Dynamische und verteilte Abwandlungen des Floyd-Warshall-Algorithmus

Vortragender: Christopher Lege

Was wird Sie heute Erwarten ?

Was wird Sie heute Erwarten ?

<u>Teil Eins :</u>

- Vorstellung des Grundproblems
- Floyd-Warshall Algorithmus zur Lösung
- Analyse

Was wird Sie heute Erwarten ?

Teil Eins :

- Vorstellung des Grundproblems
- Floyd-Warshall Algorithmus zur Lösung
- Analyse

Teil Zwei :

- Innovative Verfahren zur Dynamisierung

Teil 1: Floyd-Warshall

1.1 Das Grundproblem

Bestimmung der kürzesten Wege in einem Graphen.

(1) Kürzeste Wege zwischen zwei Ecken

(2) Kürzeste Wege zwischen einer Ecke und allen anderen Ecken

(3) Kürzeste Wege zwischen allen Paaren von Ecken

Quelle: V. Turau, Algorithmische Graphentheorie S. 243

Teil 1: Floyd-Warshall
1.1 Das Grundproblem

Floyd-Warshall Algorithmus löst das Problem (3)

Teil 1: Floyd-Warshall

1.2 Der Algorithmus

<u>Warshall</u> : Algorithmus um den transitiven Abschluss eines Graphen zu berechnen

<u>Floyd</u> : Erweiterte den Algorithmus um die kürzesten Wege zwischen allen Paaren von Ecken zu berechnen

Teil 1: Floyd-Warshall

1.2 Der Algorithmus

- Robert W. Floyd 1936 – 2001
  „Assigning Meanings to Programs“ → Hoare Logik

- Stephen Warshall 1935 – 2006
  Bewies die korrektheit des Algorithmus zum berechnen des transitiven Abschlusses

Teil 1: Floyd-Warshall

1.2 Der Algorithmus

Beispiel Anwendung :

Vorabberechnung von kürzesten Wegen in hierarchischen Straßensystemen

Dient zum Reduzieren von Daten für die Wegberechnung.

Europa: 13 Level
Amerika: 6 Level

Teil 1: Floyd-Warshall
1.2 Der Algorithmus

Dynamisches Programmieren :

- Lösen von Optimierungsproblemen
- Aufteilen komplexer Probleme in kleinere Subprobleme (rekursive Beschreibung)
- Optimale Lösung ergibt sich aus den optimalen Lösungen der Subprobleme

Teil 1: Floyd-Warshall
1.2 Der Algorithmus

Dynamisches Programmieren :

- Top-Down-Lösungsweg :

➢ Subprobleme lösen und diese Lösung abspeichern.
➢ Gleiche Subprobleme in einer Tabelle speichern, bei Bedarf abfragen
➢ Optimale Lösung ergibt sich aus allen optimalen Lösungen der Subprobleme.

- Bottom-Up-Lösungsweg :

➢ Lösen der Subprobleme
➢ Lösungen der Subprobleme werden eingesetzt um ein nächst größeres Subproblem zu lösen.

Teil 1: Floyd-Warshall
1.2 Der Algorithmus

Was hat dieses Prinzip mit dem Floyd-Warshall Algorithmus zu tun ?

Der Algorithmus setzt dieses Prinzip als Vorbild für alle solche Probleme um !

Teil 1: Floyd-Warshall
1.2 Der Algorithmus

Zuvor wird noch benötigt:

- Graph $G = (V,E)$. $V \rightarrow$ Ecken, $E \rightarrow$ Kanten
- Graph G hat keine negativen Kreise
- Eine Distanzmatrix D_j
- Eine Vorgängermatrix A_j

Teil 1: Floyd-Warshall
1.2 Der Algorithmus

Kürzester Weg W von $i \in V$ nach $k \in V$
nur mit Ecken aus $\{1,\ldots,j\}$ $j \in V$.

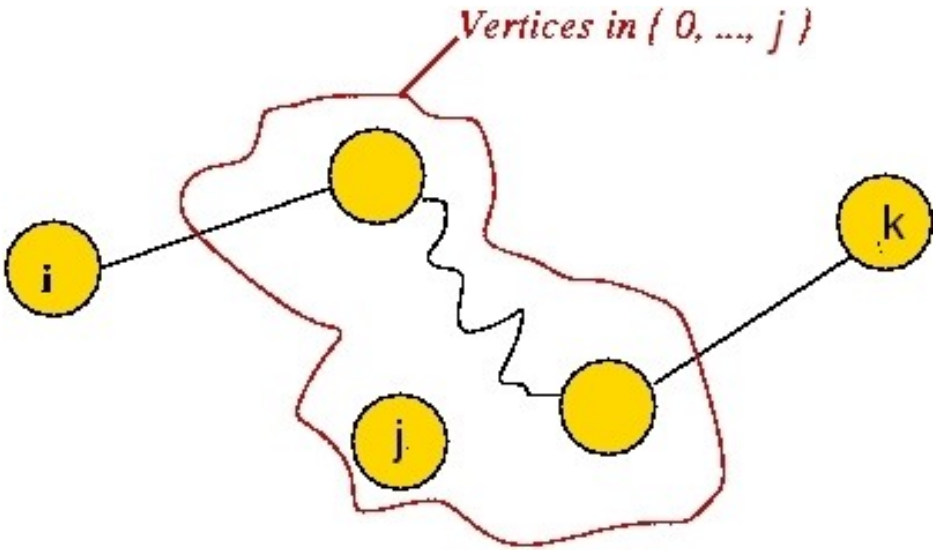
<u>2 Fälle sind zu unterscheiden</u>:

(1) Der Weg W verwendet nicht j.
(2) Der Weg W verwendet j.

Teil 1: Floyd-Warshall
1.2 Der Algorithmus

- Fall (1):

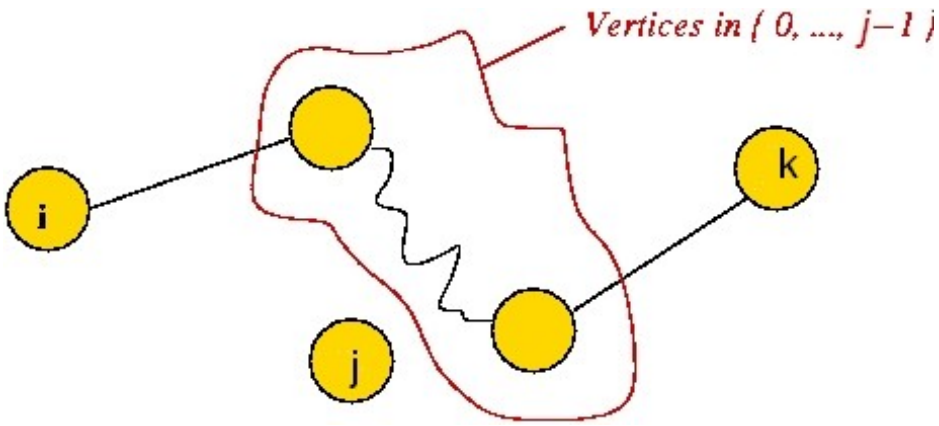
Da der Weg nicht j benutzt, muss der Weg aus Ecken aus der Menge $\{ 1,...,j-1 \}$ bestehen.



Teil 1: Floyd-Warshall
1.2 Der Algorithmus

- Fall (1):

Dieses *j* wird bei der weiteren suche des kürzesten Weges nicht mehr beachtet.



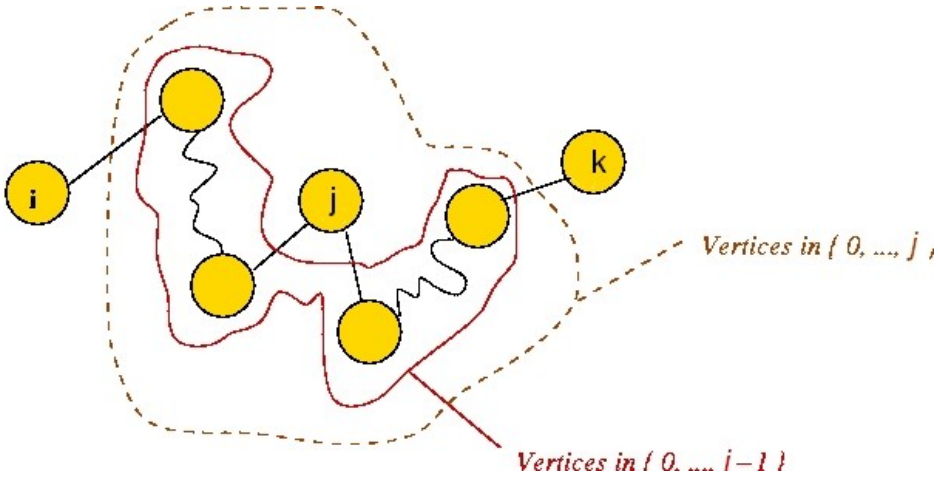
Teil 1: Floyd-Warshall
1.2 Der Algorithmus

- Fall (2):

  Da j nur einmal auf dem Weg vorkommt. Kann der Weg jetzt in zwei Teilwege aufgeteilt werden

- Beide Teilwege können wieder einzeln optimiert werden

- Nur noch Ecken aus $\{ 1,\ldots,j\text{-}1 \}$



Teil 1: Floyd-Warshall
1.2 Der Algorithmus

Die Bedingung die überprüft werden soll sieht folgendermaßen aus:

$$D[i,k] = min(D[i,k], D[i,j] + D[j,k])$$

Teil 1: Floyd-Warshall
1.2 Der Algorithmus

Der Algorithmus als Pseudo-Pascal-Code:

```
for j := 1 to n do
    for i := 1 to n do
        for k := 1 to n do
            if D[i,k]  > D[i,j] + D[j,k] then
                D[i,k] := D[i,j] + D[j,k]
```

Teil 1: Floyd-Warshall
1.2 Der Algorithmus

Mit einer zusätzlichen Zuweisung kann auch gleich der Weg in die Vorgängermatrix V eingetragen werden:

```
for j := 1 to n do
    for i := 1 to n do
        for k := 1 to n do
            if D[i,k]  > D[i,j] + D[j,k] then
                D[i,k] := D[i,j] + D[j,k]
                V[i,k] := V[j,k]
```

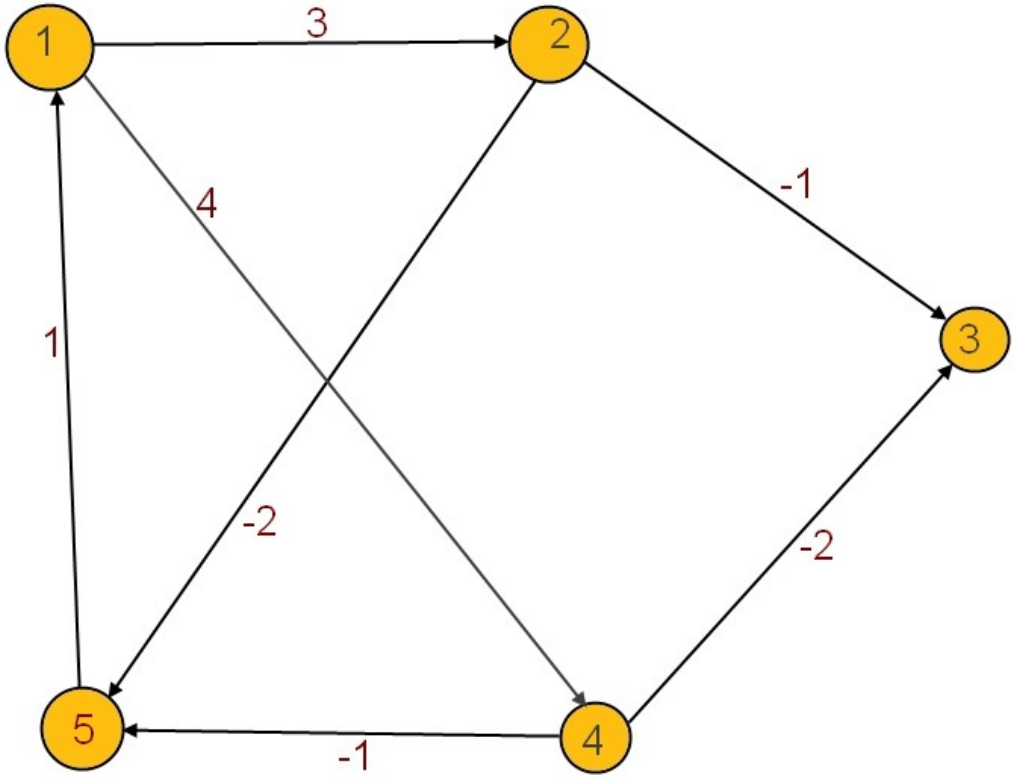
Teil 1: Floyd-Warshall
1.3 Das Beispiel

- Distanzmatrix

$$D_0 = \begin{pmatrix} 0 & 3 & \infty & 4 & \infty \\ \infty & 0 & -1 & \infty & -2 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -2 & 0 & -1 \\ 1 & \infty & \infty & \infty & 0 \end{pmatrix}$$

- Vorgängermatrix

$$V_0 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 4 \\ 5 & 0 & 0 & 0 & 0 \end{pmatrix}$$



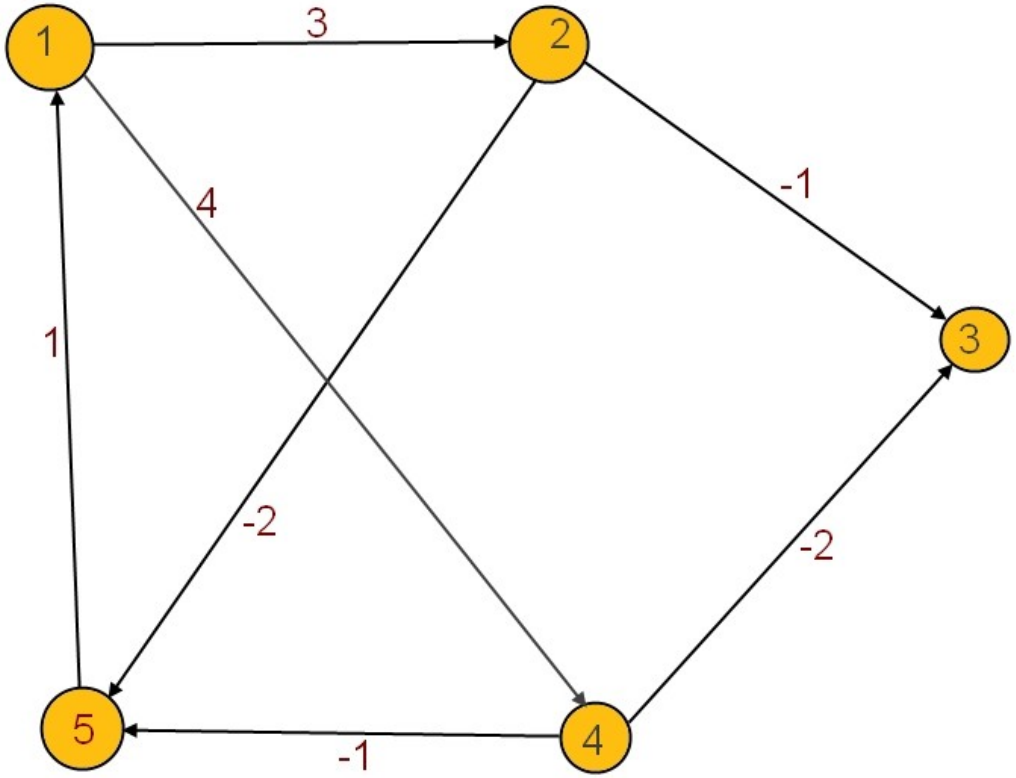
Teil 1: Floyd-Warshall
1.3 Das Beispiel

- Distanzmatrix

$$D_1 = \begin{pmatrix} ٠ & 3 & \infty & 4 & \infty \\ \infty & ٠ & -١ & \infty & -2 \\ \infty & \infty & ٠ & \infty & \infty \\ \infty & \infty & -٢ & 0 & -1 \\ ١ & \mathbf{4} & \infty & \mathbf{5} & 0 \end{pmatrix}$$

- Vorgängermatrix

$$V_1 = \begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 4 \\ 5 & \mathbf{1} & 0 & \mathbf{1} & 0 \end{pmatrix}$$



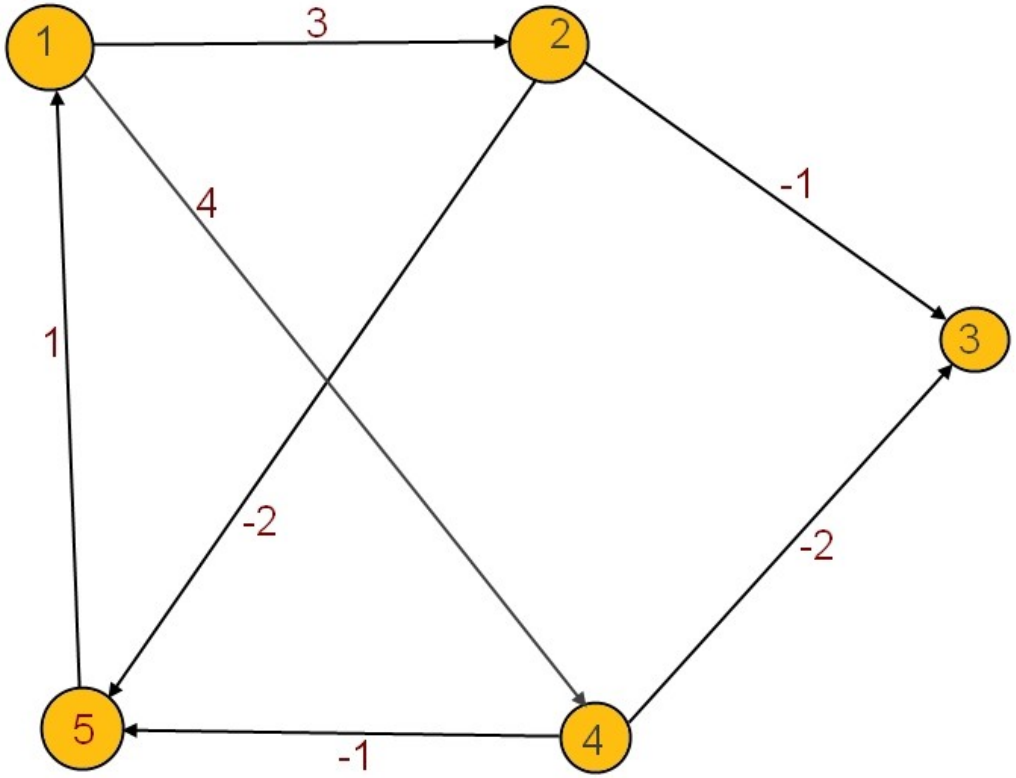
Teil 1: Floyd-Warshall
1.3 Das Beispiel

- Distanzmatrix

$$D_2 = \begin{pmatrix} 0 & 3 & \mathbf{2} & 4 & \mathbf{1} \\ \infty & 0 & -1 & \infty & -2 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -2 & 0 & -1 \\ 1 & 4 & \mathbf{3} & 5 & 0 \end{pmatrix}$$

- Vorgängermatrix

$$V_2 = \begin{pmatrix} 0 & 1 & \mathbf{2} & 1 & \mathbf{2} \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 4 \\ 5 & 1 & \mathbf{2} & 1 & 0 \end{pmatrix}$$



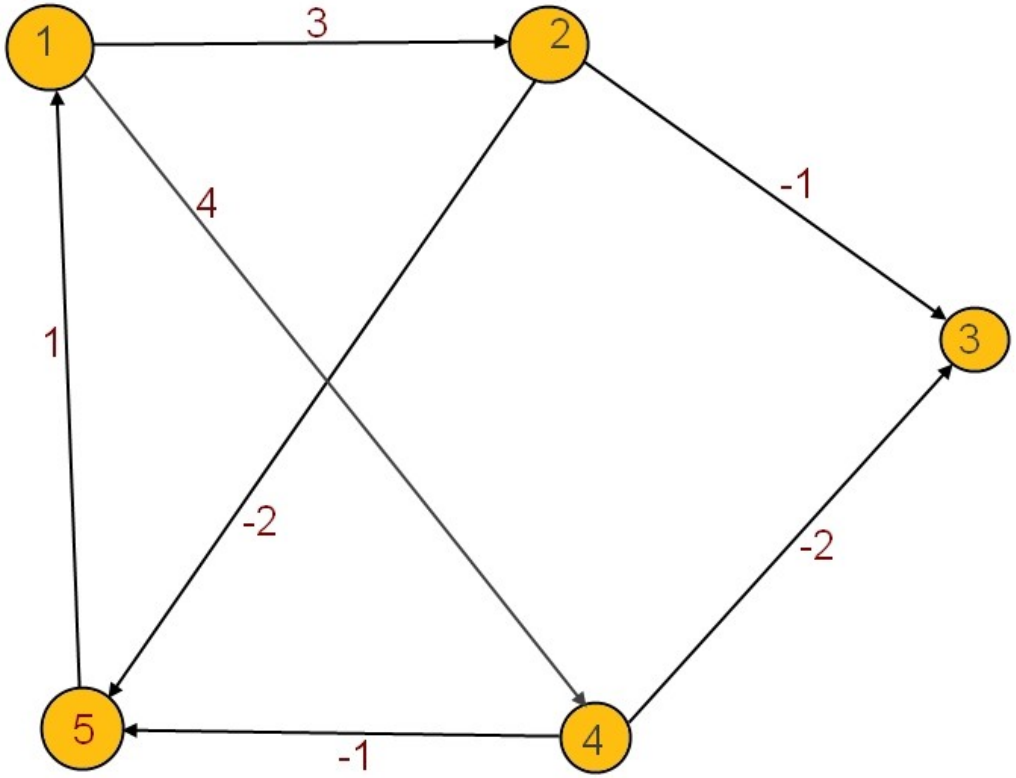
Teil 1: Floyd-Warshall
1.3 Das Beispiel

- Distanzmatrix

$$D_3 = \begin{pmatrix} 0 & 3 & 2 & 4 & 1 \\ \infty & 0 & -1 & \infty & -2 \\ \infty & \infty & 0 & \infty & \infty \\ \infty & \infty & -2 & 0 & -1 \\ 1 & 4 & 3 & 5 & 0 \end{pmatrix}$$

- Vorgängermatrix

$$V_3 = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 4 \\ 5 & 1 & 2 & 1 & 0 \end{pmatrix}$$



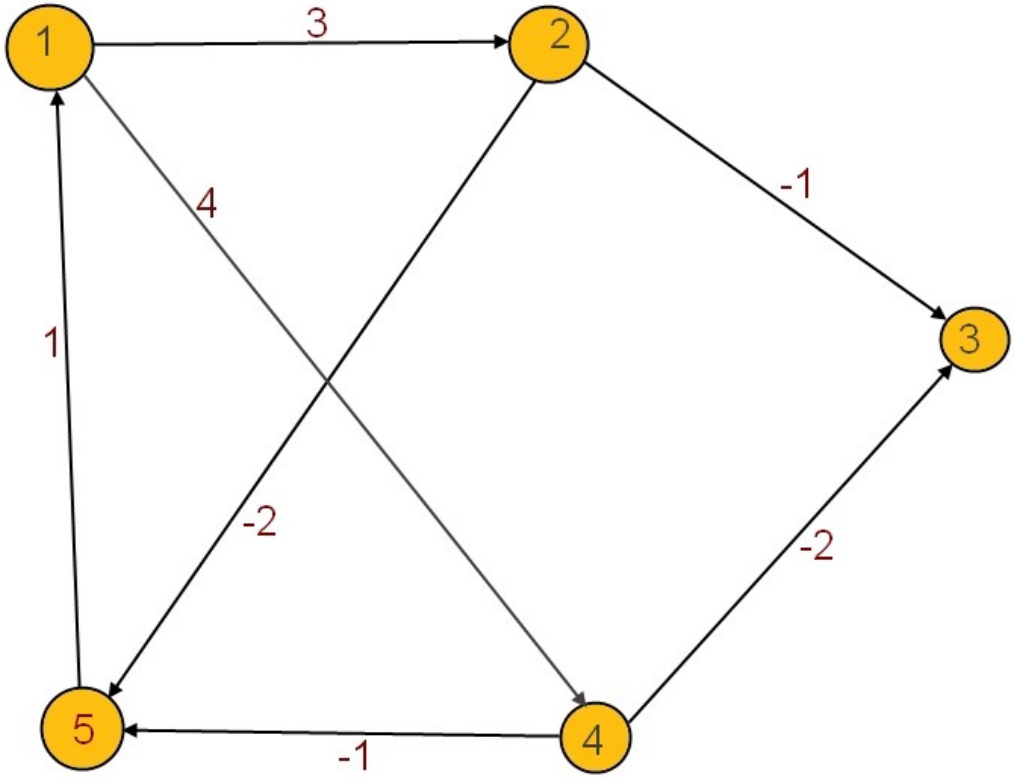
Teil 1: Floyd-Warshall
1.3 Das Beispiel

- Distanzmatrix

$$D_4 = \begin{pmatrix} 0 & 3 & ٢ & 4 & ١ \\ \infty & ٠ & -١ & \infty & -2 \\ \infty & \infty & ٠ & \infty & \infty \\ \infty & \infty & -٢ & 0 & -1 \\ ١ & 4 & ٣ & 5 & ٠ \end{pmatrix}$$

- Vorgängermatrix

$$V_4 = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 \\ 0 & 0 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 4 & 0 & 4 \\ 5 & 1 & 2 & 1 & 0 \end{pmatrix}$$



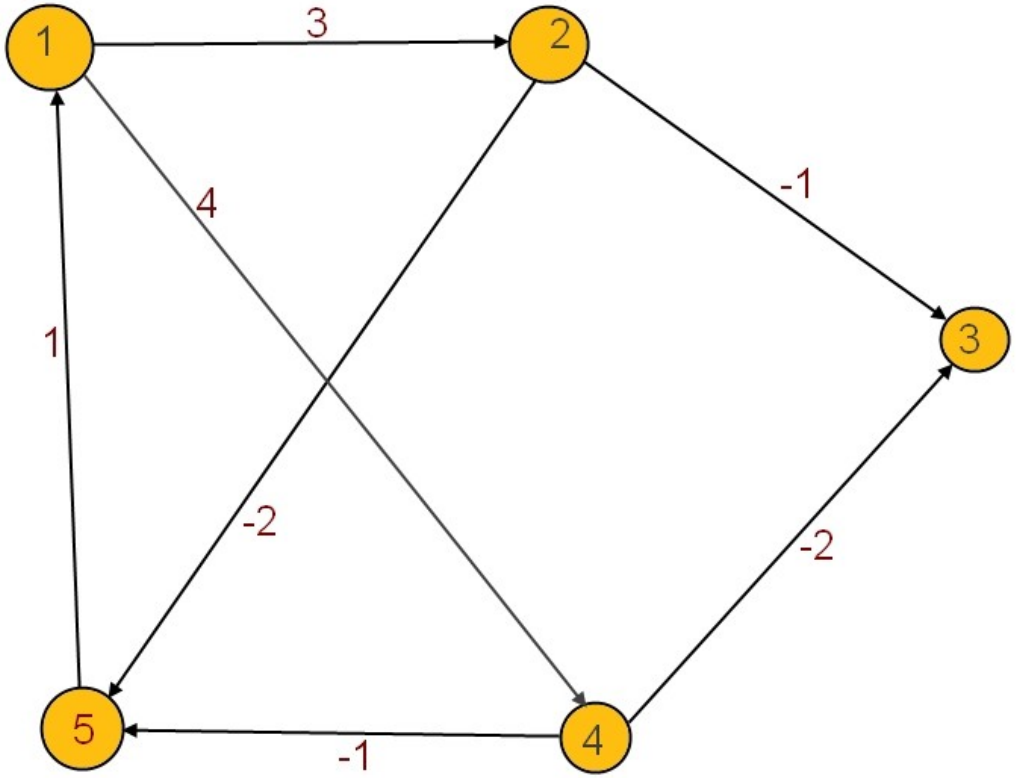
Teil 1: Floyd-Warshall
1.3 Das Beispiel

- Distanzmatrix

$$D_5 = \begin{pmatrix} 0 & 3 & 2 & 4 & 1 \\ \mathbf{-1} & 0 & -1 & \mathbf{3} & -2 \\ \infty & \infty & 0 & \infty & \infty \\ \mathbf{0} & \mathbf{3} & -2 & 0 & -1 \\ 1 & 4 & 3 & 5 & 0 \end{pmatrix}$$

- Vorgängermatrix

$$V_5 = \begin{pmatrix} 0 & 1 & 2 & 1 & 2 \\ \mathbf{5} & 0 & 2 & \mathbf{1} & 2 \\ 0 & 0 & 0 & 0 & 0 \\ \mathbf{5} & \mathbf{1} & 4 & 0 & 4 \\ 5 & 1 & 2 & 1 & 0 \end{pmatrix}$$



Teil 1: Floyd-Warshall
1.4 Die Analyse

- Laufzeit: $O(n^3)$
- Speicher: $O(2n^2)$
- statischer Algorithmus
- Einfach zu Implementieren
- Einfach zu Parallelisieren

Teil 2: Innovative Verfahren
2.0 Dynamisierungen

- Vorstellung von 2 verschiedenen Ansätzen
- Jeffrey Miller
- Demetrescu & Italiano

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Jeffrey Miller, Assistent Professor University Anchorage Alaska
- Ph.D in Computer Science, Mai 2007

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Unterscheidet 3 Klassen von Algorithmen:

  Naive Algorithmen

  Dynamische Algorithmen

  Pre-Computed-Algorithmen

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Naive Algorithmen
    - Floyd, Johnson
    - Zu langsam um bei einem Kantenupdate erneut alle Wege zu berechnen
    - Für eine echte dynamische Anwendung nicht anwendbar

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Dynamische Algorithmen
    - Demetrescu & Italiano
    - Verbesserung der Naiven Algorithmen
    - Später mehr...

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Pre-Computed Algorithmen
  - Wieder 3 Klassen
  - Hybrid-Class, Constant-Update-Class, Constant-Query-Class
  - Berechnen A L L E Wege im Graphen einmal Vorweg und speichern diese.

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Pre-Computed-Constant-Update
  - Wenn eine Kante geupdatet wird, soll dies in Konstanter Zeit passieren: $O(1)$
  - Opfert dafür aber die Zeit den kürzesten Pfad zu finden: $O(Anzahl\ Pfade * n)$

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Pre-Computed-Constant-Query
  - Wenn eine Kante geupdatet wird, soll dies in $O(n^2 * Anzahl\ Pfade \log Anzahl\ Pfade)$ passieren
  - Dafür wird der kürzeste Weg in Konstanter Zeit gefunden: $O(1)$

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Pre-Computed-Hybrid
  - Schließt einen Kompromiss zwischen den ersten beiden Klassen
  - Kantenupdates in: $O(n^2 * Anzahl\ Pfade)$
  - Kürzesten Weg gefunden in: $O(Anzahl\ Pfade)$

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Vergleich der Algorithmen anhand einer Anwendung. (97 Knoten, 101 Kanten , 10.000 Updates, 10.000 Anfragen, in ms)

Algorithmen	Pre-computation	Kantenupdate	Schnellsten Weg finden
Naive	0	13229	0
Italiano & Demetrescu	0	3358	18
Constant-Update	31545	0	0
Constant-Query	31545	2091	0
Hybrid	31545	9	0

Teil 2: Innovative Verfahren
2.1 Ansatz von Jeffrey Miller

- Fazit und Bewertung:
  - Beispielrechnungen nur für die Highways in LA Downtown
  - Enorm Speicherintensiv, da alle Pfade gespeichert werden müssen.
  - In Zusammenhang mit Highway Hierachies möglicherweise nutzbar
  - Computerfarm für größere Straßennetze notwendig
  - Vorberechnungszeit: $O(n^2 * m!)$

Teil 2: Innovative Verfahren
2.2 Ansatz Demetrescu & Italiano

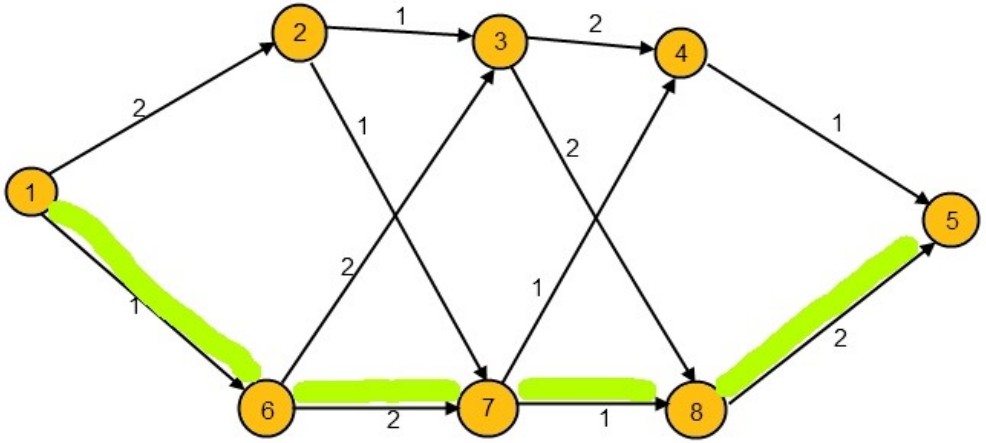
- Camil Demestrescu, geb. 1971:
  Italienischer Theoretischer Informatiker. Dissertation wurde ausgezeichnet. Lehrt an der Universität Rom „La Sapienza“
- Guiseppe F. Italiano, geb 1961:
  Demetrescus Doktorvater. Theoretischer Informatiker. Doktorarbeit: „Dynamic data structures for graphs“.
- Beide Implementieren ihre Überlegungen auch!

Teil 2: Innovative Verfahren
2.2 Ansatz Demetrescu & Italiano

- Die Grundlage für des Algorithmus von Demetrescu und Italiano ist die Definition vom „uniform path“
- Def: Ein Weg π wird „uniform path“ genannt, wenn jeder Sub-Weg von π ein kürzester Weg ist
- Der Weg π muss nicht zwangsläufig ein kürzester Weg sein

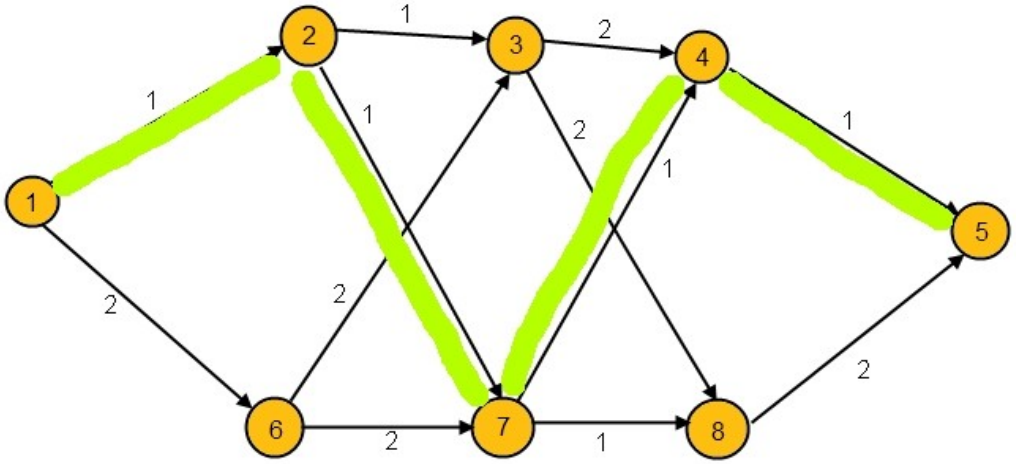
Teil 2: Innovative Verfahren
2.2 Ansatz Demetrescu & Italiano

- Ein „uniform path“ der nicht gleichzeitig ein kürzester Weg ist.



Teil 2: Innovative Verfahren
2.2 Ansatz Demetrescu & Italiano

- Ein „uniform path“ der gleichzeitig ein kürzester Weg ist.



Teil 2: Innovative Verfahren
2.2 Ansatz Demetrescu & Italiano

- Def: Historischer kürzester Weg:
  Ein Weg ist ein Historischer kürzester Weg, wenn er ein kürzester Weg während der Update-Sequenz war.
- Def: „potentially uniform path“:
  Ein Weg wird „potentially uniform“ genannt, wenn jeder Sub-Weg ein historischer kürzester Weg ist.

Teil 2: Innovative Verfahren
2.2 Ansatz Demetrescu & Italiano

- Die Hauptidee des Algorithmus ist das dynamische pflegen von „potentially uniform paths“
- Bei einem Kantenupdate wird dann folgendermaßen Verfahren:
  1. Es werden alle gepflegten Pfade gelöscht, die die aktualisierte Kanten enthalten
  2. Es wird parrallel von allen Knoten eine dynamische Version von Dijkstra gestartet
  3. Bei jedem Schritt wird ein kürzester Weg aus einer Warteschlange genommen und mit einem historischen kürzesten Weg verbunden um einen „potentially uniform path“ zu bilden.

Teil 2: Innovative Verfahren
2.2 Ansatz Demetrescu & Italiano

- Die Laufzeit im schlechtesten Fall: $n^2 * \log^{۳} n$
- Der Speicheraufwand: $m * n * \log n$

Teil 2: Innovative Verfahren
2.2 Ansatz Demetrescu & Italiano

- Sehr viel versprechender Ansatz, da die Anzahl der zu untersuchenden Kanten reduziert wird
- Demetrescu & Italiano haben auf Basis der „uniform paths“ einen statischen Algorithmus entwickelt mit einer Laufzeit von: $O\left(UP + n^2 \log n\right)$
- Benötigt nicht soviel Speicherplatz wie der Ansatz von Jeffrey Miller.
- Da immer nur eine bestimmter Ausschnitt des Graphen betrachtet wird ist dieser Algorithmus besonders für dichte Graphen geeignet.

Was sollten Sie heute mitgenommen haben ?

- Dynamisches Programmieren
- Floyd-Warshall
- Die Idee von Jeffrey Miller
- Die Ideen von Demetrescu & Italiano

Vielen Dank für Ihre Aufmerksamkeit !