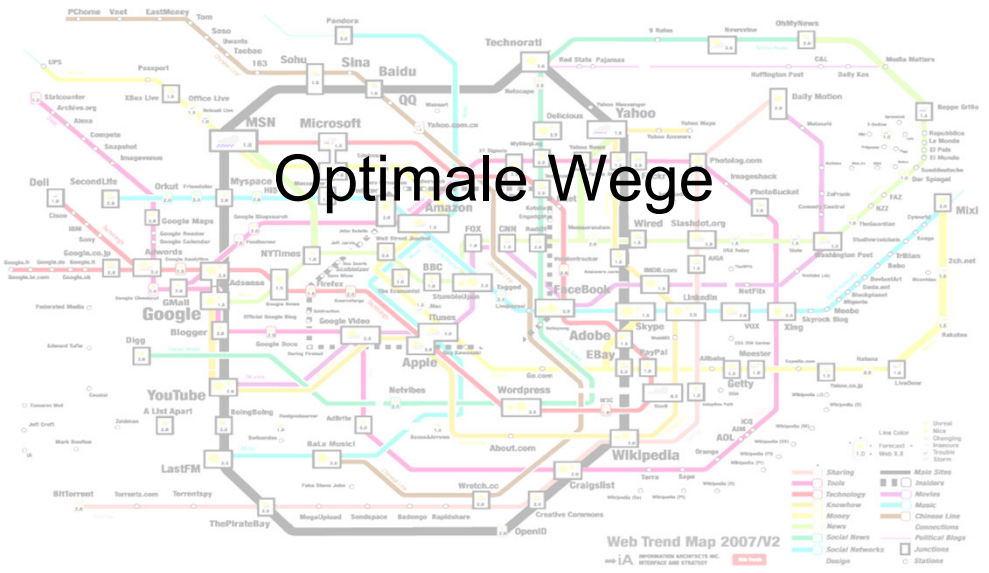
# Optimale Wege






1



## Breitensuche

*Gegeben sei ein Graph G mit zwei ausgezeichneten Ecken* s *und t.*

Schritt 1: *Man kennzeichne die Ecke s mit 0 und setze i = 0.*

Schritt 2: *Man ermittle alle nichtgekennzeichneten Ecken in G, die zu den mit i gekennzeichneten Ecken benachbart sind:*

*Falls es derartige Ecken nicht gibt, ist t nicht mit s über einen Weg verbunden.*

*Falls es derartige Ecken gibt, sind sie mit i+1 zu kennzeichnen.*

Schritt 3: *Wenn t gekennzeichnet wurde, folgt Schritt 4, wenn nicht, erhöhe man i um eins und gehe zu Schritt 2.*

Schritt 4: *Die Länge des kürzesten Weges von s nach t ist i + 1. Der Algorithmus wird beendet.*



2

# Tiefensuche

*Gegeben sei ein Graph G mit zwei ausgezeichneten Ecken* s *und t.*

Schritt 1: *Man kennzeichne die Ecke s mit 0 und setze i = 0.*

Schritt 2: *Man ermittle alle nichtgekennzeichneten Ecken in G, die zu der mit i gekennzeichneten Ecke benachbart sind:*

*Falls es derartige Ecken nicht gibt, wende das Backtracking an.*

*Falls es derartige Ecken gibt, kennzeichne eine (beliebige) mit i+1.*

Schritt 3: *Wenn t gekennzeichnet wurde, folgt Schritt 4, wenn nicht, erhöhe man i um eins und gehe zu Schritt 2.*

Schritt 4: *Die Länge eines Weges von s nach t ist i + 1. Der Algorithmus wird beendet.*

Backtracking*: ermittle eine Ecke, die Vorgänger von der aktuellen Ecke ist (also mit einem kleineren i gekennzeichnet wurde) und die noch nicht markierte adjazente Ecken besitzt.*

*Falls es keine solche Vorgänger Ecke gibt, ist t von s aus nicht erreichbar*

*Falls es eine derartige Ecke gibt, kennzeichne eine (beliebige) zu ihr adjazente (unmarkierte) Ecke mit i+1 und weiter mit Schritt 3.*





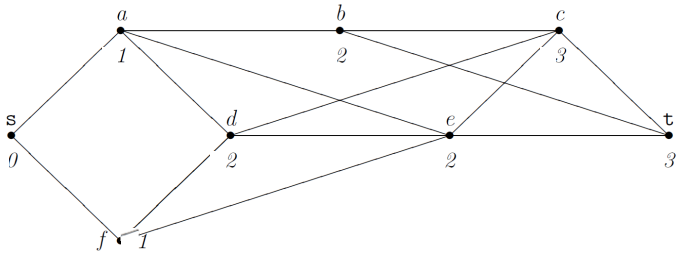
# Verwendung der Kennzeichnung

*Gegeben sei ein Graph G mit zwei ausgezeichneten Ecken* s *und t. Der Graph sei vom BFS gekennzeichnet worden.*

Schritt 1: *Man setze i = λ(t) und ordne $v_i$ = t zu.*

Schritt 2: *Man ermittle eine Ecke u, der zu $v_i$ benachbart ist und mit λ(u)=i−1 gekennzeichnet ist. Man ordne $v_{i-1}$ = u zu.*

Schritt 3: *Wenn i = 1 ist, ist der Algorithmus beendet. Wenn nicht, erniedrige man i um eins und gehe zu Schritt 2.*



# Algorithmus von Dijkstra

**Vorbereitung** *$l_{ij}$: Länge der Kante $v_iv_j$. $l_{ij}$:=∞, falls es eine solche Kante nicht gibt. Für jede Ecke $v_i \in V$ werden drei Variable angelegt:*

1. **$Entf_i$**: *die bisher kürzeste Entfernung von $v_1$ nach $v_i$ an. Startwert 0 für i=1 und ∞ sonst.*
2. **$Vorg_i$** *Vorgänger von $v_i$ auf dem bisher kürzesten Weg von $v_1$ nach $v_i$ an. Startwert $v_1$ für i=1 und undefiniert sonst.*
3. **$OK_i$** *= true, falls die kürzeste Entfernung von $v_1$ nach $v_i$ bekannt ist. Startwert false.*

**Iteration** *Wiederhole (i,j seien dabei die Laufvariablen, h ein fester Wert)*

*Suche unter den Ecken $v_i$ mit $OK_i$ = false eine Ecke $v_h$ mit dem kleinsten Wert von $Entf_i$.*

*Setze $OK_h$ := true.*

*Für alle Ecken $v_j$ mit $OK_j$=false, für die die Kante $v_hv_j$ existiert:*

*Falls gilt $Entf_j > Entf_h + l_{hj}$ dann*

*Setze $Entf_j := Entf_h + l_{hj}$*

*Setze $Vorg_j := h$*

*solange es noch Ecken $v_i$ mit $OK_i$=false gibt.*

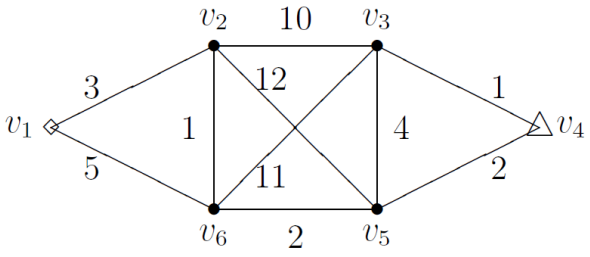






# Aufgaben

1. Führen Sie den Algorithmus von Dijkstra mit nachfolgendem Graphen durch (bis zum Abbruch!). Wählen Sie als Ausgangspunkt die Ecke $v_1$. Fertigen Sie eine Dokumentation an, aus der der Ablauf deutlich wird.



# Aufgaben

2. Führen Sie den Algorithmus von Dijkstra mit nachfolgendem Graphen bis zum Abbruch durch!. Wählen Sie als Ausgangspunkt die Ecke $v_1$. Fertigen Sie eine Dokumentation an, aus der der Ablauf deutlich wird. Zeigen Sie, dass der Algorithmus nicht den kürzesten Weg von $v_1$ nach $v_9$ findet und begründen Sie in allgemeiner Form, woran dies liegt!