Algorithmen & Komplexität

Angelika Steger
Institut für Theoretische Informatik

steger@inf.ethz.ch

Breitensuche, Tiefensuche

Wir besprechen nun zwei grundlegende Verfahren, alle Knoten eines Graphen zu durchlaufen

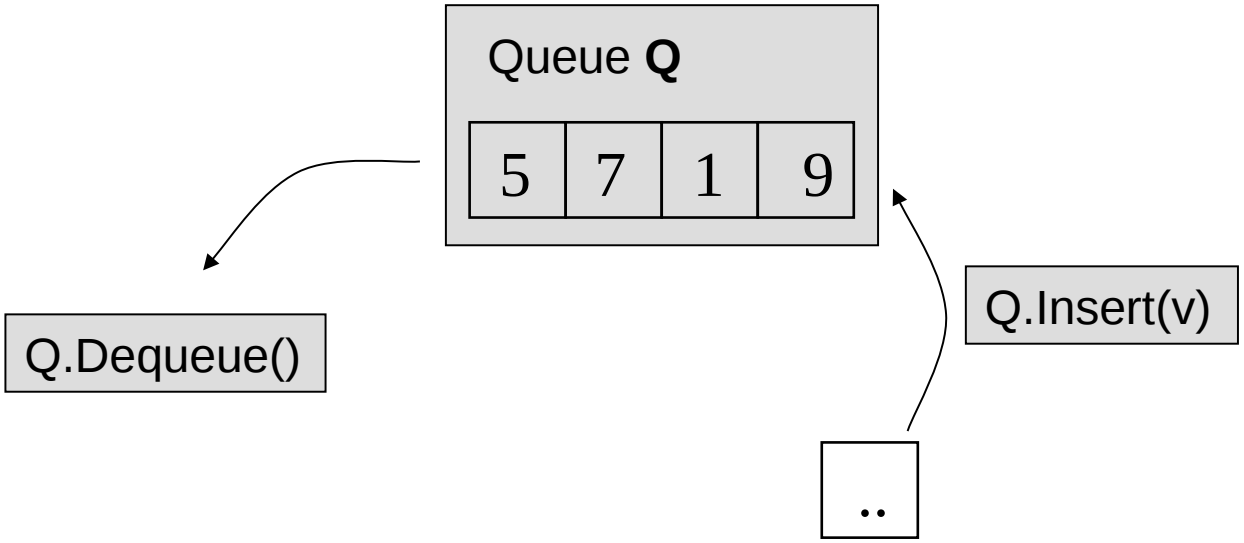
Breitensuche („*breadth first search*“, BFS)

Tiefensuche („*depth first search*“, DFS)

$\Rightarrow$ wichtige Bausteine von „fortgeschrittenen“ Graphenalgorithmen

Datenstruktur „Queue“

- Queue (dt. Warteschlange)
- FIFO („*first in first out*“)



Breitensuche

Input: Graph G=(V,E), Startknoten s∈V

Output: Felder d[v], pred[v] für v∈V

```
Setze d[v] = ∞, pred[v] = nil  ∀v∈V („unbesucht“)
d[s] = 0
Q.Insert(s)
while not Q.IsEmpty()
     v ← Q.Dequeue()
     for all u∈ Γ(v)
        if d[u]=∞ then
           d[u] = d[v]+1
           pred[u] = v
           Q.Insert(u)
         end if
     end for
```

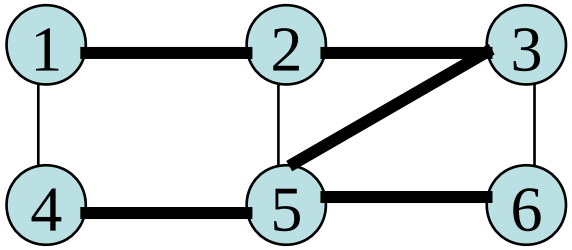
Breitensuche

Satz: $G=(V,E)$, gegeben als Adjazenzlisten, Startknoten s.

Dann gilt:

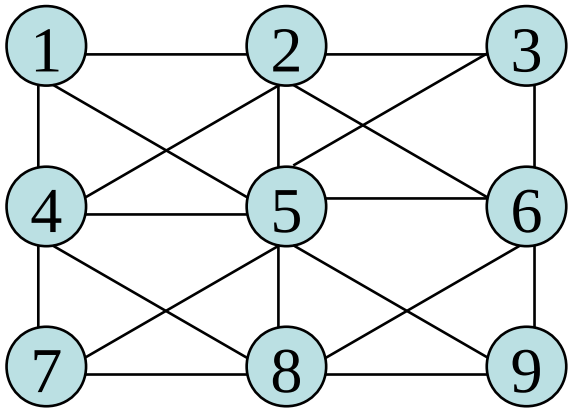
- Die Breitensuche hat eine Laufzeit von $O(|V|+|E|)$.
- $d[v]$ ist die Länge eines kürzesten s-v-Pfades bzw. $d[v]= \infty$, wenn kein solcher existiert.
- Falls G zusammenhängend, bilden die Kanten $\{ \{v,pred[v]\} \mid v \in V \backslash s \}$ einen Spannbaum T von G, mit der Eigenschaft, dass für alle $v \in V$ der eindeutige s-v-Pfad in T der kürzeste s-v-Pfad in G ist.

Tiefensuche - Beispiel

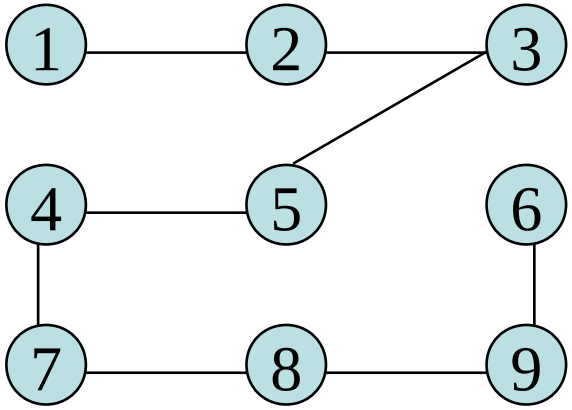


Reihenfolge: 1 2 3 5 4 6

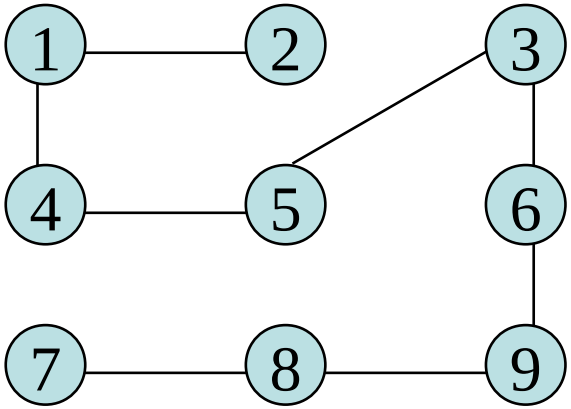
Tiefensuche



DFS ausgehend von **1**



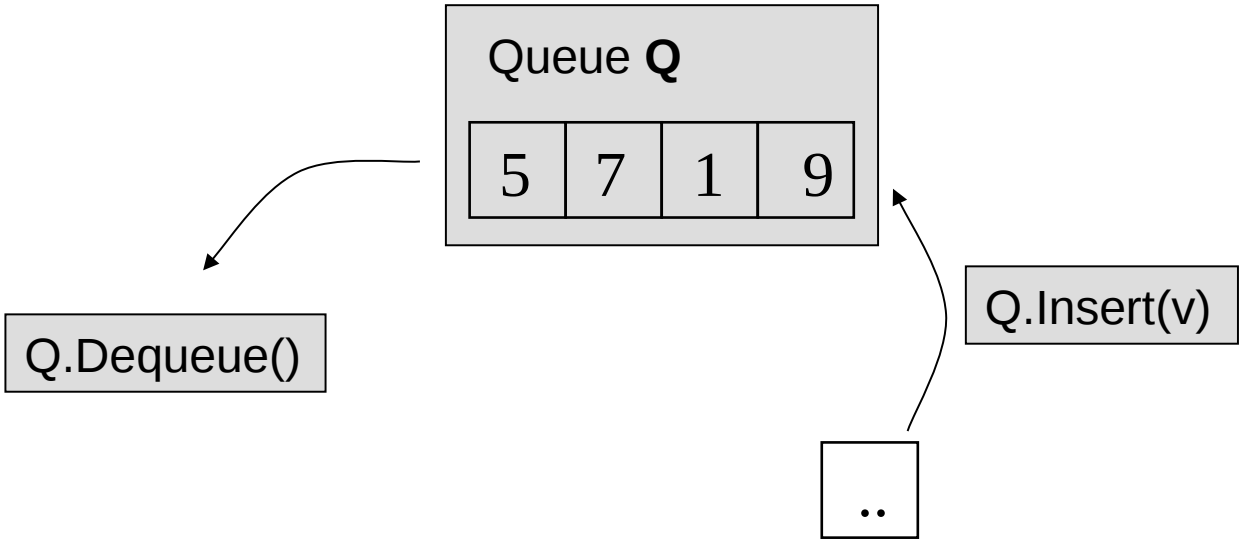
DFS ausgehend von **2**



Datenstruktur „Queue“

für BFS

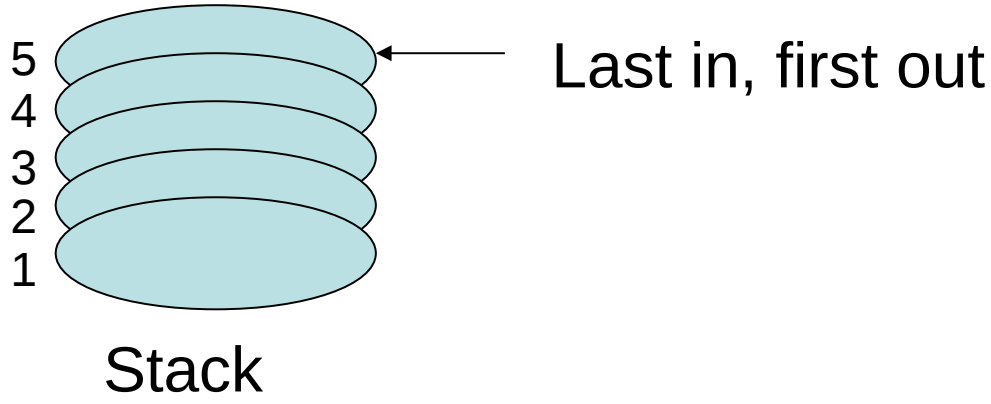
- Queue (dt. Warteschlange)
- FIFO („*first in first out*“)



Datenstruktur „Stack“ (Keller)

für DFS

- LIFO - Queue („*last in first out*“)
- Operationen: Push(v) und Pop()



F.L. Bauer, K. Samelson: *Verfahren zur automatischen Verarbeitung von kodierten Daten und Rechenmaschine zur Ausübung des Verfahrens*, Deutsches Patentamt, Auslegeschrift 1094019, B441221X/42m, 1957.

IEEE Computer Pioneer Award (1988) – „Für die Erfindung des Kellerprinzips“

Tiefensuche

Input: Graph G=(V,E), Startknoten s∈V

Output: Feld pred[v] für v∈V

```
Setze pred[v] = nil für alle v∈V („unbesucht“)
v = s
repeat
   if (∃u ∈ Γ(v) mit pred[u]=nil)  then
      Stack.Push(v)
      pred[u]=v
      v = u
   else if not Stack.IsEmpty()
      v = Stack.Pop()
   else
      v = nil
   end if
until v = nil
```

Tiefensuche

Satz: $G=(V,E)$, gegeben als Adjazenzlisten, Startknoten s.

Dann gilt:

- Die Tiefensuche hat eine Laufzeit von $O(|V|+|E|)$.
- Falls G zusammenhängend, bilden die Kanten $\{ \{v,pred[v]\} \mid v \in V \backslash s \}$ einen Spannbaum von G.

Bemerkung: Diverse Modifikationen von DFS zur Lösung anderer Graphenprobleme.