

Traversierungen

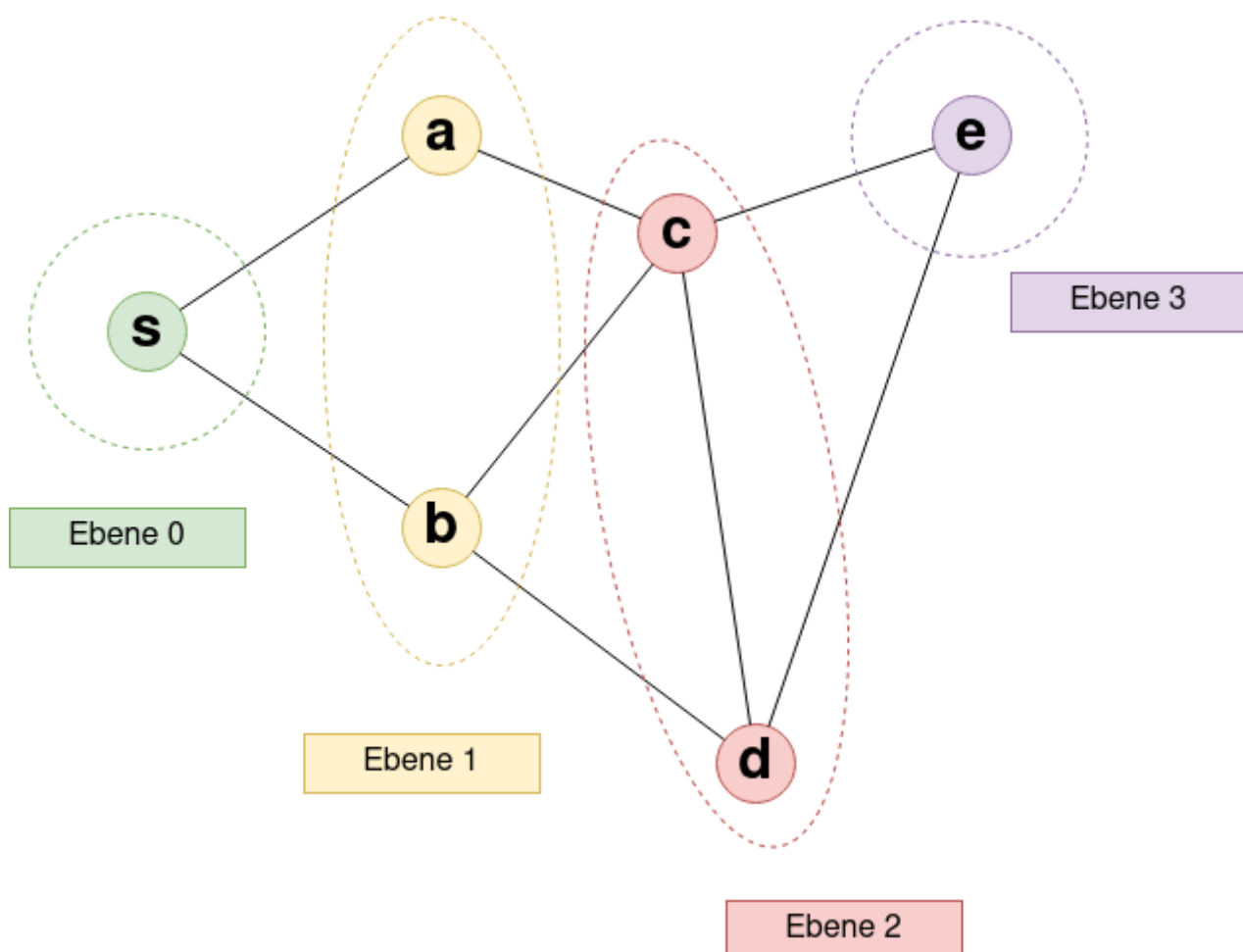
Eine der wichtigsten Operationen auf einem Graphen ist die **Traversierung**. der Begriff **Traversierung** bedeutet dabei, dass der Graph beginnend von einem Startknoten entlang der vorhandenen Kanten durchlaufen wird. Bei gerichteten Graphen wird dabei auch die Kantenrichtung berücksichtigt.

Die beiden grundlegenden Traversierungsarten sind die **Tiefensuche** und die **Breitensuche**.¹⁾

Breitensuche

Bei der Breitensuche werden die Knoten eines Graphen "in Ebenen" untersucht, und zwar in der Reihenfolge der zunehmenden Entfernung vom Startknoten.

Ebene 0 enthält den Startknoten *s*, sonst nichts. Ebene 1 ist die Menge der Knoten, die eine Kante von *s* entfernt sind, also die Nachbarn von *s*. Diese Knoten werden bei der Breitensuche direkt im Anschluss des Startknotens untersucht, die Nachbarknoten dieser Knoten werden zur Untersuchung in Ebene 3 vorgemerkt.



Im Beispielgraph sind *a* und *b* die Nachbarn des Startknotens *s* und bilden somit die Ebene 1.

Allgemein sind die Knoten, die in der Ebene i verarbeitet werden, diejenigen die zu einem Knoten in Schicht $i-1$ benachbart sind und nicht bereits zu einer der Schichten $0, 1, 2, \dots, i-1$ gehören.

Die Breitensuche bearbeitet alle Knoten der nächsten Ebene i unmittelbar nach Abschluss der Erkundung von Ebene $i-1$

Zulässige Abfolgen für eine Breitensuche im Beispielgraphen wären also z.B. s, a, b, d, c, e oder auch s, b, a, c, d, e .

Implementation

Implementiert wird die Breitensuche mit Hilfe eines [Queues](#) für die unbearbeiteten Knoten. Eine Schlange oder ein Queue ist eine einfach "First-In-First-Out"-Datenstruktur, der Pseudocode für eine Breitensuche sieht folgendermaßen aus:

```
// Breitensuche
markiere den Startknoten s als besucht, alle anderen Knoten als nicht
besucht
Q := Ist eine Schlange, die mit dem Startknoten s initialisiert wird

solange "Q ist nicht leer":
  Entferne den ersten Knoten v der Schlange Q
  für alle Nachbarn w von v:
    wenn w nicht besucht ist:
      markiere w als besucht
      füge w ans Ende der Schlange Q an
    ende_wenn
  ende_für
ende_solange
```

Die folgende Grafik zeigt den Ablauf am Beispielgraphen von oben. Rechts ist jeweils der Zustand des Queues dargestellt.

1)

Vgl. [Traversierungsarten in \(Binär-\)Bäumen](#) - Bäume sind auch nur Graphen.

From:
<https://info-bw.de/> -

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:graphen:zpg:kuerzeste_pfade:traversierungen:start?rev=1669046686

Last update: 21.11.2022 16:04

