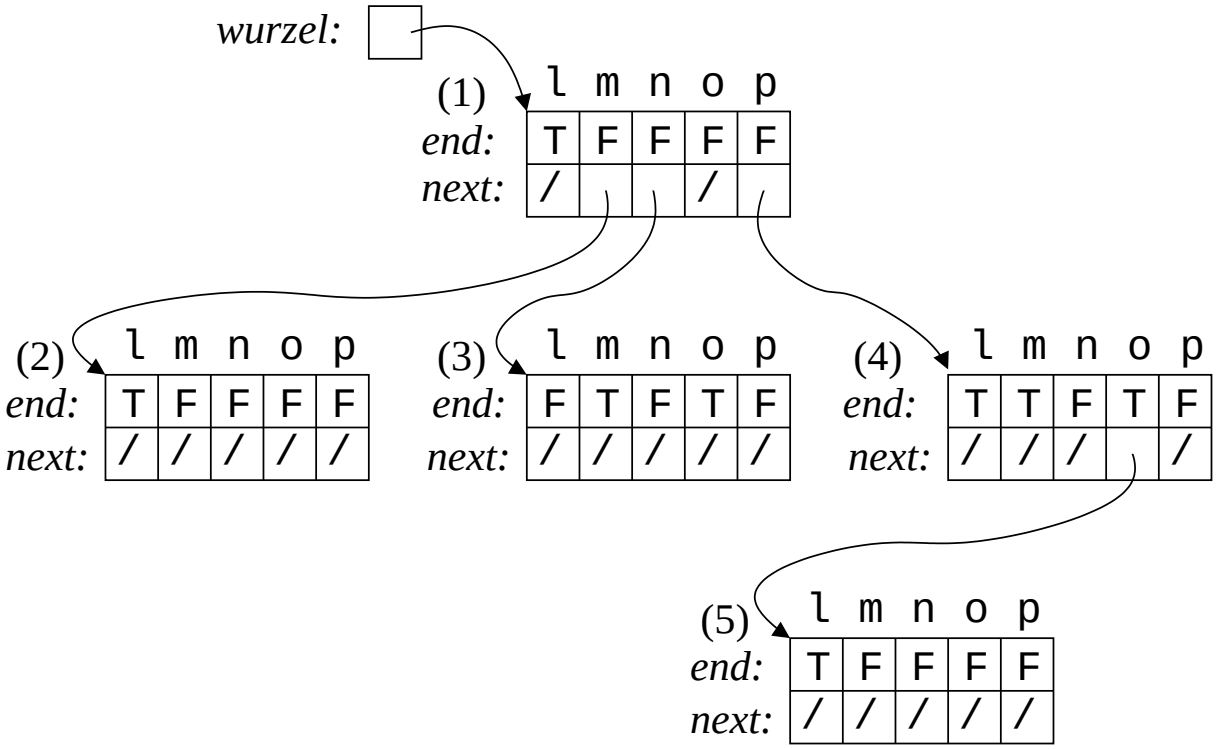
Übungsblatt 4

Aufgabe 4.1

Gegeben seien ein Alphabet Σ={‘l’, ‘m’, ‘n’, ‘o’, ‘p’} und folgender Indexed Trie:



(a) Geben Sie alle Wörter an, die der oben dargestellte Indexed Trie enthält.

(b) Führen Sie Suffix Compression im oben dargestellten Indexed Trie durch. Kennzeichnen Sie die Änderungen deutlich!

(c) Aus dem resultierenden Indexed Trie mit Suffix Compression soll nun ein Packed Trie erstellt werden. Verwenden Sie dazu die Greedy-Heuristik aus der Vorlesung bzw. aus dem Skriptum. Zeigen Sie dabei mit Hilfe einer kleinen Graphik (wie in der Vorlesung bzw. im Skriptum), auf welche Weise die Knoten gepackt werden, und zeichnen Sie den Packed Trie.

Aufgabe 4.2

(a) Gegeben sei eine Menge $A = \{0, 1, \ldots, 9, A, B, C, D, E, F\}$ der Hexadezimalziffern als Schlüsselmenge. Speichern Sie die Hexadezimalziffern $0, 4, 8, 6, 9, A, C, D$ in einem Radix Trie, repräsentieren Sie die Ziffern dabei unter Verwendung der üblichen Binärcodierung: $0 = 0000, 1 = 0001, \ldots, F = 1111$

Fertigen Sie eine Zeichnung des entstehenden Radix Trie an.

(b) Entfernen Sie die Ziffer A aus dem radix Trie und fügen Sie statt dessen die Ziffer 1 ein. Fertigen Sie eine Zeichnung des dadurch entstehenden modifizierten Radix Tries an.

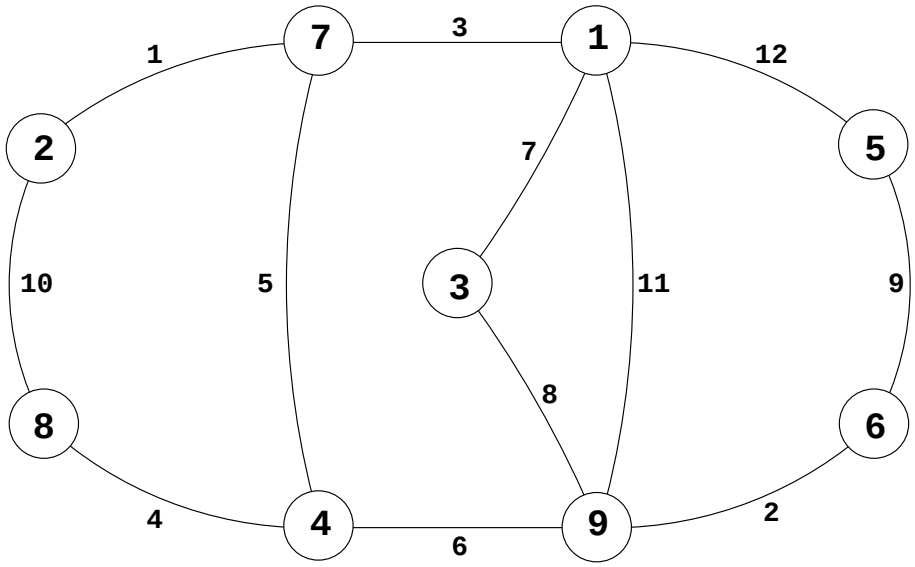
Aufgabe 4.3

Gegeben sei ein zusammenhängender, ungerichteter Graph $G = (V, E)$. Jeder Kante $e \in E$ sei ein Gewicht $w_e \geq 0$ zugeordnet. Außerdem ist folgender Algorithmus gegeben:

Eingabe: Graph $G = (V, E)$ mit Gewichten w_e für alle $e \in E$
Ausgabe: Spannbaum von G

```
1: C[1] = 1;   anzahl = 1;   i = 1;   T = ∅;
2: solange i ≤ |V| {
3:    u = C[i];
4:    F = {(u, v) ∈ E | v ∈ V \ {C[1], ..., C[anzahl]}};
5:    falls F ≠ ∅ dann {
6:       Ermittle Kante e ∈ F mit minimalem Gewicht;
7:       T = T ∪ {e};   anzahl = anzahl + 1;   C[anzahl] = v;
8:    } sonst {
9:       i = i + 1;
10:   }
11: }
```

(a) Wenden Sie diesen Algorithmus auf den nachstehenden Graphen an. Zeichnen Sie den resultierenden Spannbaum deutlich ein und schreiben Sie eine Liste der Gewichte aller Kanten, die Teil des Spannbaums sind, in genau jener Reihenfolge, in der sie in den Spannbaum aufgenommen werden. Geben Sie außerdem das Gesamtgewicht der im Spannbaum enthaltenen Kanten an.



(b) Liefert der Algorithmus einen *minimalen* Spannbaum (MST)?

Wenn ja: Beweisen oder widerlegen Sie, dass der Algorithmus nicht nur bei diesem Beispiel, sondern *immer* einen MST liefert.

Wenn nein: Zeichnen Sie (am besten in einer anderen Farbe oder einer anderen Linienart, z.B. strichliert) von Knoten 1 ausgehend einen MST ein und geben Sie das Gesamtgewicht der darin enthaltenen Kanten an. Beschreiben Sie in einem Satz, wo das Problem liegt.

Aufgabe 4.4

Führen Sie nun anhand des in 4.2 abgebildeten Graphen den Algorithmus von *Kruskal* für das Finden eines minimalen Spannbaums durch. Geben Sie immer, wenn der Algorithmus eine Kante zu dem Baum hinzunimmt, das eindeutige Kantengewicht der Kante an.

Aufgabe 4.5

Führen Sie nun anhand des in Aufgabe 4.2 abgebildeten Graphen den Algorithmus für das Finden des minimalen Spannbaums von *Prim* durch. Beginnen Sie mit Knoten 4.

(a) Geben Sie den Zustand und der in der entsprechenden Tabelle gegebenen Kantenkosten denaller benötigten Datenstrukturen (Knotenmenge, Kantenmenge und Gewicht des aktuellen Spannbaums, Menge der freien Knoten) nach jeder Iteration des Algorithmus an. Schreiben Sie insbesondere in jeder Iteration deutlich den neu hinzugekommenen Knoten und die entsprechende Kante dazu.

(b) Markieren Sie am Ende Ihrer Berechnungen jene Kanten des Graphens in der Abbildung, die den minimalen Spannbaum bilden, und geben Sie das Gewicht des minimalen Spannbaums an.

(c) Wie würde sich das Ergebnis ändern, wenn Knoten 6 der Startknoten wäre? Begründen Sie Ihre Antwort mit wenigen Worten.

Aufgabe 4.6

Gegeben sind zwei Graphen G_1 und G_2 mit jeweils einer Million Knoten. Graph G_1 hat vier Millionen Kanten, während G_2 250 Milliarden Kanten hat. Die Kanten in beiden Graphen haben ganzzahlige Kosten. Sie wollen nun in beiden Graphen jeweils einen aufspannenden Baum mit minimalen Kosten berechnen.

Welchen Algorithmus benutzen Sie für G_1 und welchen für G_2, um möglichst kurze Laufzeiten zu erreichen? Begründen Sie Ihre Antwort, indem Sie die Laufzeiten der von Ihnen verwendeten Algorithmen angeben. Dabei sollten Sie auch angeben, warum die Algorithmen die von Ihnen angegebenen Laufzeiten aufweisen (kurze Beschreibung der Algorithmen und der Funktionsweise und Eigenschaften der verwendeten Datenstrukturen).

Aufgabe 4.7

Benutzen Sie die *Union-Find-Datenstruktur*, die Sie in der Vorlesung bzw. im Skriptum kennen gelernt haben, um einen Algorithmus in kommentiertem Pseudocode anzugeben, der die Zusammenhangskomponenten eines ungerichteten Graphen $G = (V, E)$ identifiziert. Dabei soll jedem Knoten eine Zahl zugewiesen werden, wobei Knoten in der gleichen Komponente die gleiche Zahl haben müssen.

(a) Wie oft ruft Ihr Algorithmus die Funktionen *findset* und *union* für einen Graphen mit k Komponenten auf?

(b) Wie ist die Laufzeit Ihres Algorithmus in Θ–Notation, abhängig von $|V|$, $|E|$ und k?

(c) Wie verhält sich die Laufzeit Ihres Algorithmus zu der Realisierung, die DFS benutzt?

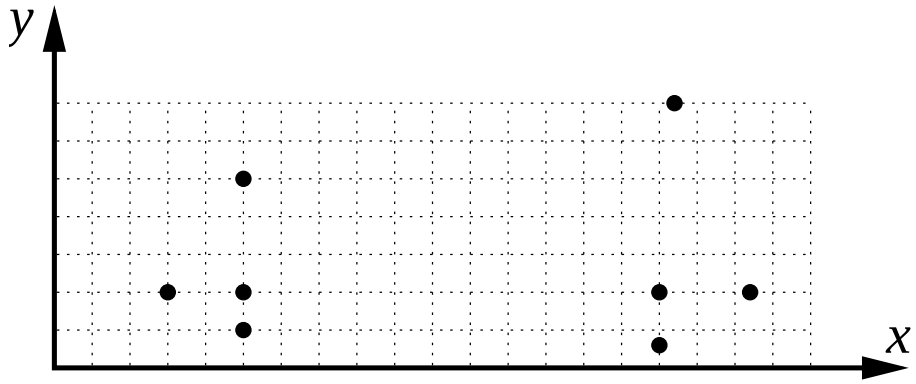
Aufgabe 4.8

Gegeben sei ein vollständiger, ungerichteter Graph $G = (V, E)$. Jeder Kante $e \in E$ sei ein Gewicht $w_e \geq 0$ zugeordnet.

Ein *Grad-3-beschränkter Spannbaum* von G ist ein Spannbaum von G für den gilt, dass der Grad eines jeden Knoten maximal drei ist. Ein *minimaler* Grad-3-beschränkter Spannbaum $T \subseteq E$ ist ein Grad-3-beschränkter Spannbaum mit minimalem Gesamtgewicht $w(T) = \sum_{e \in T} w_e$.

(a) Beschreiben Sie in einfachem Pseudocode (ohne auf eventuelle Datenstrukturen genauer einzugehen) einen Algorithmus, der auf *Prims* Algorithmus zum Finden eines minimalen Spannbaums basiert und immer einen gültigen Grad-3-beschränkten Spannbaum mit möglichst geringem, aber nicht unbedingt minimalem, Gewicht zurückliefert.

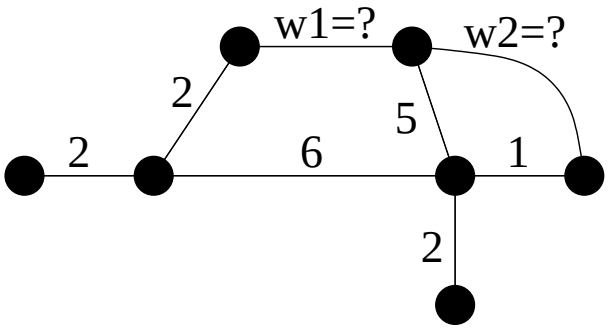
(b) Die in der folgenden Zeichnung gezeigten acht Punkte seien die Knoten eines Graphen G. Kanten existieren zwischen allen Paaren von Punkten, und die Euklidischen Abstände sind die Kantenkosten.



Wenden Sie Ihren Algorithmus für den Grad-3-beschränkten Spannbaum auf diesen Graphen an, wobei Sie mit dem Knoten mit kleinster x-Koordinate als Startknoten beginnen. Zeichnen Sie in die Zeichnung alle Kanten ein, die in den Spannbaum übernommen werden, und nummerieren Sie diese in der entsprechenden Reihenfolge.

(c) Da das Finden eines Grad-3-beschränkten Spannbaums im Allgemeinen NP-schwierig ist, können Sie davon ausgehen, dass Ihr Algorithmus nicht immer einen minimalen Grad-3-beschränkten Spannbaum zurückliefert. Das sollen Sie nun mit einem Beispiel beweisen.

Nennen Sie für den folgenden nicht-Euklidischen Graphen konkrete Beispielwerte für die zwei fehlenden Kantengewichte $w1$ und $w2$, sodass Ihr Algorithmus sicher einen gültigen, aber nicht minimalen Grad-3-beschränkten Spannbaum zurückliefert, wenn der Knoten mit Grad 4 der Startknoten ist. Erklären Sie die Situation auch mit einem Satz. (Um entsprechend der Definition von einem vollständigen Graphen auszugehen, nehmen Sie einfach an, dass alle nicht eingezeichneten Kanten Gewicht ∞ haben.)



Aufgabe 4.9

(a) Gegeben sei ein ungerichteter zusammenhängender Graph $G = (V, E)$. Gesucht ist ein *Hamiltonscher Pfad*, d.h., ein Pfad der jeden Knoten des Graphen genau einmal besucht. Ausgangs- und Endknoten können dabei beliebig sein.

Lösen Sie diese Aufgabe mit begrenzter Enumeration. Geben Sie dazu den Algorithmus in Pseudocode an.

(b) Geben Sie den Worst-Case Aufwand Ihres Algorithmus in O-Notation an und begründen Sie Ihre Antwort.

Aufgabe 4.10

Beim *fraktionalen Rucksackproblem* darf man im Gegensatz zu dem in der Vorlesung vorgestellten *0/1-Rucksackproblem* nicht nur ganze, sondern beliebig große Bruchteile von Gegenständen einpacken. Geben Sie den Pseudocode für einen Greedy-Algorithmus an, um das fraktionale Rucksackproblem zu lösen. Wie ist die Laufzeit Ihres Algorithmus in Θ-Notation? Liefert Ihr Greedy-Algorithmus immer eine optimale Lösung?

Aufgabe 4.11

Geben Sie für die Brettgrößen von 4x4 bis 10x10 die Anzahl der möglichen Lösungen für das jeweilige Damenproblem an und skizzieren Sie bei drei beliebigen verschiedenenen Brettgrößen diese. Verwenden Sie zur Lösung dieser Aufgabe im Internet angebotene Mittel, geben Sie die Quelle dieser an.

Aufgabe 4.12

Gegeben sei folgendes Rucksackproblem mit 5 Gegenständen:

i	Wert c_i	Gewicht w_i
1	4	2
2	2	1
3	6	5
4	4	3
5	4	3

Das maximal erlaubte Gesamtgewicht der auszuwählenden Gegenstände sei $K = 5$.

Führen Sie den in der Vorlesung besprochenen Algorithmus zur Dynamischen Programmierung an diesem Beispiel durch.

Behandeln Sie dabei der Reihe nach die Teilprobleme $l = 1, \ldots, 5$, in denen jeweils die Gegenstände 1 bis l berücksichtigt werden. Geben Sie für jedes Teilproblem die Menge M_l aller im weiteren Verlauf *noch zu berücksichtigender* Tripel $(S, c, b_l(c))$ an. S ist hierbei die Menge der eingepackten Gegenstände, c ist der Gesamtwert und $b_l(c)$ ist das Gesamtgewicht dieser Konfiguration. Sollten Tripel mit identen c und $b_l(c)$ auftreten, so behalten Sie das erste und verwerfen die neue Konfiguration, die Gegenstand l beinhaltet.

Nennen Sie abschließend eine optimale Auswahl an Gegenständen.

Aufgabe 4.13

Sei $G = (V, E)$ ein gewichteter Graph, sei $t \in V$ ein frei gewählter Zielknoten. Gesucht sind die kürzesten Wege von allen anderen Knoten aus G nach t.

Geben Sie einen auf dynamischer Programierung basierenden Algorithmus in Pseudocode für das oben beschriebene kürzeste Wege Problem an.

Hinweis:
Beginnen Sie mit dem Zielknoten, und berechnen Sie dann rekursiv die kürzesten Wege.

Aufgabe 4.14

Für die 5 Beispiele auf dem Übungsblatt zu einer besonders anspruchsvollen Speziallehrveranstaltung aus „Algorithmen und Datenstrukturen" erhalten Sie jeweils eine unterschiedliche Anzahl von Bonuspunkten, aber die Beispiele verursachen natürlich auch einen unterschiedlich hohen Zeitaufwand. In der folgenden Tabelle sind diese Daten angegeben:

Beispiel	Bonuspunkte	Zeit in Minuten
Algorithmus A	7	40
Algorithmus B	3	10
Algorithmus C	10	60
Algorithmus D	4	20
Algorithmus E	13	90

Leider sind Sie wieder einmal spät dran und haben nur noch drei Stunden Zeit, bis die Übung beginnt.

Geben Sie einen Branch-and-Bound-Algorithmus für das Problem in Pseudocode an.

Aufgabe 4.15

Wenden Sie Ihren Algorithmus aus Aufgabe 4.14 auf die dort angegebene Beispielinstanz an und veranschaulichen Sie dabei alle Schritte auf dem Weg zur Lösung.
