

Tutorium 8

Algorithmen I SS 14

Institut für Theoretische Informatik



Graphtraversierung

- Funktionsweise:
 - ① Beginne bei einem Startknoten s
 - ② Besuche alle (unbesuchten) Nachbarknoten von s
 - ③ Besuche die Nachbarknoten dieser Knoten
 - ④ ...
- Implementierung über Queue
- Graph wird „Schichtenweise“ durchgegangen (Abstand zum Startknoten)

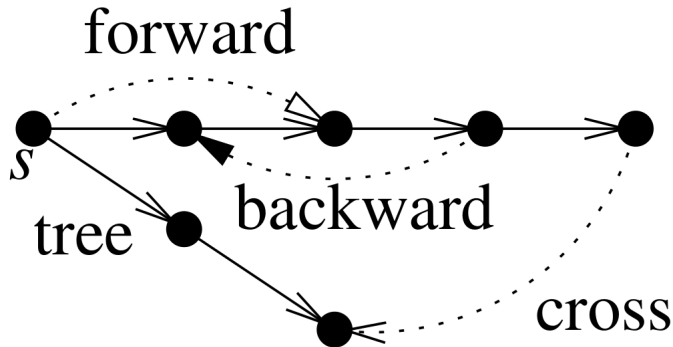
- Ein Graph $G = (V, E)$ heißt bipartit : $\Leftrightarrow \exists A, B \subseteq V : A \cap B = \emptyset, A \cup B = V \wedge \forall (u, v) \in E : (u, v) \in ((A \times B) \cup (B \times A))$
- Also: die Knoten lassen sich in zwei Mengen aufteilen, so dass es keine Kanten innerhalb der Knoten in einer Menge gibt
- Aufgabe: Entwickle einen Algorithmus, der für einen stark zusammenhängenden Graphen eine bipartite Zerlegung findet (oder erkennt, dass keine existiert). Laufzeit: $\mathcal{O}(|E|)$

- Funktionsweise:

- ➊ Beginne bei einem Startknoten s
- ➋ Besuche den ersten (unbesuchten) Nachbarknoten von s
- ➌ Besuchen den ersten Nachbarknoten dieses Knotens
- ➍ ...
- ➎ Besuche den zweiten Nachbarknoten von s
- ➏ ...

- Intuitiv rekursiv

- Iterative Implementierung?



- Sortiere die Knoten eines Graphen nach der Relation $a \leq b :\Leftrightarrow$ Es gibt einen Pfad von a nach b
- Lösung über modifizierte Tiefensuche:
 - ① Speichere „Zeitpunkt“ des Knotenabschlusses (keine Nachbarn mehr)
 - ② Falls Rückwärtskante gefunden wird \Rightarrow Abbruch
 - ③ Sortiere die Knoten absteigend nach Abschlusszeit
- Aufgabe: Wie kann Breitensuche modifiziert werden, um eine topologische Sortierung zu berechnen?

Wiederholung

- O-Kalkül, Mastertheorem
- Invarianten, Korrektheit von Algorithmen
- Verkettete Listen (einfach und doppelt)
- Unbounded Arrays
- Stack, Queue, Deque
- Hashing, Hashtabellen
- Sortieren
 - Insertionsort
 - Mergesort
 - Quicksort (Quickselect)
 - Heapsort
 - Bucketsort
 - Radixsort
- Heaps
- (Binäre) Suchbäume
- Graphen

Zeigen oder widerlegen Sie, dass $2^{3n} = \mathcal{O}(5^n)$

Bestimmen Sie die Lösung der folgenden Rekurrenz im Θ -Kalkül mit dem Master-Theorem:

$$T(1) = 5, T(n) = n + 2S(n/2), n = 2^k, k \in \mathbb{N}$$

Nennen Sie zwei Operationen, die doppelt verkettete Listen in konstanter worst-case-Zeit unterstützen, einfach verkettete Listen jedoch nicht.

Nennen Sie einen Vorteil und einen Nachteil von einfach verketteten Listen gegenüber unbeschränkten Feldern.

Wie lautet die Invariante für doppelt verkettete Listen aus der Vorlesung?

Nennen Sie einen Vorteil und einen Nachteil von Hashing mit verketteten Listen gegenüber offenem Hashing mit linearer Suche.

Nennen sie einen Vorteil und einen Nachteil von Mergesort gegenüber Quicksort.

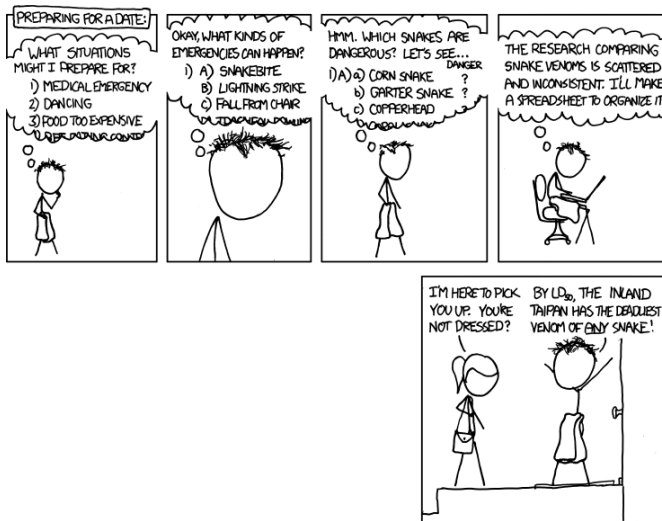
Wie lautet die Heap-Eigenschaft?

Nennen Sie einen Vorteil und einen Nachteil von Adjazenzfeldern gegenüber Adjazenzlisten.

Gegeben sei ein stark zusammenhängender Graph in folgender Darstellung:

- Knotenarray: Für jeden Knoten ID und Zeiger auf Array mit Kanten
- Kantenarray: Für jede Kante ID der Kante und ID des Zielknotens

Entwerf eine abgewandelte BFS, die *in-place* arbeitet, also nur $\mathcal{O}(1)$ zusätzlichen Platz benötigt. Die Laufzeit darf schlechter sein als eine herkömmliche Breitensuche.



I REALLY NEED TO STOP
USING DEPTH-FIRST SEARCHES.