

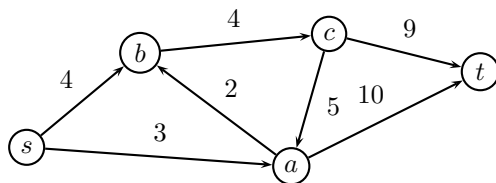
## 1 Kürzeste Wege

### 1.1 Das Single-Source-Shortest-Path Problem (SSSP)

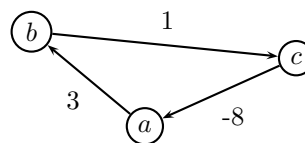
Bei dem SSSP-Problem sind gegeben

- ein gerichteter Graph  $G = (V, E)$ ,
- ein Knoten  $s \in V$ , der Ursprung der kürzesten Wege, die berechnet werden sollen,
- eine Kostenfunktion  $c : E \rightarrow \mathbb{R}$ , die so gewählt ist, dass der Graph keine negativen Kreise enthält. Ein negativer Kreis ist ein Kreis, dessen Durchlauf negative Kosten verursacht (vgl. Abbildung 1.2).

Gesucht sind kürzeste Wege vom Knoten  $s$  zu *jedem* Knoten  $t \in V$ .



1.1: Ein erlaubter Graph



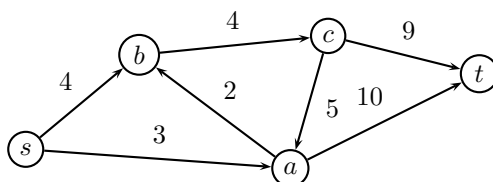
1.2: Ein Graph mit negativem Kreis

Abbildung 1: Zwei gewichtete, gerichtete Graphen

### 1.2 Der Algorithmus von Dijkstra

Der Algorithmus von Dijkstra löst das SSSP mit der zusätzlichen Einschränkung, dass keine negativen Kosten erlaubt sind. Am Beispiel des Graphen in Abbildung 1.1 geht er wie folgt vor:

1. Extrahiere den Knoten aus  $Q$ , der momentan den niedrigsten Distanzwert hat, also den Knoten  $s$ :



$dist$	$s$	$a$	$b$	$c$	$t$
	0	$M$	$M$	$M$	$M$

$$S = \{ \}$$

$$Q = \{s, a, b, c, t\}$$

Abbildung 2: Nach der Initialisierung von  $dist$ ,  $S$  und  $Q$

2. Extrahiere den Knoten aus  $Q$ , der momentan den niedrigsten Distanzwert hat, also den Knoten  $s$ . Anschließend wird *edge relaxation* durchgeführt: Für alle von  $s$  aus erreichbaren Knoten (in diesem Fall also  $a$  und  $b$ ) wird der Distanzwert aktualisiert, wenn man den Knoten über  $s$  schneller erreichen kann. Das ist für  $a$  und  $b$  der Fall, da man sie vorher gar nicht erreichen konnte. Im Folgenden **blau** sind Kanten, die bei der *edge relaxation* betrachtet wurden, **rot** sind Knoten, zu den der kürzeste Weg berechnet wurde bzw. Kanten, die zu einem kürzesten Weg gehören.

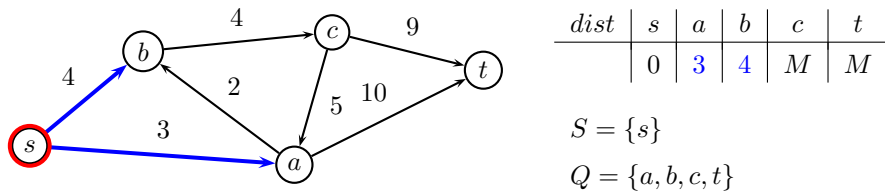


Abbildung 3: Nach der Extraktion von  $s$  und *edge relaxation*

3. Extrahiere den Knoten  $a$  aus  $Q$  und führe *edge relaxation* durch:

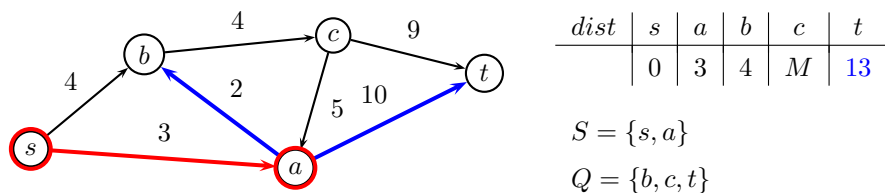


Abbildung 4: Nach der Extraktion von  $a$  und *edge relaxation*

4. Extrahiere den Knoten  $b$  aus  $Q$  und führe *edge relaxation* durch:

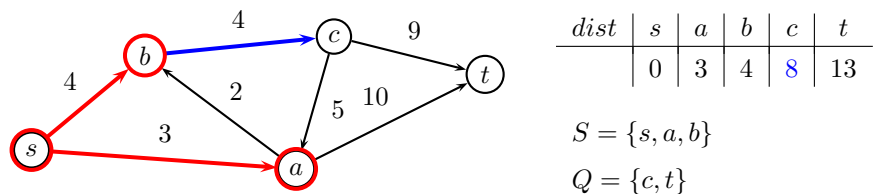


Abbildung 5: Nach der Extraktion von  $b$  und *edge relaxation*

5. Extrahiere den Knoten  $c$  aus  $Q$  und führe *edge relaxation* durch:

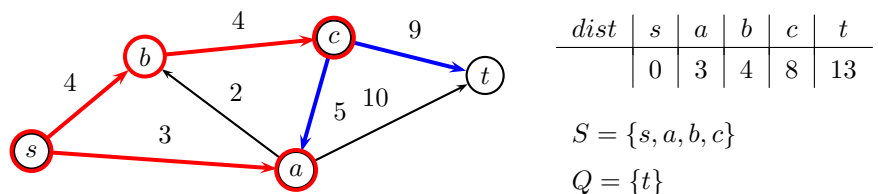


Abbildung 6: Nach der Extraktion von  $c$  und *edge relaxation*

6. Extrahiere den Knoten  $t$  aus  $Q$  und führe edge relaxation durch:

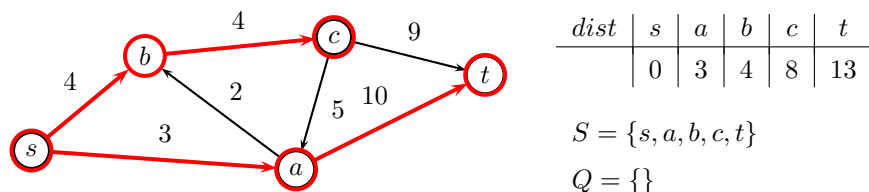


Abbildung 7: Nach der Extraktion von  $t$  und edge relaxation

7.  $Q$  ist nun leer, der Algorithmus hat alle kürzesten Wege von  $s$  aus berechnet.

Die folgende Tabelle zeigt die Analyse des Dijkstra-Algorithmus in Abhängigkeit von der für die Prioritätswarteschlange verwendeten Datenstruktur:

Operationen	Binärer Heap	Fibonacci-Heap
$n$ InsertPrioQ	$n \log n$	$n$
$n$ ExtractMinQ	$n \log n$	$n \log n$
$e$ DecreasePrioQ	$e \log n$	$e$
Gesamt	$(n + e) \log n$	$e + n \log n$

Tabelle 1: Analyse des Algorithmus von Dijkstra

### 1.3 Das All-Pairs-Shortest-Path Problem (APSP)

Bei dem APSP-Problem sind gegeben

- ein gerichteter Graph  $G = (V, E)$ ,
- eine Kostenfunktion  $c : E \rightarrow \mathbb{R}$ , die so gewählt ist, dass der Graph keine negativen Kreise enthält. Ein negativer Kreis ist ein Kreis, dessen Durchlauf negative Kosten verursacht (vgl. Abbildung 1.2).

Gesucht sind die kürzeste Wege von *jedem* Knoten  $s \in V$  zu *jedem* Knoten  $t \in V$ .

### 1.4 Der Algorithmus von Dijkstra für APSP

Der Algorithmus von Dijkstra kann APSP lösen, indem man ihn für jeden Knoten  $s \in V$  ausführt. Es ergibt sich somit folgende Laufzeit:

Operationen	Binärer Heap	Fibonacci-Heap
$n$ Dijkstra-SSSP	$n(n + e) \log n$	$n(e + n \log n)$
Gesamt	$n^2 + ne \log n$	$ne + n^2 \log n$

Tabelle 2: Analyse des Algorithmus von Dijkstra für APSP

## 1.5 Interleaved Dijkstra für APSP

Die folgende Tabelle zeigt die Analyse des Interleaved-Dijkstra-Algorithmus in Abhängigkeit von der für die Prioritätswarteschlange verwendeten Datenstruktur:

Operationen	Binärer Heap	Fibonacci-Heap
$n^2$ InsertPrioQ	$n^2 \log n$	$n^2$
$n^2$ ExtractMinQ	$n^2 \log n$	$n^2 \log n$
$e$ DecreasePrioQ	$ne \log n$	$ne$
Gesamt	$(n^2 + ne) \log n$	$ne + n^2 \log n$

Tabelle 3: Analyse des Interleaved-Dijkstra-Algorithmus