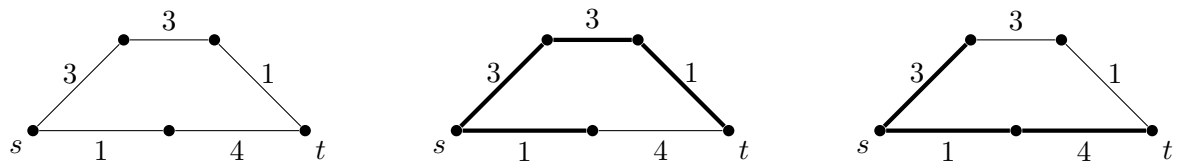
7.3 Der Algorithmus von Dijkstra

Wir werden jetzt einen eng mit dem Algorithmus von Prim verwandten Algorithmus betrachten: den Algorithmus von Dijkstra. Er löst das Problem der Bestimmung des kürzesten Pfades, vgl. Abschnitt 2.2. Dieses Problem hat viele Anwendungen, z.B. die offensichtliche in Navigationssystemen. Wie beim Algorithmus von Prim werden wir sukzessive einen Teilgraphen B von G aufbauen der aus den minimalen Pfaden von s zu den Knoten von B besteht. Zur Verwaltung der Knoten verwenden wir eine Minimum-Prioritätswarteschlange. Der wesentliche Unterschied besteht darin, dass die Priorität $v.x$ eines Knoten v nicht mehr seine Anschlusskosten sind sondern seine B-Distanz von s, d.h. die Länge des kürzesten Pfades von s nach v der bis auf die letzte Kante nur aus Knoten in B besteht. Ebenso wie beim Algorithmus von Prim repräsentieren wir durch $v.\mathit{Vorgänger}$ implizit alle konstruierten Pfade. Der Algorithmus von Dijkstra berechnet die kürzesten Pfade von s zu jedem Knoten $v \in V$. Die Antwort für ein bestimmtes $t \in V$ kann daraus leicht in Zeit $O(|V|)$ bestimmt werden.

Algorithmus 29 Algorithmus von Dijkstra

Prozedur DIJKSTRA(V, E, c, s, t)
  Sei B ein neues Datenfeld der Länge $|V|$, überall mit **falsch** initialisiert
  Für alle $v \in V$
    $v.x := \infty$
    $v.\mathit{Vorgänger} :=$ NIL
  Ende Für
  $s.x := 0$
  $Q :=$ MIN-PRIORITÄTSWARTESCHLANGE(V)
  Solange Q nicht leer
    $v :=$ EXTRAHIEREMINIMUM(Q)
    $B[v] :=$ **wahr**
    Für alle $(v, w) \in E$
      Falls $v.x + c(v, w) < w.x$ **und** $B[w] =$ **falsch dann**
        $w.\mathit{Vorgänger} := v$
        REDUZIEREPRIORITÄT$(Q, w, v.x + c(v, w))$
      Ende Falls
    Ende Für
  Ende Solange
  Antworte $(t, t.\mathit{Vorgänger}, t.\mathit{Vorgänger}.\mathit{Vorgänger}, \ldots, s)^{-1}$
Ende Prozedur

Beispiel 7.3. Ein Graph G mit Kantenkosten (links), der vom Algorithmus von Prim bei Eingabe G erzeugte Spannbaum (Mitte) und die vom Algorithmus von Dijkstra erzeugten Pfade bei Besuch von t (rechts).



Satz 7.4. *Der Algorithmus von Dijkstra ist korrekt.*

Beweis. Sei $G = (V, E)$ sowie $c : E \to \mathbb{R}_{\geq 0}$ die Eingabe, sei $|V| = n$. Für $i = 1, \ldots, n$ sei S_i die Menge der nach dem i-ten Schleifendurchlauf abgearbeiteten, d.h. aus Q extrahierten, Knoten

und für $v \in S_i$ sei p_v der durch die Vorgänger-Felder codierte Pfad von s nach v. Wir behaupten dass für alle $i = 1, \ldots, n$ und alle $v \in S_i$ gilt dass p_v ein kürzester Pfad von s nach v ist. Wir gehen mit Induktion nach i vor. Für $i = 1$ ist das trivial. Für den Induktionsschritt sei v der Knoten in $S_{i+1} \setminus S_i$ und sei u der Vorgänger von v. Dann ist nach Induktionshypothese p_u ein kürzester Pfad von s nach u und $p_v = p_u, v$. Sei p ein anderer Pfad von s nach v, dann hat p die Form p_1, x, y, p_2 wobei p_1, x in S_i ist und y außerhalb von S_i. Da aber nach Definition des Algorithmus v unter allen mit einer Kante aus S_i erreichbaren Knoten in $V \setminus S_i$ minimale Distanz zu s hat, ist $c(p_1, x, y) \geq c(p_v)$ und damit $c(p) \geq c(p_v)$. □

Die Laufzeit vom Algorithmus von Dijkstra ist, wie jene vom Algorithmus von Prim, $O(|E| \log |V|)$.

7.4 Matroide

Definition 7.2. Sei E eine endliche Menge und $\mathcal{U} \subseteq \mathfrak{P}(E)$. $M = (E, \mathcal{U})$ heißt *Matroid* falls:

1. $\emptyset \in \mathcal{U}$,
2. $A \subseteq B \in \mathcal{U}$ impliziert $A \in \mathcal{U}$ und
3. es gilt die folgende *Austauscheigenschaft*:
   Ist $A, B \in \mathcal{U}$ mit $|A| > |B|$, dann gibt es $x \in A \setminus B$ so dass $B \cup \{x\} \in \mathcal{U}$.

Die Bezeichnung Austauscheigenschaft rührt daher, dass in der Menge $A = \{x\} \cup A'$ die Teilmenge A' durch B ausgetauscht werden kann ohne die Eigenschaft der Unabhängigkeit zu verlieren. Wegen Bedingung 2. ist Bedingung 1. erfüllt genau dann wenn die Bedingung 1'. $\mathcal{U} \neq \emptyset$ erfüllt ist, was zu einer alternativen Definition führt.

Beispiel 7.4. Sei M eine Matrix über einem Körper, E die Menge der Spaltenvektoren von M und $\mathcal{U}$ die Menge der linear unabhängigen Mengen von Spaltenvektoren von M. Dann bildet $(E, \mathcal{U})$ ein Matroid:

Klar ist dass $\emptyset$ linear unabhängig ist und dass jede Teilmenge einer linear unabhängigen Menge auch linear unabhängig ist, $(E, \mathcal{U})$ erfüllt also 1. und 2. Für die Austauscheigenschaft seien A und B linear unabhängig und $|A| > |B|$, dann ist $\dim\langle A\rangle = |A| > |B| = \dim\langle B\rangle$. Angenommen für alle $x \in A$ wäre $B \cup \{x\}$ linear abhängig, dann wäre ja $\langle A \cup B\rangle = \langle B\rangle$ und damit $\dim\langle A \cup B\rangle = \dim\langle B\rangle < \dim\langle A\rangle$. Das widerspricht aber $\dim\langle A \cup B\rangle \geq \dim\langle A\rangle$.

Dieser Spezialfall aus der linearen Algebra hat auch historisch den Begriff des Matroids motiviert. Die Austauscheigenschaft ist eine Verallgemeinerung des Austauschsatzes von Steinitz. Der Terminologie der linearen Algebra folgend können wir nun definieren und beweisen:

Definition 7.3. Sei $(E, \mathcal{U})$ ein Matroid. Eine Menge $A \in \mathcal{U}$ heißt *Basis* falls kein A' existiert mit $A \subset A' \in \mathcal{U}$.

Da E endlich und $\mathcal{U}$ nicht leer ist hat jedes Matroid mindestens eine Basis.

Satz 7.5. *Sei $M = (E, \mathcal{U})$ ein Matroid, seien B_1, B_2 Basen von M, dann ist $|B_1| = |B_2|$.*

Beweis. Angenommen B_1 und B_2 wären Basen von M mit $|B_1| > |B_2|$, dann gäbe es nach der Austauscheigenschaft ein $x \in B_1 \setminus B_2$ so dass $B_2 \cup \{x\} \in \mathcal{U}$, also wäre B_2 nicht maximal unabhängig. □

Definition 7.4. Der *Rang* eines Matroids M ist die Kardinalität seiner Basen.

Beispiel 7.5. Sei $G = (V, E)$ ein zusammenhängender Graph. Wir bezeichnen $A \subseteq E$ als unabhängig falls (V, A) zyklenfrei, d.h. ein Wald, ist. Sei $\mathcal{U}$ die Menge aller unabhängigen Kantenmengen, dann ist $M_G = (E, \mathcal{U})$ ein Matroid:

Es ist nämlich $(V, \emptyset)$ zyklenfrei und falls (V, A) zyklenfrei ist und $B \subseteq A$, dann ist auch (V, B) zyklenfrei. Für die Austauscheigenschaft seien $A, B \in \mathcal{U}$ mit $|A| > |B|$. Nach Korollar 2.1 besteht der Wald (V, A) aus $|V| - |A|$ Bäumen und der Wald (V, B) aus $|V| - |B|$ Bäumen. Nach dem Schubfachprinzip gibt es also in (V, A) einen Baum T dessen Knoten zu mindestens zwei Bäumen in (V, B) gehören. Da T zusammenhängend ist, gibt es eine Kante $\{v, w\}$ in T so dass v und w zu verschiedenen Bäumen in (V, B) gehören. Damit ist $B \cup \{\{v, w\}\} \in \mathcal{U}$.

Die Basen von M_G sind dann die maximalen unabhängigen Mengen, d.h. die maximalen zyklenfreien Teilgraphen, d.h. nach Satz 2.3 die Spannbäume von V.

Definition 7.5. Sei $(E, \mathcal{M})$ ein Matroid und $g : E \to \mathbb{R}_{\geq 0}$, dann heißt $(E, \mathcal{M}, g)$ *gewichtetes Matroid*.

Wir können also das Problem der Bestimmung eines minimalen Spannbaums abstrahieren zur Bestimmung einer Basis minimalen Gewichts in einem gewichteten Matroid:

Basis minimalen Gewichts
Eingabe: gewichtetes Matroid $(E, \mathcal{M}, g)$
Ausgabe: Basis $B \in \mathcal{M}$ so dass $\sum_{b \in B} g(b)$ minimal

Dann besteht die korrespondiere Abstraktion des Algorithmus von Kruskal in dem generischen gierigen Algorithmus, der in Algorithmus 30 angegeben ist.

Algorithmus 30 Generischer gieriger Algorithmus

```
Prozedur GIERIG(E, U, g)
    Sortiere E[1, ..., n] so dass g(E[1]) ≤ ··· ≤ g(E[n])
    A := ∅
    Für i := 1, ..., n
        Falls A ∪ {E[i]} ∈ U dann
            A := A ∪ {E[i]}
        Ende Falls
    Ende Für
    Antworte A
Ende Prozedur
```

Beispiel 7.6. Der generische gierige Algorithmus angewandt auf das in Beispiel 7.5 diskutierte Matroid M_G entspricht genau dem Algorithmus von Kruskal.

Satz 7.6. *Sei $M = (E, \mathcal{U}, g)$ ein gewichtetes Matroid. Dann berechnet* GIERIG$(E, \mathcal{U}, g)$ *eine Basis minimalen Gewichts von M.*

Beweis. Sei $(E, \mathcal{U}, g)$ ein gewichtetes Matroid mit Rang r und seien $a_1, \ldots, a_r \in E$ die vom generischen gierigen Algorithmus ausgewählten Elemente in dieser Reihenfolge. Dann ist $g(a_1) \leq \cdots \leq g(a_r)$. Sei $\{b_1, \ldots, b_r\}$ eine Basis mit $g(b_1) \leq \cdots \leq g(b_r)$. Es reicht zu zeigen dass $g(a_i) \leq g(b_i)$ für alle $i \in \{1, \ldots, r\}$. Angenommen es gäbe ein $k \in \{1, \ldots, r\}$ so dass $g(a_k) > g(b_k)$, dann ist $k \geq 2$ da $a_1 \in E$ minimales Gewicht hat. Sei $A = \{a_1, \ldots, a_{k-1}\}$ und $B = \{b_1, \ldots, b_k\}$.

Dann ist $|B| > |A|$ und damit gibt es nach der Austauscheigenschaft ein $b_j \in B \setminus A$ so dass $A \cup \{b_j\} \in \mathcal{U}$. Dann ist aber $g(b_j) \leq g(b_k) < g(a_k)$ und damit steht b_j strikt vor a_k in der Sortierung $e_1, \ldots, e_n$ von E. Der gierige Algorithmus hätte also b_j zu A hinzugefügt bevor er a_k betrachtet hätte, Widerspruch. □

Beispiel 7.7. Sei $G = (V, E)$ ein zusammenhängender Graph, $g : E \to \mathbb{R}_{\geq 0}$ und $s \in V$. Wir definieren dann P als die Menge aller bei s beginnenden zyklenfreien Pfade und erweitern g auf P durch $g(p) = \sum_{e \text{ auf } p} g(e)$. Eine Menge $A \subseteq P$ heißt unabhängig wenn die Pfade in A paarweise verschiedene Endknoten haben. Sei $\mathcal{U}$ die Menge der unabhängigen Pfadmengen, dann ist $M = (P, \mathcal{U}, g)$ ein gewichtetes Matroid.

Klar ist $\emptyset \in \mathcal{U}$ und $A \subseteq B \in \mathcal{U}$ impliziert $A \in \mathcal{U}$. Die Austauscheigenschaft gilt, da jede unabhängige Menge von $k + 1$ Pfaden einen Endknoten enthält der nicht Endknoten in einer unabhängigen Menge von k Pfaden ist.

Eine Basis von M ist eine Menge von Pfaden die alle Knoten erreichen, eine Basis minimalen Gewichts enthält für jeden Knoten v nur einen kürzesten Pfade von s nach v. Der Rang von M ist $|V|$. Der generische gierige Algorithmus entspricht dann in Ablauf und Ergebnis dem Algorithmus von Dijkstra.