Graphen

Definition:

Ein Graph ist ein Paar (V,E), wobei V eine Menge von Knoten und E eine Menge von Kanten (v,w) mit v,w in V ist.

Begriffe:

Gerichteter Graph: Alle Kanten haben eine Richtung vom Anfangsknoten zum Endknoten.

Ungerichteter Graph: Kanten haben keine Richtung bzw. jede Kante existiert in beiden Richtungen (Spezialfall des gerichteten Graphen).

Schlinge: Eine Kante (v,v).

Kantenfolge: Eine endliche Folge von Kanten, so dass der Endknoten des Vorgängers der Startknoten des Nachfolgers ist. Die Kantenfolge heißt geschlossen, wenn der Endknoten des letzten Folgenelements der Startknoten des ersten Knotenelements ist, ansonsten offen.

Weg: Offene Kantenfolge, in der alle Knoten verschieden sind.

Kreis: Geschlossene Kantenfolge, in der alle Knoten verschieden sind.

Zyklus: gerichteter Kreis

Azyklischer Graph: Gerichteter Graph ohne Zyklus

Speicherung von Graphen:

Dicht besetzt:

- **Adjazenzmatrix**
- **Inzidenzmatrix**

Dünn besetzt:

- **Kantenliste**
- **Knotenliste**

Graphenalgorithmen

Ablaufen:

1. **Sei N1 die Menge der jeweils noch zu verarbeitenden Knoten; initialisiere N1 mit V.**
2. **Sei N2 die Menge der Knoten, die schon verarbeitet sind; initialisiere N2 mit der leeren Menge.**
3. **Solange N1 nicht leer ist, führe folgende Schritte aus:**
4. **Bewege einen Knoten von N1 nach N2.**
5. **Solange N2 nicht leer ist, führe folgende Schritte aus:**
6. **Wähle einen Knoten aus N2 und lösche ihn aus N2.**
7. **Verarbeite diesen Knoten.**
8. **Füge die Nachfolger dieses Knotens in N2 ein.**

Falls N2 als Stack organisiert ist, führt dieser Algorithmus zur Tiefensuche (Depth first search), falls N2 als Queue organisiert ist, führt dieser Algoithmus zur Breitensuche (Breadth first search).

Durch die Organisation von N2 als geeignete Prority Queue lassen sich aus dem Algorithmus einige interessante Graphenalgorithmen ableiten:

Topologisches Sortieren:

Unter Topologischem Sortieren eines gerichteten Graphen versteht man eine Anordnung der Knoten, so daß für alle Knotenpaare (v(i),v(j)) mit I<j keine Kante von v(j) nach v(i) existiert.

Topologisches Sortieren ist nur für azyklische gerichtete Graphen möglich.

Zum Topologischen Sortieren eines gerichteten Graphen kann man den Algorithmus zum Ablaufen verwenden, wobei N2 als Priority Queue organisiert ist und als Priorität die Anzahl der Vorgänger, die noch nicht in der Topologischen Ordnung vorkommen, gewählt wird. Die Priorität muß im Algorithmus in Schritt 8 angepaßt werden für die Nachfolger des Knotens, der in Schritt 7 verarbeitet wird.

Falls alle Knoten in N2 positive Priorität haben, ist der Graph nicht azyklisch!

Datenstruktur für die Priority Queue:

Falls die Prioritätsschlange als Heap organisiert ist, muß für jede Kante des Graphen eine Neuberechnung der Priorität und die Neupositionierung in der Priority Queue erfolgen. Daher ergibt sich die Zeitkomplexität des Algorithmus als #E * log(#V) und damit O(n^2*log(n)) im Falle eines dicht besetzten Graphen.

Falls die Prioritätsschlange als unsortierte Liste organisiert ist, können die Neuberechnung der Priorität und das Auffinden des Minimums in einem Schritt durchgeführt werden. Dies bedeutet, dass die Zeitkomplexität des Algorithmus im falle eines dicht besetzten Graphen O(n^2) ist.

Die Priority Queue kann als ungeordnete Liste der Knoten mit Priorität 0 organisiert werden, wobei zusätzlich eine Folge verwaltet wird, in der zu jedem Knoten die Priorität gespeichert wird.

Kürzeste (und längste) Wege Probleme

Topologisches Sortieren eines gerichteten Graphs kann als Vorbereitung für das Lösen einiger interessanterer Probleme verwendet werden:

Bestimmung des kürzesten Wegs von einem Startknoten zu allen Endknoten in einem azyklischen gerichteten Graphen. Der Algorithmus lässt sich auch in dem Fall verwenden, dass der Graph ungerichtet ist und die Kanten alle ein positives Gewicht haben.

Bestimmung des länsten Wegs von einem Startknoten zu allen Endknoten in einem azyklischen gerichteten Graphen. Die Lösung dieses Problems ist die Grundlage der CPM Methode (critical path method), die im Zeitmanagement von Projekten angewandt wird. Um nicht nur den spätesten Endzeitpunkt für sämtliche Aktivitäten zu bestimmen, sondern auch die maximalen Verzögerungen, die das Projektende nicht gefährden, muß der Algorithmus zunächst vorwärts und dann rückwärts angeandt werden.

Falls der Graph topologisch sortiert ist und der beste Weg vom Startknoten zum Knoten v(i) über alle Knoten v(j) mit j<I bestimmt ist, so kann es keinen besseren Weg geben, da nach Definition der topologischen Sortierung keine Kante von v(j) nach v(i) mit j>I existiert.

Daher kann der Algorithmus zur Lösung des kürzeste (längste) Wege Problems in einem azyklischen gerichteten Graphen wie folgt formuliert werden:

1. **Bestimme die topologische Ordnung der Knoten, v(0), ..., v(n-1).**
2. **Für alle Knoten v(1), ..., v(n-1) wird die beste Weglänge als möglichst schlecht initialisiert.**
3. **Für alle Knoten v(i), i=0,…,n-1, sind die folgenden Schritte durchzuführen:**
4. **Die beste Weglänge nach v(i) und der Vorgänger auf dem Weg von v(0) nach v(i) werden festgehalten.**
5. **Für jeden Nachfolger v von v(i) wird Schritt 6 ausgeführt:**
6. **Falls es einen besseren Weg von v(0) nach v über v(i) gibt als der bisher bestimmte beste Weg, so wird die Weglänge und v(i) als Vorgänger von v auf dem bisher bestimmten besten Weg festgehalten.**

Bemerkungen:

1. **Die Lösung des single source – longest path problem basiert auf der Tatsache, daß azyklische gerichtete Graphen betrachtet werden. Dies gilt nicht für das single source – longest path problem; hier genügt die Voraussetzung, daß jede Kante ein nichtnegatives Gewicht hat. Zur Lösung des Problems kann in diesem Fall der Algorithmus zum Ablaufen eines Graphen angewandt werden, wobei N2 als Priority Queue mit der Länge des besten bisher gefundenen Wegs vom Startknoten zu dem Knoten als Priorität oranisiert ist. Diese Technik der Problemlösung wird als "greedy" und funktioniert, weil die Kanten des Graphen alle ein nichtnegatives Gewicht haben. Der Algorithmus ist als Algorithmus von Dijkstra bekannt. Die Zeitkomplexität des Algorithmus ist quadratisch für einen dicht besetzten Graphen und O(#E * log(#V)) für dünn besetzte Graphen.**

2. **Das all source – shortest path problem für Graphen mit nichtnegativen Kantengewichten wird durch den Algorithmus von Floyd gelöst, der auf der Idee basiert, daß man für jeden Zwischenknoten v und jedes Paar von Start- und Endknoten (v(i),v(j)) untersucht wird, ob es einen kürzeren Weg von v(i) nach v(j) über v gibt als den bisher bestimmten kürzesten Weg; falls ja, wird für (i,j) die Weglänge gespeichert und v als Vorgänger von v(j) auf dem bisher kürzesten Weg von v(i) nach v(j). Dieser Algorithmus hat eine kubische Zeitkomplexität, ist aber im Falle eines dicht besetzten Graphen effizienter als die Anwendung des Algorithmus von Dijkstra für alle Startknoten, da der Algorithmus sehr einfach ist. Im Falle eines dünn besetzten Graphen ist die Anwendung des Algorithmus von Dijkstra für alle Startknoten die effizientere Lösung des all source – shortest path problem.**

3. **Der Algorithmus von Warshall Basiert auf den gleichen Ideen und lost das Problem der Erreichbarkeit in einem Graphen.**

4. **Der Algorithmus kann modifiziert werden, um Zyklen in einem Graphen zu erkennen.**

Beispiel:

	V1	V2	V3	V4	V5	V6	V7	V8	V9
V1	4					7			
V2			9	2					
V3									5
V4					8				
V5			2					2	3
V6				3			6		
V7					7			8	
V8									4
V9									