Tiefensuche

Ablauf

Zuerst wird ein Startknoten u ausgewählt. Von diesem Knoten aus wird nun die erste Kante (u,v) betrachtet und getestet, ob der gegenüberliegende Knoten v schon entdeckt wurde bzw. das gesuchte Element ist. Ist dies noch nicht der Fall, so wird rekursiv für diesen Knoten die Tiefensuche aufgerufen, wodurch wieder der erste Nachfolger dieses Knotens expandiert wird. Diese Art der Suche wird solange fortgesetzt, bis das gesuchte Element entweder gefunden wurde oder die Suche bei einer Senke im Graph angekommen ist und somit keine weiteren Nachfolgeknoten mehr untersuchen kann. An dieser Stelle kehrt der Algorithmus nun zum zuletzt expandierten Knoten u zurück und untersucht den nächsten Nachfolger des Knotens. Sollte es hier keine weiteren Nachfolger mehr geben, geht der Algorithmus wieder Schritt für Schritt zum jeweiligen Vorgänger zurück und versucht es dort erneut.

Algorithmus

1. Bestimme den Knoten an dem die Suche beginnen soll
2. Expandiere den Knoten und speichere alle Nachfolger in einem Stack
3. Rufe rekursiv für jeden der Knoten in dem Stack DFS (depth first search oder Tiefensuche) auf

Falls der Stack leer sein sollte, tue nichts
Falls das gesuchte Element gefunden worden sein sollte, brich die Suche ab und liefere ein Ergebnis

Beispiel: Java

Quellcode

```
1.  void tiefensuche(Graph g) {
2.  Set<String> visited = new TreeSet<String>();
3.  for (Iterator<String> it = g.nodeIterator(); it.hasNext(); ) {
4.  String n = it.next();
5.  if (!visited.contains(n))
6.  tiefensucheRek(g, n, visited);
7.  }
8.  }
9.  void tiefensucheRek(Graph g, String node, Set<String> visited) {
11. visited.add(node);
12. action(node);
13. for (Iterator<String> it = g.edgeIterator(node); it.hasNext(); ) {
14. String expanded = it.next();
15. if (!visited.contains(expanded)) {
16. visited.add(expanded);
17. tiefensucheRek(g, expanded, visited);
18. }
19. }
20. }
```

Alles anzeigen