Globalübung

10. Juli 2018

Problem

Erläutern Sie, wie man in einem ungerichteten Graphen G in linearer Zeit herausfinden kann, ob G kreisfrei ist. Begründen Sie die Korrektheit des Verfahrens.

Lösung

Wir können einfach eine Tiefensuche durchführen und dann sehen, ob eine Rückwärtskante existiert. Der Graph ist genau dann kreisfrei, wenn es keine Rückwärtskante gibt.

Wenn es tatsächlich eine Rückwärtskante gibt, dann gibt es auch einen Kreis, da die beiden Endknoten der Rückwärtskante durch einen aus Baumkanten bestehenden Pfad verbunden sind. Umgekehrt kann es ohne Rückwärtskanten keinen Kreis geben, denn bei einem ungerichteten Graphen gibt es nur Baum- und Rückwärtskanten. Ohne Rückwärtskanten gibt es also nur Baumkanten und der ganze Graph ist ein Wald und damit kreisfrei.

Problem

Entwerfen Sie einen Algorithmus, der testet ob ein binärer Baum ein binärer Suchbaum ist.
Schreiben Sie ihn in Pseudocode oder einer vernünftigen Programmiersprache nieder.

Lösung

```
boolean isBinarySearchtree(Searchtree<K, D> tree) {
  Searchtreenode<K, D> min = tree.root, max = tree.root;
  while(min.left ≠ null) min = min.left;
  while(max.right ≠ null) max = max.right;
  return isBinarySearchtree(tree.root, min.key, max.key);
}

boolean isBinarySearchtree(Searchtreenode<K, D> node, K min, K max) {
  if(node == null) {
    return true;
  }
  if(node.key.compareTo(min) ≤ 0 || node.key.compareTo(max) ≥ 0) {
    return false;
  }
  return isBinarySearchtree(node.left, min, node.key)
      && isBinarySearchtree(node.right, node.key, max);
}
```

Problem

Gegeben sei ein Integer-Array der Größe n, welches jede Zahl von 1 bis m genau einmal enthält, wobei $m \leq n$. Die restlichen $n - m$ Positionen sind mit 0 gefüllt.
Entwerfen Sie einen Algorithmus, der ein gegebenes solches Array in $O(n)$ Schritten sortiert.

Lösung

```
void orderArray(int[] a) {
  int numZeros = 0;
  for(int i = 0; i < a.length; i++) {
    if(a[i] == 0) {
      numZeros++;
    }
  }

  int c = 0;
  while(c < a.length) {
    if(a[c] == 0 || c == a[c] + numZeros - 1) {
      c++;
      continue;
    }
    if(a[c] < 0 || a[c] + numZeros - 1 ≥ a.length) {
      throw new RuntimeException("Value in Array outside"
                                 + "of allowed range.");
    }
    if(a[c] == a[a[c] + numZeros - 1]) {
      throw new RuntimeException("Array contains " + a[c]
                                 + " more than once.");
    }
    int temp = a[c];
    a[c] = a[temp + numZeros - 1];
    a[temp + numZeros - 1] = temp;
  }
}
```