di programmazione Go.

Rappresentazione di un Albero Binario in Go

Prima di iniziare a verificare la simmetria di un albero binario, è importante comprendere come rappresentare un albero binario in Go. Una delle rappresentazioni comuni è utilizzare una struttura con campi per il valore del nodo, il sottoalbero sinistro e il sottoalbero destro. Ecco come potrebbe apparire questa struttura:

```
type TreeNode struct {
    Val   int
    Left  *TreeNode
    Right *TreeNode
}
```

Ogni nodo contiene un valore (di solito un numero intero) e due puntatori che rappresentano i sottoalberi sinistro e destro. Ora che abbiamo una rappresentazione dell'albero binario, possiamo procedere con la verifica della simmetria.

Verifica della Simmetria di un Albero Binario

La verifica della simmetria di un albero binario può essere fatta utilizzando una visita in profondità (depth-first search) dell'albero. L'idea principale è confrontare il sottoalbero sinistro di un nodo con il sottoalbero destro del nodo corrispondente. Se entrambi i sottoalberi sono nulli o se i valori dei nodi corrispondenti sono uguali e i sottoalberi sinistri e destri sono simmetrici, l'albero è simmetrico.

Ecco una funzione Go che verifica la simmetria di un albero binario:

```
func isSymmetric(root *TreeNode) bool {
    if root == nil {
        return true // Un albero vuoto è simmetrico
    }
    return isMirror(root.Left, root.Right)
}

func isMirror(left *TreeNode, right *TreeNode) bool {
    if left == nil && right == nil {
        return true // Entrambi i sottoalberi sono
vuoti, quindi sono simmetrici
    }
    if left == nil || right == nil {
        return false // Solo uno dei sottoalberi è
vuoto, quindi non sono simmetrici
    }
    return (left.Val == right.Val) &&
        isMirror(left.Left, right.Right) &&
        isMirror(left.Right, right.Left)
}
```

La funzione `isSymmetric` verifica se l'albero binario passato come argomento è simmetrico. La funzione ausiliaria `isMirror` verifica la simmetria dei sottoalberi sinistro e destro di un nodo. Questo approccio ricorsivo esamina tutti i nodi dell'albero e restituisce true se l'albero è simmetrico, altrimenti restituisce false.

Ecco un esempio di come utilizzare la funzione `isSymmetric` con un albero binario:

```
func main() {
    // Creare un albero binario simmetrico
    root := &TreeNode{Val: 1}
    root.Left = &TreeNode{Val: 2}
    root.Right = &TreeNode{Val: 2}
    root.Left.Left = &TreeNode{Val: 3}
    root.Left.Right = &TreeNode{Val: 4}
    root.Right.Left = &TreeNode{Val: 4}
    root.Right.Right = &TreeNode{Val: 3}

    isSymmetric := isSymmetric(root)
    if isSymmetric {
        fmt.Println("L'albero è simmetrico.")
    } else {
        fmt.Println("L'albero non è simmetrico.")
    }
}
```

In questo esempio, l'albero binario creato è simmetrico, e quindi il programma stamperà "L'albero è simmetrico."

Ora hai una solida comprensione di come verificare la simmetria di un albero binario utilizzando il linguaggio di programmazione Go. Puoi

applicare questo concetto per risolvere problemi che coinvolgono alberi binari simmetrici in situazioni reali.