1 Graphentheorie

Graphen sind diskrete mathematische Objekte, mit denen Netzwerke verschiedenster Art modelliert werden können. Es existieren viele praktische Anwendungen und entsprechende Algorithmen und Datenstrukturen dazu.

1.1 Definitionen und Begriffe

Knotenmenge V (Vertexset). Oft $\{0, 1, 2, \ldots, n-1\}$. Anzahl Knoten $n = |V|$.

Kantenmenge E (Edgeset). Menge aus *Paaren* (gerichtet, Pfeile, Einbahnstrassen) oder **Mengen** der Grösse 2 (ungerichtet, Linien) von Knoten. Anzahl Kanten $m = |E|$.

Graph $G = (V, E)$. Ein Paar von Knoten- und Kantenmenge.

Beispiel 1 (ungerichtet)
$V = \{0, 1, 2, 3, 4\}$
$E = \big\{\{0,1\}, \{0,2\}, \{0,3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{2,4\}, \{3,4\}\big\}$

Beispiel 2 (gerichtet)
$V = \{0, 1, 2, 3, 4\}$
$E = \big\{(0,1), (0,2), (3,0), (1,2), (3,1), (3,2), (2,4), (4,3)\big\}$

Nachbarschaft eines Knotens:
Gerichtet: $N^+(v) = \{u \in V \mid (v,u) \in E\}$ und $N^-(v) = \{u \in V \mid (u,v) \in E\}$ und $N(v) = N^+(v) \cup N^-(v)$
Ungerichtet $N(v) = \{u \in V \mid \{u,v\} \in V\}$.

Grad eines Knotens $d^+(v) = |N^+(v)|, \quad d^-(v) = |N^-(v)|, \quad d(v) = |N(v)|$

Weg Folge von Knoten $\{v_1, v_2, \ldots, v_k\}$, so dass $(v_i, v_{i+1}) \in E \;\; \forall i = 1, \ldots, k-1$.

Zyklus Weg mit $v_1 = v_k$.

Zusammenhängender Graph Für alle Knotenpaare gibt es einen Weg von einen zum anderen.

Baum Zusammenhängender Graph ohne Zyklus.

Kompletter Graph Alle möglichen Kanten existieren.

Planarer Graph Kann in der Ebene ohne Kantenüberschneidungen gezeichnet werden.

Aufgabe 1 Für die Beispiele 1 und 2,

a) bestimmen Sie für die Knoten 0 und 4 die verschiedenen Nachbarschaften und Grade der Knoten.

b) finden Sie zwei Wege vom Knoten 0 zum Knoten 4.

c) finden Sie einen Zyklus mit 4 Knoten

d) bestimmen Sie eine Untermenge $E' \subset E$, so dass $G' = (V, E')$ ein Baum ist.

e) bestimmen Sie, ob die Graphen zusammenhängend sind.

Aufgabe 2 In Beispiel 2 wurde die Orientierung einer Kante geändert und der Graph ist immer noch zusammenhängend. Welche Kanten kommen dafür in Frage?

Aufgabe 3 Ist Beispiel 1 planar?

Aufgabe 4 Ist a) K_4 und b) K_5, der komplette Graph mit 4, bzw. 5 Knoten planar?

Aufgabe 5 Finden Sie für Beispiel 1 einen Weg, der jede Kante genau einmal benutzt. Gleiche Frage für Beispiel 2.

Aufgabe 6 Wie gross ist die Anzahl Kanten in einem kompletten Graphen mit n Knoten, wenn er a) ungerichtet und b) gerichtet ist?

Aufgabe 7 Wie gross ist die Anzahl Kanten eines Baumes mit n Knoten mindestens und höchstens?

Aufgabe 8 Finden Sie einen Weg, der die Kanten eines Würfels alle genau einmal beschreitet. Wie sieht es mit einem Tetraeder, Oktaeder, Dodekaeder und Ikosaeder aus?

Aufgabe 9 Die Koordinaten der Eckpunkte des Einheitsquadrates sind $(0,0)$, $(0,1)$, $(1,1)$ und $(1,0)$. Überlegen Sie sich, wie die Koordinaten des Einheitswürfels sind und welche Eckpunkte miteinander verbunden werden. Verallgemeinern Sie auf 4 Dimensionen und schreiben Sie ein Programm, das die Knotenmenge und die Kantenmenge eines 4D-Würfels generiert.
Finden Sie einen Weg, der die Kanten eines 4D-Würfels alle genau einmal beschreitet.

Aufgabe 10 Gegeben ist ein ungerichteter Graph $G = (V, E)$ und ein Zyklus $\{v_1, v_2, \ldots, v_{m+1}\}$ der sämtliche Kanten genau einmal benutzt.
Was lässt sich über $d(v)$ für $v \in V$ aussagen?
Was lässt sich über $d(v)$ aussagen, wenn der Zyklus ein Weg ist (d.h. nicht geschlossen), insbesondere für $d(v_1)$ und $d(v_{m+1})$?

Definition 1 Eulerzyklus/weg

Ein **Eulerzyklus** bzw. -weg ist ein Zyklus bzw. Weg, der jede Kante genau einmal benutzt.

Aufgabe 11 Entwickeln Sie einen Algorithmus, der für einen gegebenen Graphen einen Eulerweg bestimmt.

Aufgabe 12 Schätzen Sie die Anzahl Rechenschritte ihres Algorithmus' zur Bestimmung eines Eulerweges als Funktion des Graphen ab (typischerweise n, evtl. m oder sogar $d(v)$).

Aufgabe 13 Entwickeln Sie einen Algorithmus, der für einen gegebenen Graphen *alle* Eulerwege bestimmen kann. Schätzen Sie ab, wie viele solche Wege es gibt, als Funktion von n, evtl. m und $d(v)$.

Aufgabe 14 Entwickeln Sie einen Algorithmus der überprüft, ob ein Graph zusammenhängend ist oder nicht. Schätzen Sie dessen Komplexität ab.

Definition 2 Hamiltonzyklus

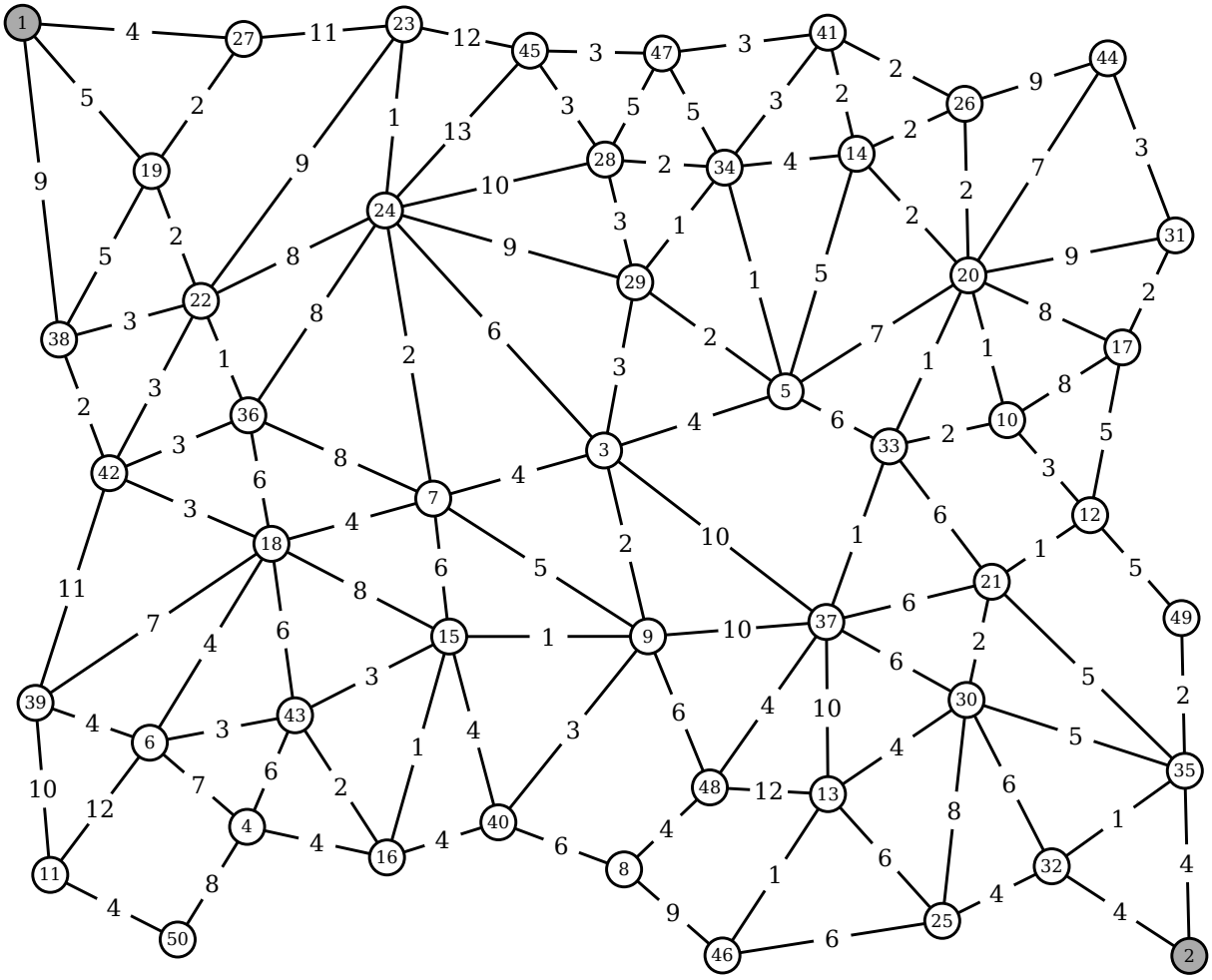
Ein **Hamiltonzyklus** ist Weg, der jeden Knoten genau einmal besucht.

Aufgabe 15 Entwickeln Sie einen Algorithmus, der für einen gegebenen Graphen einen Hamiltonzyklus bestimmt.

Aufgabe 16 Schätzen Sie die Anzahl Rechenschritte ihres Algorithmus' zur Bestimmung eines Hamiltonzyklus als Funktion des Graphen ab (typischerweise n, evtl. m oder sogar $d(v)$).

1.2 Kürzeste Wege

Aufgabe 17 Finden Sie den kürzesten Weg vom Knoten 1 zum Knoten 2. Die Weglänge ist die Summe der Längen der benutzten Kanten. Diese Länge ist auf den Kanten markiert und stimmt nicht mit der geometrischen Länge überein.



Aufgabe 18

a) Wenn es Kanten mit negativer Länge gibt, was ist die Voraussetzung, dass es überhaupt einen kürzesten Weg gibt?

b) Finden Sie ein Anwendungsbeispiel, wo man am längsten Weg interessiert ist? Was sind die Vorausetzungen, dass ein längster Weg existiert?

c) Stellen Sie sich die Knoten als Kugeln vor, und die Kanten als Fäden entsprechender (positiver) Länge. Nehmen wir an, alle Kugeln liegen am Boden. Beschreiben Sie, was passiert, wenn man den Startknoten langsam anhebt (und man vernachlässigt, dass es ein «Gnusch» geben könnte).

d) Formulieren Sie einen Algorithmus für Graphen mit positiver Kantenlänge der von einem Startknoten aus die kürzesten Wege zu allen anderen Knoten bestimmt. Inspirieren Sie sich dabei bei Teilaufgabe c)

1.3 Suchalgorithmen auf Graphen

Merke Dijkstras Algorithmus

Der Algorithmus von Dijkstra findet in einem Graphen mit positiven Kantenlängen den kürzesten Weg von einem Startknoten zu allen anderen Knoten.

Aufgabe 19 Schätzen Sie die Komplexität vom Algorithmus von Dijkstra ab. Siehe Aufgabe 18 d).

Der Algorithmus von Dijkstra kann als Spezialfall einer ganzen Familie von Suchalgorithmen auf Graphen aufgefasst werden:

Merke Suchalgorithmus auf Graphen

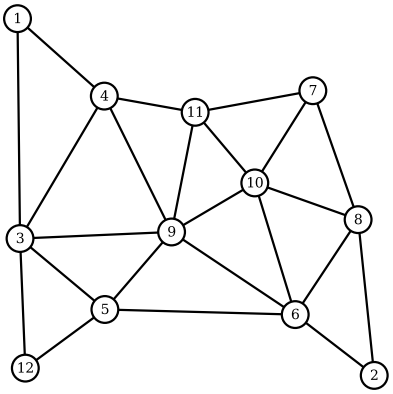
Input: Graph $G = (V, E)$, Startknoten s

1. Initialisierung: Todo-Liste $T = \{s\}$, Markierung $x(v) = 0\ \forall v \in V$, $x(s) = 1$.
2. So lange $T \neq \varnothing$:
   (a) Ordne die Todo-Liste nach einem bestimmten Kriterium
   (b) Sei u der erste Knoten in der Todo-Liste
   (c) «Was sinnvolles mit u tun.»
   (d) u aus T entfernen.
   (e) Für alle $v \in N^+(u)$ mit $x(v) = 0$:
      i. Sei $x(v) = 1$
      ii. Füge v zu T hinzu.

Aufgabe 20 Wie muss die Todo-Liste beim Suchalgorithmus sortiert werden, damit der Algorithmus von Dijkstra herauskommt? Welche zusätzlichen Operationen und Markierungen sind nötig?

Aufgabe 21

Bestimmen Sie die Reihenfolge, in der die Knoten vom Suchalgorithmus besucht werden, wenn beim Knoten 10 gestartet wird. Man nimmt an, dass die unmarkierten Nachbarn in aufsteigender Reihenfolge durchgegangen werden.
a) Die Knoten werden hinten an T angefügt und vorne entnommen.
b) Die Knoten werden vorne an T angefügt und vorne entnommen.
c) Die Todo-Liste wird nach aufsteigender Knotennummer sortiert und der jeweils erste Knoten entnommen.



Aufgabe 22 Wie muss die Todo-Liste gehandhabt werden (ohne Sortieren), damit

a) Die Knoten in der Reihenfolge der Distanz (in Anzahl Kanten gemessen) besucht werden? Diese Art von Suche nennt man «Breadth-First», oder «Breitensuche».

b) Die Knoten «möglichst weit weg» möglichst früh besucht werden? Diese Art von Suche nennt man «Depth-First» oder Tiefensuche.

Aufgabe 23 Schreiben Sie einen Pseudocode, der die Tiefensuche rekursiv und ohne Todo-Liste implementiert.

1.4 Lösungen

Hinweise zu den Symbolen:

⚒ Diese Aufgaben könnten (mit kleinen Anpassungen) an einer Prüfung vorkommen. Für die Prüfungsvorbereitung gilt: "If you want to nail it, you'll need it".

Diese Aufgaben sind wichtig, um das Verständnis des Prüfungsstoffs zu vertiefen. Die Aufgaben sind in der Form aber eher nicht geeignet für eine Prüfung (zu grosser Umfang, nötige «Tricks», zu offene Aufgabenstellung, etc.). **Teile solcher Aufgaben können aber durchaus in einer Prüfung vorkommen!**.

Diese Aufgaben sind dazu da, über den Tellerrand hinaus zu schauen und oder die Theorie in einen grösseren Kontext zu stellen.

⚒ Lösung zu Aufgabe 1 ex-definitionen-anwenden

a) Für die beiden Beispiele sind $N(0) = \{1,2,3\}$ und $N(4) = \{2,3\}$. Für Beispiel 2 ist $N^+(0) = \{1,2\}$, $N^-(0) = \{3\}$ und $N^+(4) = \{3\}$, $N^-(4) = \{4\}$.

b) Beispiel 1: $\{0,3,4\}$ und $\{0,2,4\}$ (andere Wege sind möglich). Beispiel 2: $\{0,1,2,4\}$ und $\{0,2,4\}$ (andere einfache Wege (ohne Mehrfachnennungen von Knoten) sind nicht möglich).

c) Beispiel 1: $\{0,1,2,3,0\}$ und $\{1,2,4,3,1\}$. Beispiel 2: $\{1,2,4,3,1\}$ und $\{0,2,4,3,0\}$.

d) Nur für Beispiel 1: z.B. $\{\{0,1\},\{0,2\},\{2,4\},\{2,3\}\}$

e) Beispiel 1 offensichtlich ja. Beispiel 2: Es gibt z.B. einen Zyklus $\{0,1,2,4,3,0\}$, somit kann entlang dieses Zyklus von jedem zu jedem Knoten gegangen werden.

⚒ Lösung zu Aufgabe 2 ex-kanten-aendern-konnex

Der Graph ist nicht mehr zusammenhängend, wenn es einen Knoten gibt, bei dem alle Kanten in die gleiche Richtung gehen. Das ist bei den Kanten $(1,2)$, $(2,4)$, $(4,3)$, $(3,0)$ der Fall. D.h. diese dürfen auf keinen Fall entfernt werden. Dank dem Zyklus $\{0,1,2,4,5,0\}$ spielt die Orientierung der Kanten $(2,3)$, $(3,1)$ und $(0,2)$ keine Rolle. D.h. diese dürfen umgedreht werden.

Für die letzte Kante $(0,1)$ stellt man fest, dass der Graph noch zusammenhängend ist. Z.B. via den Zyklus $(0,2,3,1,2,4,3,0)$.

⚒ Lösung zu Aufgabe 3 ex-planar-bsp1

Ja. Z.B. mit dem Knoten 1 in der Mitte, Knoten 0,2,3 als Dreieck drum herum, Knoten 4 als Dreieck darüber.

Lösung zu Aufgabe 4 ex-k4-k5-planar

K_4 ist planar (ein Punkt in der Mitte eines Dreiecks). K_5 hingegen ist nicht planar. Der Beweis dafür ist etwas komplizierter und beruht auf der Euler-Identität für Polyeder.

⚒ Lösung zu Aufgabe 5 ex-euler-pfad-beispiel1

Z.B. $\{0,1,2,4,3,2,0,3,1\}$.

Für Beispiel 2 ist das nicht möglich, weil z.B. im Knoten 2 drei Kanten hineinführen aber nur eine hinaus. Die Differenz kann höchstens 1 sein und auch das nur in den beiden Knoten, wo der Weg startet oder endet.

⚒ Lösung zu Aufgabe 6 ex-anzahl-kanten-komplette-graphen

Ungerichtet: $\frac{n(n-1)}{2}$. Jeder der n Knoten ist mit $n-1$ Knoten verbunden. Dabei zählt man jede Kante doppelt.

Gerichtet: $n(n-1)$ (bzw. n^2 wenn man Schlaufen erlaubt (Kanten von eine Knoten zu sich selbst)).

⚒ Lösung zu Aufgabe 7 ex-anzahl-kanten-im-baum

Die Anzahl ist immer $n-1$. Ein möglicher Beweis ist wie folgt:

Behauptung: «Es gibt in jedem Baum mit $n > 1$ einen Knoten mit Grad 1».

Beweis der Behauptung: In einem Baum gibt es keine Zyklen. D.h. wenn man sich durch den Baum bewegt, kommt man früher oder später in eine «Sackgasse» (auch Blatt eines Baumes genannt).

Beweis: Sei G ein Baum und v ein Knoten mit $d(v) = 1$. Dieser Knoten mit der einen Kante wird entfernt. Was übrigbleibt ist wieder ein Baum mit einem Knoten und einer Kante weniger. Diesen Prozess kann man so lange fortsetzen, bis nur noch 1 Knoten und 0 Kanten übrigbleiben. D.h. man insgesamt $n-1$ Kanten entfernt.

⚒ Lösung zu Aufgabe 8 ex-eulerpfad-auf-polyeder
Auf dem Würfel ist so ein Weg nicht möglich, da von jeder Ecke 3 Kanten ausgehen. Für alle auser der ersten und letzten Ecke muss man aber gleich viel mal in einen Knoten gehen wir man herauskommt. D.h. die Grade müssen gerade sein.

Tetraeder und Dodekaeder haben ebenfalls 3 Kanten pro Ecke, das Ikosaeder hat 5. Nur das Oktaeder hat 4 Kanten pro Ecke. Dort existiert so ein Weg.

⚒ Lösung zu Aufgabe 9 ex-hyperwuerfel-eulerpfad
Ein Quadrat, Würfel, 4-Würfel wird durch die Einheitsvektoren aufgespannt. Die Eckpunkte sind alle Kombinationen von 0/1 und der entsprechenden Anzahl Koordinaten. Für 3D wären das $(0,0,0)$, $(0,0,1)$, $(0,1,0)$,…, $(1,1,1)$. Dadurch ergibt sich eine «natürliche» Nummerierung der Knoten 0 bis 7, indem man die Koordinaten als Binärzahl interpretiert.

In 4 Dimensionen entsprechen die Koordinaten allen 4-stelligen Binärzahlen 0b0000 bis 0b1111 (16 Knoten). Zwei Knoten sind genau dann miteinander verbunden, wenn sie sich um genau 1 Bit unterscheiden (was der Bewegung entlang einer Koordinatenrichtung entspricht).

Ein möglicher Weg ist z.B.

0,1,3,7,15,14,12,8,10,11,9,13,15,11,3,2,6,4,12,13,5,7,6,14,10,2,0

⚒ Lösung zu Aufgabe 10 ex-grade-wenn-eulerzyklus
Bei jedem Besuch eines Knoten werden zwei Kanten benutzt (eine ankommend, eine abgehend). Damit sind alle Grade gerade.

Bei einem Weg geht vom ersten Knoten eine Kante mehr aus, beim letzten eine mehr ein. D.h. die beiden Knoten haben einen ungerade Grad. Alle anderen Knoten haben gerade Grade.

⚒ Lösung zu Aufgabe 11 ex-euler-algo
Input: Graph $G = (V, E)$. **Output:** Eulerweg oder Meldung, dass es keinen gibt.

1. Wenn es mehr als zwei Knoten mit ungeradem Grad gibt: **kein Euler-Weg**.
2. Wenn es mehr als zwei Knoten mit $N^+(v) \neq N^-(v)$ gibt: **kein Euler-Weg**.
3. Wenn es einen Knoten mit $|N^+(v) - N^-(v)| > 1$ gibt: **kein Euler-Weg**.
4. Starknoten s wählen:
   - Knoten mit $N^+(s) - N^-(s) = 1$, falls ein solcher existiert.
   - Knoten mit $d(s)$ ungerade, falls ein solcher existiert.
   - Andernfalls ein zufälliger Knoten
5. Resultatweg $p := \{s\}$
6. Unterweg p' suchen ab s: Aktueller Knoten $v := s$, $p' = \{\}$.
7. Den Weg p' unmittelbar hinter s in p einfügen.
8. Wenn alle Kanten makiert, **Resultat p zurückgeben**.
9. Knoten in p mit unbenutzter Kante finden und in s speichern. Gehe zu 6.

Idee: Sucht man zufällig einen Weg, muss dieser wieder zurückkommen, weil sonst überall, wo man hingeht, noch eine unbenutzte Kante weiterführt. Ab Knoten mit noch unbenutzen Kanten können weitere Unterzyklen gefunden un eingefügt werden.

⚒ Lösung zu Aufgabe 12 ex-komplexitaet-euler-algo
Die Abschätzung hängt auch davon ab, wie genau der Graph gespeichert wird.

Beim Durchsuchen der Kanten müssen im schlimmten Fall alle angeschaut werden, was einen Aufwand von $O(m)$ pro Kante bedeutet. Um den nächsten Knoten zu bestimmen, müssen u.U. alle Knoten abgesucht werden, also $O(n)$. D.h. man kommt auf einen Rechenaufwand von $O(m^2)$.

Werden die Kanten aber für jeden Knoten gespeichert und jeweils entfernt, kann beim Wegsuchen die nächste Kante in konstanter Zeit gewählt werden.

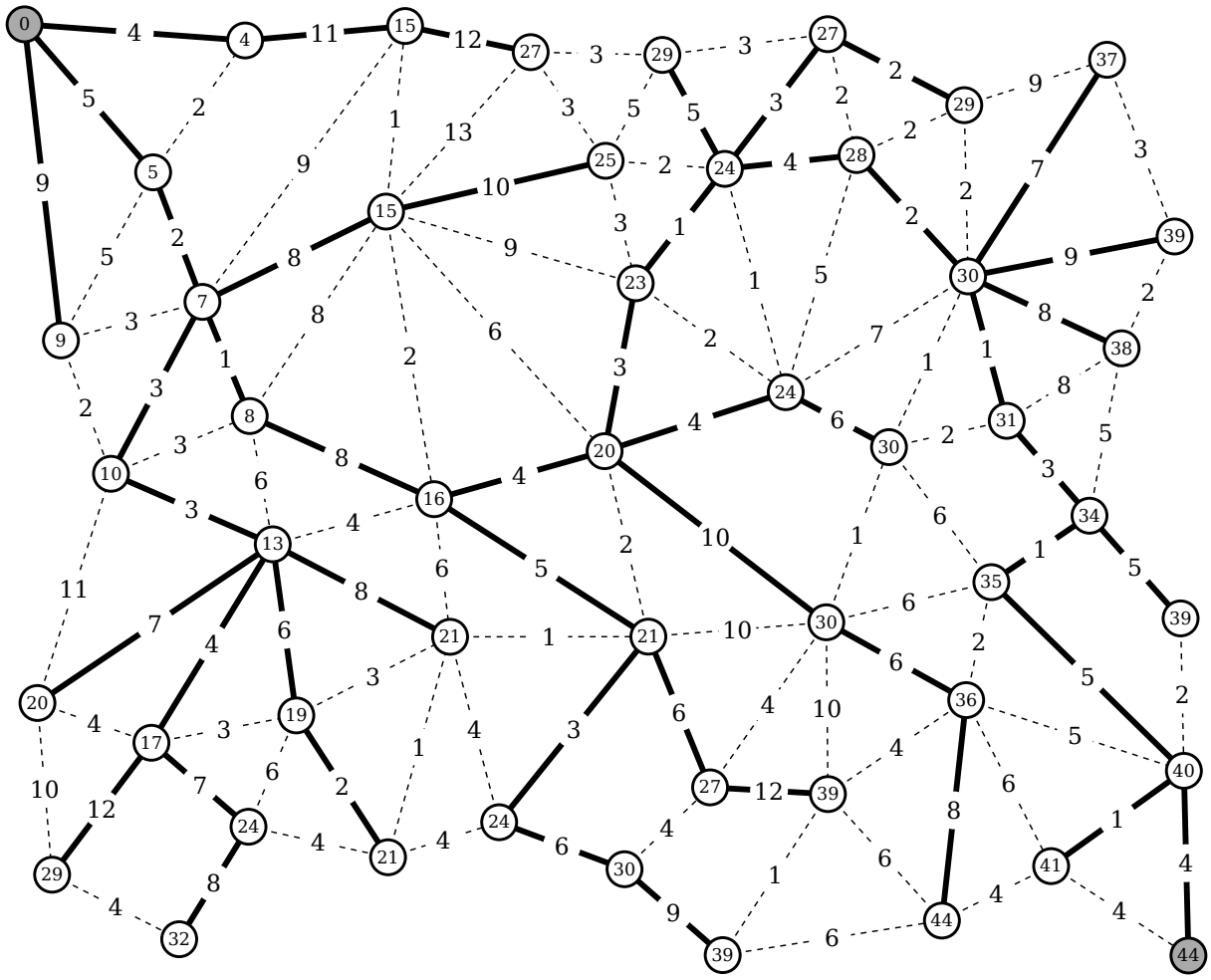
Um den nächsten ungesättigten Knoten im unfertigen Pfad zu finden können diese ebenfalls «geschickt» gespeichert werden, so dass die Knoten in konstanter Zeit gefunden werden können. Es ist möglich, diesen Algorithmus in $O(m)$ zu implementieren.

⚒ Lösung zu Aufgabe 15 ex-hamilton-zyklus
Der folgende Algorithmus ist rekursiv, d.h. definiert eine Funktion, die sich selbst wieder aufruft.
Input: Graph $G = (V, E)$, Partieller (unvollständiger) Weg $P = \{v_1, \ldots, v_k\}$.
Output: Vollständiger Hamiltonzyklus, der den partiellen Weg enthält, oder `nil`, wenn kein solcher mit diesem partiellen Weg existiert.
Funktion `hamilton(`G`,` P`)`

1. Falls $k = n$ (alle Knoten im Weg P)
   (a) Wenn $(v_n, v_1) \in E$, dann v_1 hinten an P anhängen und P zurückgeben.
   (b) Sonst `nil` zurückgeben.
2. Sonst
   (a) Für alle Nachbarn u von v_k, mit $u \notin P$
      - $P' =$`hamilton(`G`,` $P \cup \{u\}$`)`
      - Wenn $P' \neq$`nil`, dann P' zurückgeben.
   (b) `nil` zurückgeben.

⚒ Lösung zu Aufgabe 16 ex-hamilton-komplexitaet
Für jeden Knoten u, der angefügt wird, wird im schlimmsten Fall der Loop $d(u)$ Mal durchlaufen. Wenn $\overline{d}$ der durchschnittliche Grad eines Knotens ist, ist der Aufwand in etwa $O(\overline{d}^n)$. Das ist eine exponentielle Komplexität. D.h. man stösst sehr schnell an eine Grenze, wo das Problem nicht mehr praktisch lösbar ist. Man kann den Algorithmus zwar noch einiges effizienter programmieren, die exponentielle Komplexität bringt man aber nicht weg. Bewiesen ist dies allerdings nicht. Ein Beweis oder Gegegenbeweis ist 1'000'000$ wert, weil damit eines Milleniumprobleme gelöst wäre, nämliche die Frage ob $P = NP$.

⚒ Lösung zu Aufgabe 17 ex-shortest-path
Sämtliche kürzeste Wege vom Knoten 1 aus sind markiert. Die Knoten sind mit der entsprechenden Weglänge markiert.



Lösung zu Aufgabe 18 ex-shortest-path-theorie

a) Es darf keine Zyklen negativer Länge geben. Sonst könnte dieser beliebig oft durchlaufen werden und damit immer kürzere Wege erzeugt werden.

b) Planung von Aufgaben, die von einander abhängen, wobei die Länge einer *gerichteten* Kante (A, B) angibt, wie lange man mit dem Start von B mindestens warten muss, gemessen vom Start von A an.

Fügt man einen fiktiven Startknoten s mit allen Kanten (s, v) der Länge 0 und einen fiktiven Endknoten e mit allen Kanten (v, e) der Länge 0 hinzu, stellt der längste Weg von s nach e die Mindestdauer dar, die es für die Realisierung vom Projekt braucht. Zusätzlich entsprechen die Knoten auf dem längsten Weg den *kritischen* Aufgaben, d.h. eine Verspätung bei diesen Aufgaben führt zu einer Verspätung beim ganzen Projekt.

c) Es wird zunächst diejenige Kugel angehoben, die am nächsten beim Startknoten liegt. Danach diejenige, die am zweitnächsten liegt (entweder direkt durch eine Kante mit dem Startknoten verbunden oder via dem nächsten Knoten). Danach wird die dritt-nächste Kugel angehoben etc.

d) **Input**: Graph $G = (V, E)$, Kantenlängen $d : E \to \mathbb{R}_0^+$. Startknoten s
Output: Kürzeste Entfernung $\ell : V \to \mathbb{R}_0^+$, Vorgängerinformation $p : V \to V$.

1. Intitialisierung: $\ell(v) = \infty \; \forall v \in V$, $\ell(s) = 0$, $p(s) = \varnothing$, $x(v) = 0 \; \forall v \in V$ (alle Knoten als noch auf dem Boden liegend markieren).
2. Sei u der Knoten mit kleinstem $\ell(u)$ und $x(u) = 0$
3. Wenn u nicht existiert, **fertig!**
4. Markiere $x(u) = 1$.
5. Für alle $v \in N^+(u)$ mit $x(v) = 0$
   (a) Wenn $\ell(u) + d((u, v)) < \ell(v)$
      i. Sei $p(u) = v$
      ii. Sei $\ell(u) = \ell(u) + d((u, v))$
6. Gehe zu Punkt 2

⚒ Lösung zu Aufgabe 19 ex-dijkstra-complexity
Initalisierung: $O(n)$

Folgende Punkte werden n mal ausgeführt:
Punkt 2 (nächsten Knoten finden): Naiv $O(n)$ (alle anschauen), mit Heap $O(\log(n))$.
Punkt 5 (Nachbarn von u überprüfen): $O(d(u))$, d.h. $O(n)$ im schlimmsten Fall.

D.h. wir haben eine Komplexität von $O(n^2)$, bzw. $O(n \cdot \max(d, \log(n)))$ mit d gleich dem maximalen Grad eines Knotens im Graph (für Strassennetzwerke typischerweise etwa 4). Für grosse Netzwerke mit beschränktem d überwiegt dann der $\log(n)$ Teil.

⚒ Lösung zu Aufgabe 20 ex-dijkstra-mit-todoliste
Die Todo-Liste wird aufsteigend nach der provisorischen Entfernung (anfangs alle ∞) vom Startknoten sortiert.
Zusätzlich brauchen die Knoten zwei Markierungen: Die erste gibt an, ob die Entfernung schon definitiv ist, die zweite gibt an, welcher Knoten auf dem kürzesten Weg der Vorgängerknoten ist.
Das Update der provisorischen Länge auf den Nachbarsknoten muss für alle noch nicht definitiven Knoten gemacht werden, nicht nur für Knoten mit $x(v) = 0$.

⚒ Lösung zu Aufgabe 21 ex-suchalgorithmen
a) Zustand der Todo-Liste (nur Zustände, wo die Liste verlängert wird sind aufgeführt).

1. $\{10\}$
2. $\{6, 7, 8, 9, 11\}$
3. $\{7, 8, 9, 11, 2, 5\}$
4. $\{11, 2, 5, 3, 4, 5\}$
5. $\{3, 4, 5, 12\}$
6. $\{4, 5, 12, 1\}$

Reihenfolge: 10, 6, 7, 8, 9, 11, 2, 5, 3, 4, 12, 1
b) Zustand der Todo-Liste (nur Zustände, wo die Liste verlängert wird sind aufgeführt).

1. $\{10\}$
2. $\{11, 9, 8, 7, 6\}$
3. $\{4, 9, 8, 7, 6\}$

4. $\{3, 1, 9, 8, 7, 6\}$
5. $\{12, 5, 1, 9, 8, 7, 6\}$
6. $\{8, 7, 6\}$
7. $\{2, 7, 6\}$

Reihenfolge: 10, 11, 4, 3, 12, 5, 1, 9, 8, 2, 7, 6

c) Zustand der Todo-Liste (nur Zustände, wo die Liste verlängert wird sind aufgeführt).

1. $\{10\}$
2. $\{6, 7, 8, 9, 11\}$
3. $\{2, 5, 7, 8, 9, 11\}$
4. $\{3, 7, 8, 9, 11, 12\}$
5. $\{1, 4, 7, 8, 9, 11, 12\}$

Reihenfolge: 10, 6, 2, 5, 3, 1, 4, 7, 8, 9, 11, 12

⚒ Lösung zu Aufgabe 22 ex-bfs-dfs

a) Die Knoten werden hinten an die Liste angefügt und vorne entnommen. So werden die Knoten in der Reihenfolge der Distanz abgearbeitet. Werden die Knoten in Distanz k abgearbeitet, werden alle Knoten der Distanz $k + 1$ hinten angehängt und als nächstes abgearbeitet. Dieses Prinzip des Managments einer Liste wird auch FIFO, für «First in, first out» genannt und entspricht einer Warteschlange.

b) Die Knoten werden vorne in die Liste eingefügt und auch wieder vorne entnommen. Dieses Prinzip des Managments einer Liste wird auch FILO, für «First in, last out» genannt und entspricht einem Stack (Stapel). Manchmal wird auch von LIFO gesprochen.

⚒ Lösung zu Aufgabe 23 ex-dfs-rekursiv

Funktion dfs mit Parametern Graph $G = (V, E)$, aktueller Knoten v, Markierungen x auf den Knoten, ob besucht oder nicht

1. Markiere $x(v)$ =besucht.
2. Für alle $u \in N^+(v)$
   (a) Wenn $x(u)$ unbesucht, führe **dfs**(G, u, x) aus.

Je nachdem wird vor oder nach dem Besuch der Nachbarn noch etwas sinnvolles mit dem aktuellen Knoten v gemacht.