Algorithmen und Datenstrukturen

Graphen

Begriffsdefinitionen

- Ein *Graph* besteht aus Knoten und Kanten.
- Ein *Knoten(Ecke)* ist ein benanntes Objekt.
- Eine *Kante* verbindet zwei Knoten.
  Kanten haben ein Gewicht und Richtung.
- *Ungerichtet* Graphen haben für jede Kante eine Kante in die Gegenrichtung.

Begriffsdefinitionen

- Ein *Pfad von A nach B* ist eine Liste von Knoten, die mit A beginnt und mit B enden und durch Kanten schrittweise verbunden sind.
- Ein Graph heißt *zusammenhängend* wenn Pfade für alle Knotenpaare existieren.
- *Einfache Pfade* haben keine Knoten mehrfach.
- Ein *Zyklus* ist ein einfacher Pfad von einem Knoten zu sich selbst.
- Ein *Baum* ist ein zyklenfreier Graph.

Begriffsdefinitionen

- Ein *Spannbaum* ist ein Teilgraph, der alle Knoten enhält aber nur die Kanten, die zum Zusammenhang nötig sind.
- *Vollständige* Graphen haben alle möglichen Kanten.
- Einem *dichter* Graph fehlen wenige Kanten.
- Ein *lichter* Graph hat wenige Kanten.
- *Netzwerke* sind gerichtete, gewichtete Graphen.

Darstellung

- Knoten werden durch Punkte dargestellt
- Kanten werden durch Linien dargestellt
- Schreibweise für Knoten sind Großbuchstaben
- Schreibweise für Kanten sind Buchstabenpaare
- Implementation als Adjazenzmatrix (dichte)
  - Zweidimensionales Feld indiziert mit den Knoten
- Implementation als Adjazenzliste (lichte)
  - Pro Knoten eine Liste von Nachbarknoten

Elementare Algorithmen

- Ist ein Graph zusammenhängend?
  Enthält ein Graph Zyklen?
  Wie verläßt man ein Labyrinth?
- Durchmustern des Graphen
  - Tiefensuche rekursiv oder mit Stack
  - Breitensuche mit Queue (Warteschlange)
  - Generelle Traversierungen mit Heap (Prioritätswarteschlange)

Zusammenhang

- Auffinden von zusammenhängenden Komponenten durch (Tiefen)suche mit Markierung der besuchten Knoten
- Durchmustern aller Knoten, nicht erreichte Teile (Beginn einer neuen Komponente) aufzufinden.
- Oft vorbereitender Schritt, um anderen Algorithmen einen Zusammenhang zu garantieren.

Zweifacher Zusammenhang

- Tiefensuche, dabei den Level mitschreiben.
- Beim rekursiven Abstieg erfassen, welcher höchste Knoten im Suchbaum durch Rückkanten hätte erreicht werden können.
- Ist beim Aufstieg ein Knoten von seinen Kindern im Suchbaum nicht zu übertreffen, so ist dies ein *Gelenkpunkt* und es besteht nur ein einfacher Zusammenhang.
- Praktischer Test für Redundanzfragen.

Vereinigungssuche

- Gehören zwei Knoten zu gleichen zusammenhängenden Komponente?
- Aufbau eines aufwärtslinkenden Baumes
- Schrittweise alle Kanten hinzufügen und pro Knoten, den Startknoten des anderen übernehmen
- Beim Mergen entstehen Bäume, diese immer mit einem extra Durchlauf zum Startknoten vereinfachen (Pfadverdichtung)
- Aufbau in O(n), dann O(1) für Test

Minimaler Spannbaum

- Gesucht ist der Spannbaum des geringsten Gesamtgewichtes. Einsatz z.B. in redundanten Rechnernetzwerken.
- Prioritätssuche nach dem nächstleichten Knoten
- Bewege den günstigsten Knoten vom *Rand* in den *Baum* und nimm alle *unsichtbaren* Knoten, die dem verschobenen Knoten benachbart sind, in den Rand auf.
- Viele Probleme nur in der Priorität verschieden

Minimaler Spannbaum

- Völlig anderer Ansatz von **Kruskal**
- Für jede Zerlegung des Graphen in zwei Teile enthält der minimale Spannbaum die kürzeste Kante, die die Teile untereinander verbindet.
- Prioritätswarteschlange oder Sortierung aller Kanten nach Gewicht
- Nur Kanten aufnehmen, die keine Zyklen bilden
- Laufzeit O(E log E)

Kürzester Pfad

- Gesucht: Kürzeste Verbindung zwischen A und B
- Gefunden: Alle kürzesten Verbindungen von A
- Prioritätssuche wie beim minimalen Spannbaum
- Unterschied: Korrektur des Randes nach Abstand von A
- Algorithmus von **Disjkstra**
- Geographische Daten gestatten Graphreduktion
- Anwendung für Navigationsgeräte, Internetrouter

Gerichtete Graphen

- Algorithmen grundsätzlich gleich
- Vorsicht bei der Wahl des Startpunkts
- Neue Frage nach der *transitiven Hülle* statt eines Zusammenhangs
- Markierungen der Knoten sind nun nicht mehr per se zulässig, sondern abhängig von der Richtung aus der man kam.

Topologisches Sortieren

- Voraussetzung: Zyklenfreiheit (*DAG*)
- Lösung: Postorder Tiefensuche
- Vorsicht: Liefert die umgekehrte Sortierung
- Anwendungen:
  - Erfüllung von Abhängigkeiten bei serieller Arbeit
  - Fertigungsprozesse
  - Erstellung von Studienplänen

Fluß in Netzwerken

- Aufgabe: Maximaler Durchsatz durch einen gerichteten, gewichteten Graphen
- Lösung: Betrachtung aller (nicht nur einfache) Pfade, dadurch auch „Rückwärtskanten“
- Falls jeder mögliche Pfad von der Quelle zur Senke eine volle Vorwärts oder eine leere Rückwärtskante hat, ist der Fluß maximal.
- Prioritätssuche nach maximaler Erhöhung
- Algorithmus von **Ford-Fulkerson**

Paarung

- Aufgabe: Bipariter Graph (z.B. Präferenzlisten von Studenten zu Arbeitsplätzen) maximal paaren
- Lösung: Eine Quelle vor alle oberen Knoten (Studenten) legen und eine Senke hinter alle unteren Knoten (Arbeitsplätze)
- Ford-Fulkerson zur Flußmaximierung

Paarung

- Problem der stabilen Ehe
- Gegeben: Präferenzlisten (Reihenfolgen)
- Der Reihe nach wählt eine Gruppe. Jeder auf seinen persönlichen Vorteil bedacht. Wenn gewähle Person bereits liirt ist, aber den neuen Bewerber vorzieht, wird getauscht.
- In den Präferenzlisten wird nicht zurückgegangen Deswegen terminiert das Verfahren.