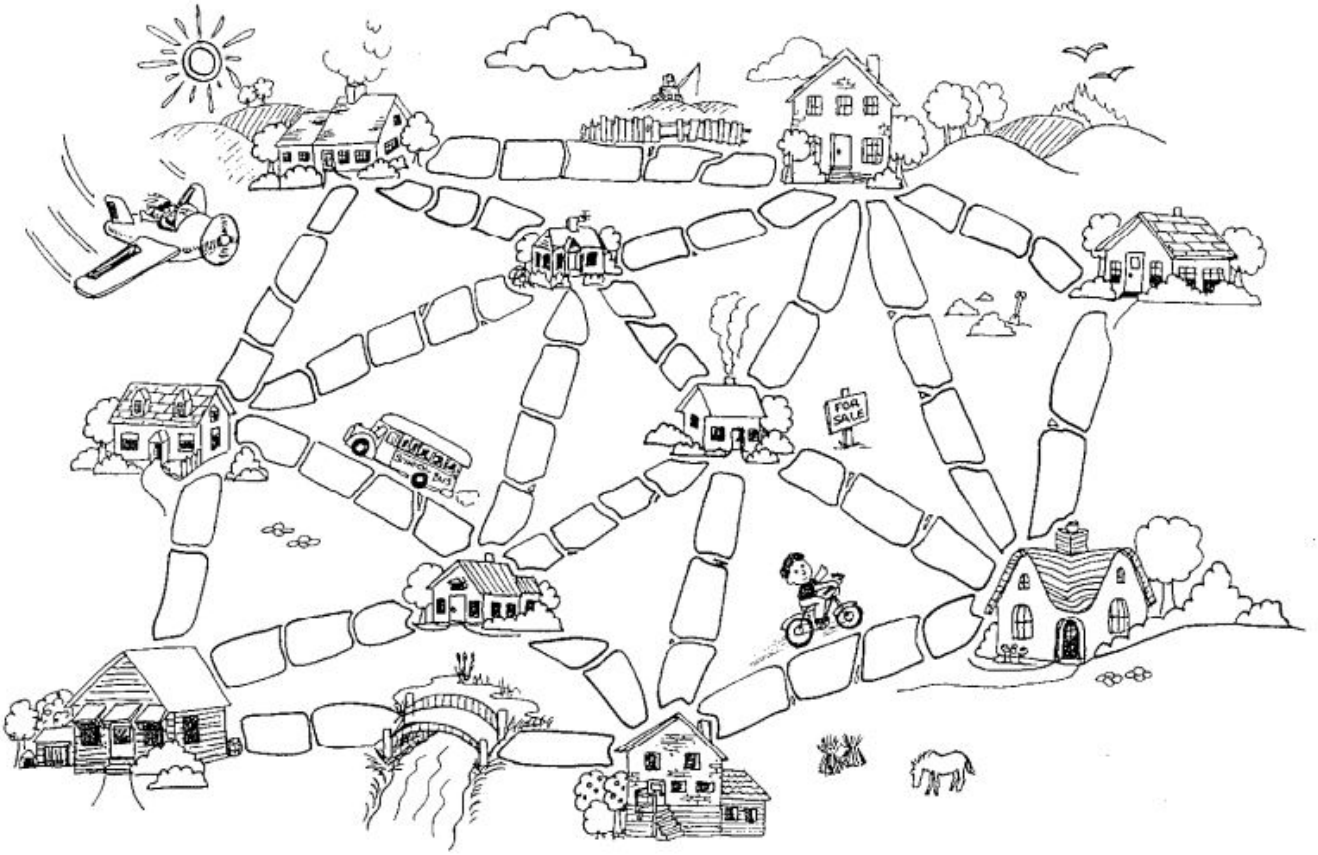
Minimalspannende Bäume

Muddy-City

Vor langer Zeit gab es eine Stadt, die keine Straßen hatte. Sich in dieser Stadt zu bewegen war insbesondere dann schwierig, wenn der Regen nach starken Gewittern die Wege sehr matschig machte. Fahrzeuge blieben stecken und jeder hatte ziemlich dreckige Schuhe. Daher entschied der Bürgermeister, einige Straßen zu asphaltieren. Allerdings wollte er nicht mehr Geld ausgeben als unbedingt nötig, damit auch noch ein Schwimmbad gebaut werden konnte.



1)

Also legte er zwei Bedingungen fest:

- Es müssen ausreichend Straßen asphaltiert werden, damit jeder von seinem Haus jedes andere Haus trockenen Fußes erreichen kann.
- Das Asphaltieren soll so wenig kosten wie möglich. Die Anzahl der Pflastersteine ist das Maß für die Kosten.

Aufgaben zum Muddy-City-Problem

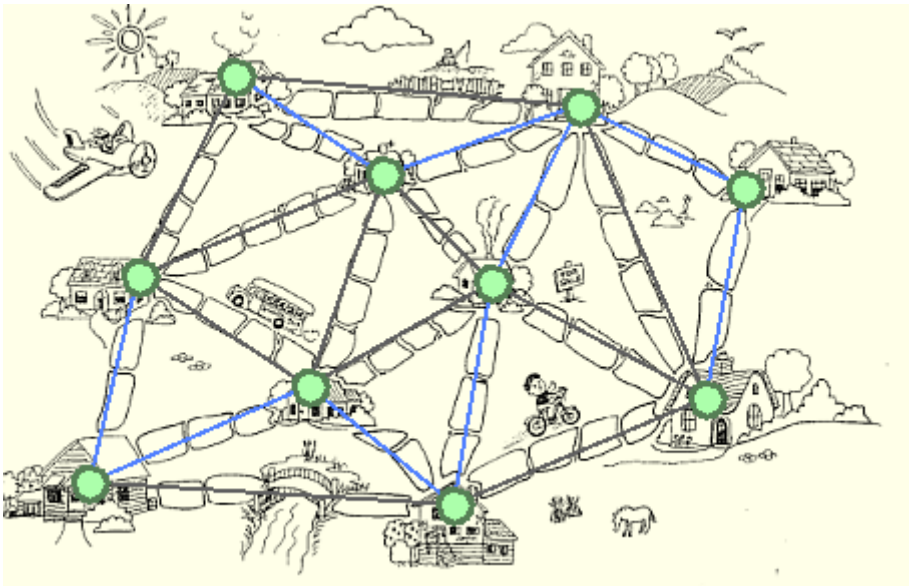
(A1)

Finde die Wege, die alle Häuser miteinander verbinden und deren Asphaltierung am wenigsten kostet (=Pflastersteine beinhaltet).

Tipp

Die optimale Lösung besteht aus 23 Pflastersteinen.

Lösung



(A2)

Welche Strategien hast du genutzt, um das Problem zu lösen?

Das Minimaler Spannbaum Problem (Minimal spanning tree)

Gegeben ist ein zusammenhängender, gewichteter, ungerichteter Graph.

Gesucht ist die Teilmenge der Kanten, so dass der Graph zusammenhängend bleibt und die Summe der Kantengewichte möglichst klein ist.

Vertiefung: Fragen & Aufgaben

(A3)

Wende den Algorithmus "MST (Prim)" zur Bestimmung des minimalen Spannbaums im Graphentester auf die Karte der größten Städte in Deutschland an (`02_deutschlandkarte.csv` im Ordner `08_minimalspanningtree`). Beachte, dass nur markierte Kanten sichtbar sind, da es zu viele Kanten gibt, um sie alle übersichtlich darzustellen. Vergleiche mit dem Autobahnnetz in Deutschland. Nenne Ursachen für Unterschiede.

(A4)

Öffne den Stadtplan von Baden-Baden (`03_badenbaden.csv`) und die Karte mit den Fährstrecken (`04_inseln.csv`) jeweils im Graphentester. Lass dir auch hier mit dem Algorithmus `MST (Prim)` einen minimalen Spannbaum anzeigen (Achtung, die Kanten werden ungünstigerweise in blau gefärbt und sind kaum zu sehen). Gib je eine realitätsnahe Problemstellung für diese Graphen an, bei denen ein minimaler Spannbaum die beste Lösung ist.

Anders ausgedrückt: Für was bräuchte man in der Realität jeweils so einen minimalen Spannbaum? Für was für Probleme oder Anwendungen kann das relevant sein?

Lösungsvorschläge

- Stadtplan: z. B. Wasserrohre, Gasleitungen, Netzwerkkabel verlegen.
- Inselkarte: Beschränkung auf Fährverbindungen mit einer möglichst kurzen Gesamtstrecke, so dass trotzdem alle Inseln verbunden sind.

Zwei Algorithmen

Wir betrachten zwei Algorithmen zur Lösung des Problems des minimal spannenden Baums:

- Algorithmus von Kruskal[2)]
- Algorithmus von Prim[3)]

Weitere Fragen und Aufgaben

(A5)

Untersuche, ob die Algorithmen zur Bestimmung des minimalen Spannbaums auch mit negativen Kantengewichten zurechtkommen.

Lösung

Negative Kantengewichte stellen kein Problem dar, da es beim Algorithmus nur auf die Sortierreihenfolge der Kanten ankommt und nicht auf den Wert des Gewichts. Diese Sortierung funktioniert auch bei negativen Kantengewichten.

(A6)

Begründe, warum es sich bei den Algorithmen zur Bestimmung des minimalen Spannbaums um Greedy-Algorithmen handelt.

Lösung

Kruskal ist ein Greedy-Algorithmus, da in jedem Schritt die günstigste Kanten ausgewählt wird, die nicht zu einem Zyklus führt.

Prim ist ein Greedy-Algorithmus, da immer die kürzeste Kante gewählt wird, um den Baum zu erweitern.

Näherungslösung für das TSP mit MST Algorithmen

(A7)

Beim Traveling Salesman Problem (TSP) sucht man nach der kürzesten Route durch eine gegebene Liste von Städten, die ein Handlungsreisender besuchen muss. Am Ende soll er wieder zu Hause ankommen.

Für dieses Problem kennt man keinen effizienten Algorithmus. Man kann aber die Algorithmen zur Bestimmung eines MST abändern, so dass sie eine Näherungslösung für das TSP darstellen.

Beschreibe die Veränderungen die am Algorithmus für den MST notwendig sind, um das TSP zu lösen. Du kannst dazu im Graphentester die Algorithmen `TSP (Greedy: Knoten)` und `TSP (Greedy:`

`kürzeste Kante)` zu Hilfe nehmen.

Erläuterung zur Lösung

Der Prim-Algorithmus vereinfacht sich, da die kürzeste Kante von einem markierten zu einem unmarkierten Knoten für das TSP nicht im ganzen bisherigen Baum gesucht werden muss, sondern am Ende der Route liegen muss. Man muss also nur die ausgehenden Kanten des Routenendes zu allen noch nicht markierten Knoten betrachten (vgl. TSP-Greedy im Graphentester). Am Ende wird die Rundreise durch eine Kanten zwischen den beiden Blättern des Baums geschlossen.

Der Kruskal-Algorithmus wird etwas komplizierter, da man beim Zusammenführen zweier Teilbäume oder Vergrößern eines Teilbaums darauf achten muss, dass keine innere Knoten sondern nur Blätter verwendet werden dürfen. Außerdem muss zum Schließen der Rundreise am Ende einmal erlaubt werden, einen Zyklus zu bilden.

1)
Quelle: Minimal Spanning Tree Activity, csunplugged.org (Lizenz: CC BY-NC-SA 4.0), URL: https://classic.csunplugged.org/minimal-spanning-trees/#The_Muddy_City
2)
https://de.wikipedia.org/wiki/Algorithmus_von_Kruskal
3)
https://de.wikipedia.org/wiki/Algorithmus_von_Prim