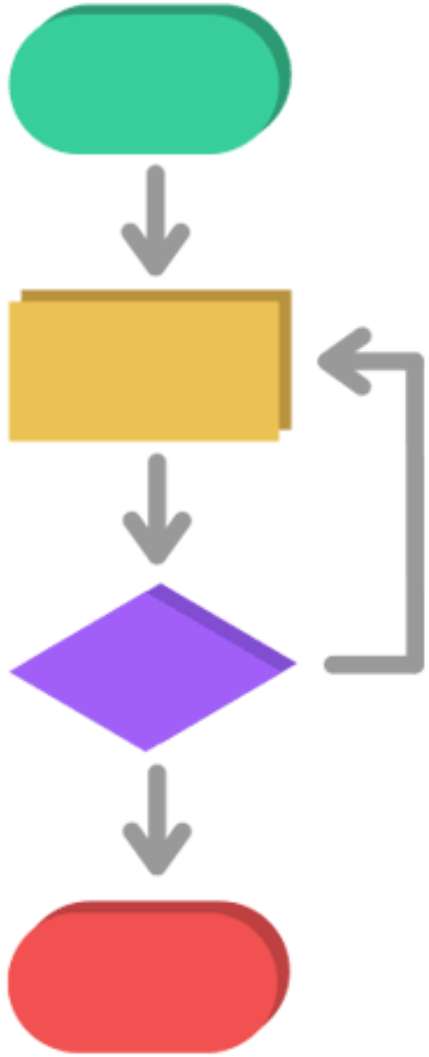
Algorithmen Tutorium 6

Beginn: 16:15



24.11.2021

Organisation

Graphen - Suchen

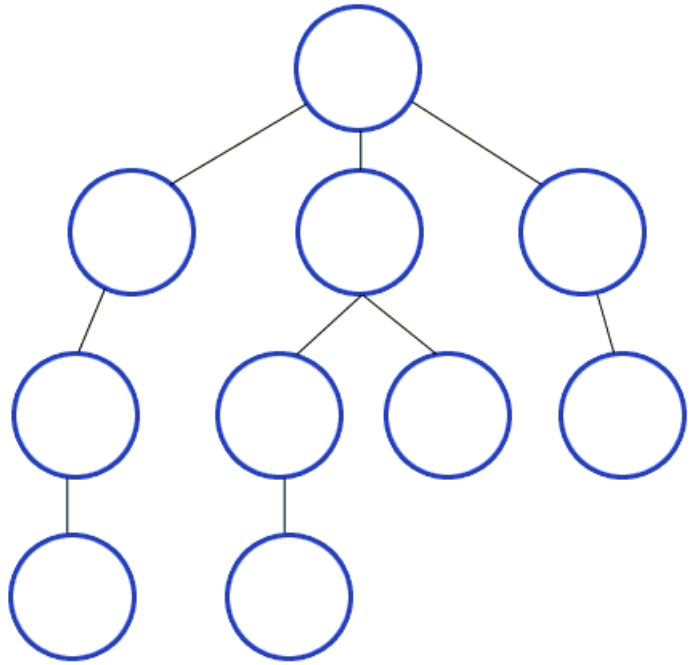
- Suche nach Element(en)
- Finde erreichbare Elemente

- Suche nach dem *kürzesten Pfad(en)*

- Für die Beispiele: Bäume sind spezielle Graphen!

Tiefensuche (DFS)

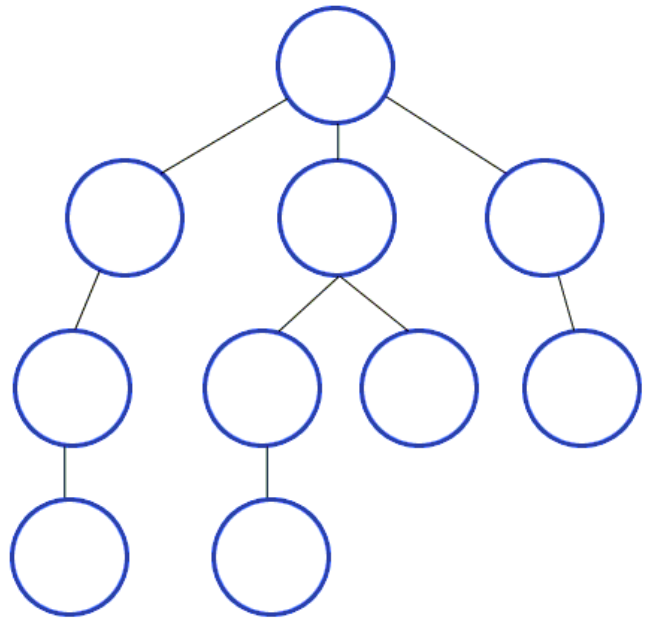
- Beschreite einen Pfad komplett
- Implementation: Stack (bzw. Rekursion)



Quelle: https://de.wikipedia.org/wiki/Tiefensuche#/media/Datei:Depth-First-Search.gif
Attribution: Mre, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons

Breitensuche (BFS)

- Besuche Knoten nach Entdeckungsreihenfolge
  - Alle Pfade mit Länge 1, dann 2, …
- Implementation: Queue



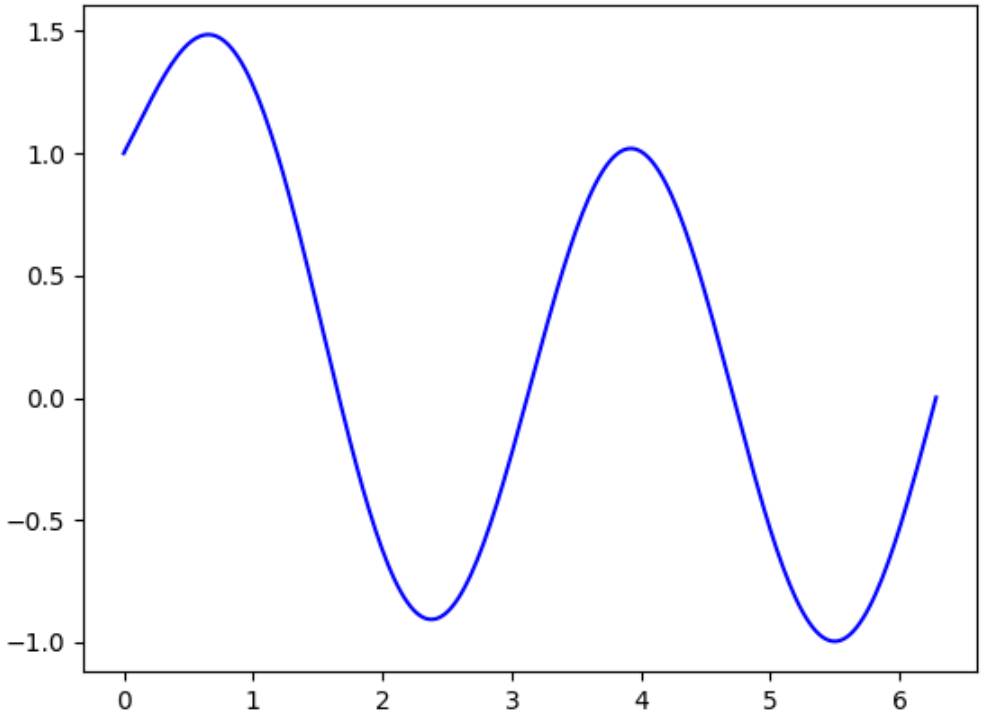
Quelle: https://de.wikipedia.org/wiki/Breitensuche#/media/Datei:Breadth-First-Search-Algorithm.gif
Attribution: Mre, CC BY-SA 3.0 <https://creativecommons.org/licenses/by-sa/3.0>, via Wikimedia Commons

Greedy-Algorithmusklasse

- Versucht das Ergebnis in jedem Schritt zu maxi-/minimieren
- Schwäche: Lokale Maxi-/Minima
  - Greedy-Ansatz nicht immer möglich
- Bekanntestes Beispiel: Dijkstra

Greedy Maximum finden

- Gehe entlang der Steigung bei x = 2 ☺
- Gehe entlang der Steigung bei x = 3 ☹



Dijkstra-Algorithmus

- Suche kürzesten(billigsten) Pfad von Startknoten s zu einem Zielknoten *z*
- Kann alle kürzesten Pfade von s berechnen
  - Komplett durchlaufen lassen: Nicht beenden, wenn *z* erreicht ist
- Wichtige Einschränkung:
  - Keine Negativen Kantengewichte! (Weil Greedy)
- Prinzip:
  - Erkunde solang den kürzesten entdeckten Pfad, bis z gefunden wurde

Dijkstra-Algorithmus

- Initialisiere Liste/PriorityQueue mit allen Knoten
  - Pfadkosten für s = 0
  - Pfadkosten für den Rest = ∞
  - Vorgängerknoten = NULL
- Beginne mit dem Startknoten s
- Solange z nicht erreicht
  - `Expandiere` den aktuellen Knoten
  - Besuche den nächsten unbesuchten Knoten mit geringster Priorität
- Return `Pfadkosten` oder `Backtracking`

Dijkstra (Expandieren)

- Für alle ausgehenden Pfade vom aktuellen Knoten `v`:
  - Überspringe Pfade zu bereits besuchten Knoten
  - `neuePriorität` = Kosten bis zum Knoten + Pfadkosten
  - Falls `neuePriorität` geringer als bisherige Priorität:
    - Update Vorgänger mit `v` und Priorität mit `neuePriorität`

Dikstra (Backtracking)

- Wie kann der Pfad rekonstruiert werden
  - Wenn wir z erreichen, wissen wir nur die Kosten, aber nicht den Pfad
- Speichere den Vorgängerknoten von dem ein Knoten erreicht wurde
- Erstelle eine Liste aus den Vorgängern
  - Beginne bei *z*
  - Vorgänger von *z*
  - Vorgänger vom Vorgänger von *z*
  - …
  - Endend bei s (oder keinem Vorgänger)

A*-Algorithmus

- Verbesserung von Dijkstra
- Dijkstra-Nachteil: Wir suchen in alle Richtungen
  - Evtl. wissen die Richtung von *z*
- Restkostenabschätzung (= Metrik)
  - Metrik darf Kosten niemals unterschätzen!!!
  - Z.B. Weltkarte -> Luftlinie
- Alternative Priorität:
  - Bisherige Kosten + geschätzte Kosten (Heuristik)

IPE-Software

- Zip-Archiv auf Festplatte entpacken
- (Erstelle Verknüpfung zu `ipe.exe` im bin-Ordner)
- Verknüpfung oder `ipe.exe` öffnen
- PDF öffnen und bearbeiten

Misc

- Ihr dürft euch in den Übungen auf PÜs beziehen;)
- Viele Interessante Videos zu DFS/BFS/Shortest Path
  - https://www.youtube.com/watch?v=yqZE5O8VPAU&ab_channel=GDC
  - https://www.youtube.com/watch?v=GC-nBgi9r0U&ab_channel=JohnSong
- Interessante Anwendungen:
  - Irrgärten Generierung/Lösung
  - Status eines Graphen ermitteln (Beispiel)

Welches Zahnrad dreht sich wie?