Ergänzungen und Korrekturen zur Theoretischen Informatik I

Seid im Skript zu ergänzen

Inhaltsübersicht

11. Divide-and-conquer und Rekursionsgleichungen

12. Weitere Beispiele zum dynamischen Programmieren

13. Sortieren (eine untere Schranke)

Ergänzungen und Korrekturen.

Zu 2.1.14:

Letzte Zeile: Die Größe des arrays ist nicht unbedingt gleich der Kantenanzahl, aber sicherlich $\leq |E| + |V|$. Hier ist $A[1..6]$ falsch, wir haben im Beispiel $A[1..7]$. Die Größe des arrays ist damit

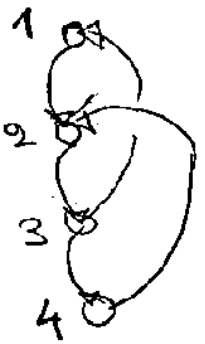
$|E|$ + # Knoten vom A-Grad 0.

Rückkante: $|E| \geq \frac{n}{2}$ stabb $|E| \geq n-1$. Bei $|E| \leq \frac{n}{2}$ haben wir isolierte Knoten, was i.a. nicht erwünscht ist.

Diese Seite mußte aus rechtlichen Gründen entfernt werden!

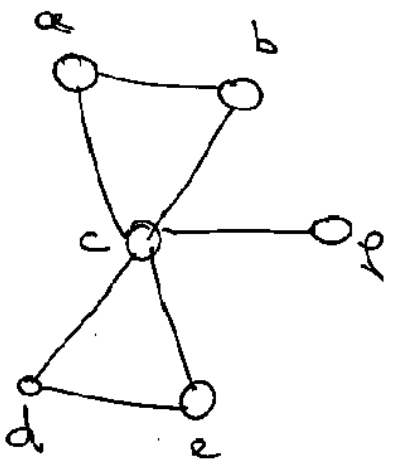
Zu S. 4.27: In dem Kommentar ist zweimal weiß durch grau zu ersetzen.

Zu §. 5.4 Man beachte auch noch folgendes Beispiel.

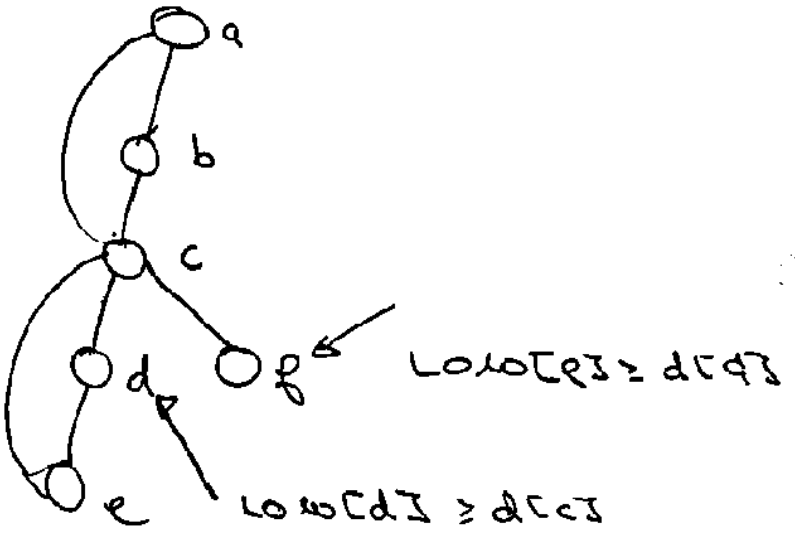


Ist stark zshg. Die Wege $1 \circ \to \circ 4$ und $4 \circ \to \circ 1$ sind nicht kantendisjunkt. Zu beiden kommt $2 \circ \to \circ 3$ vor.

WIE FINDET MAN DIE ZWEIFACHEN KOMPONENTEN



MÖGLICHER BAUM:

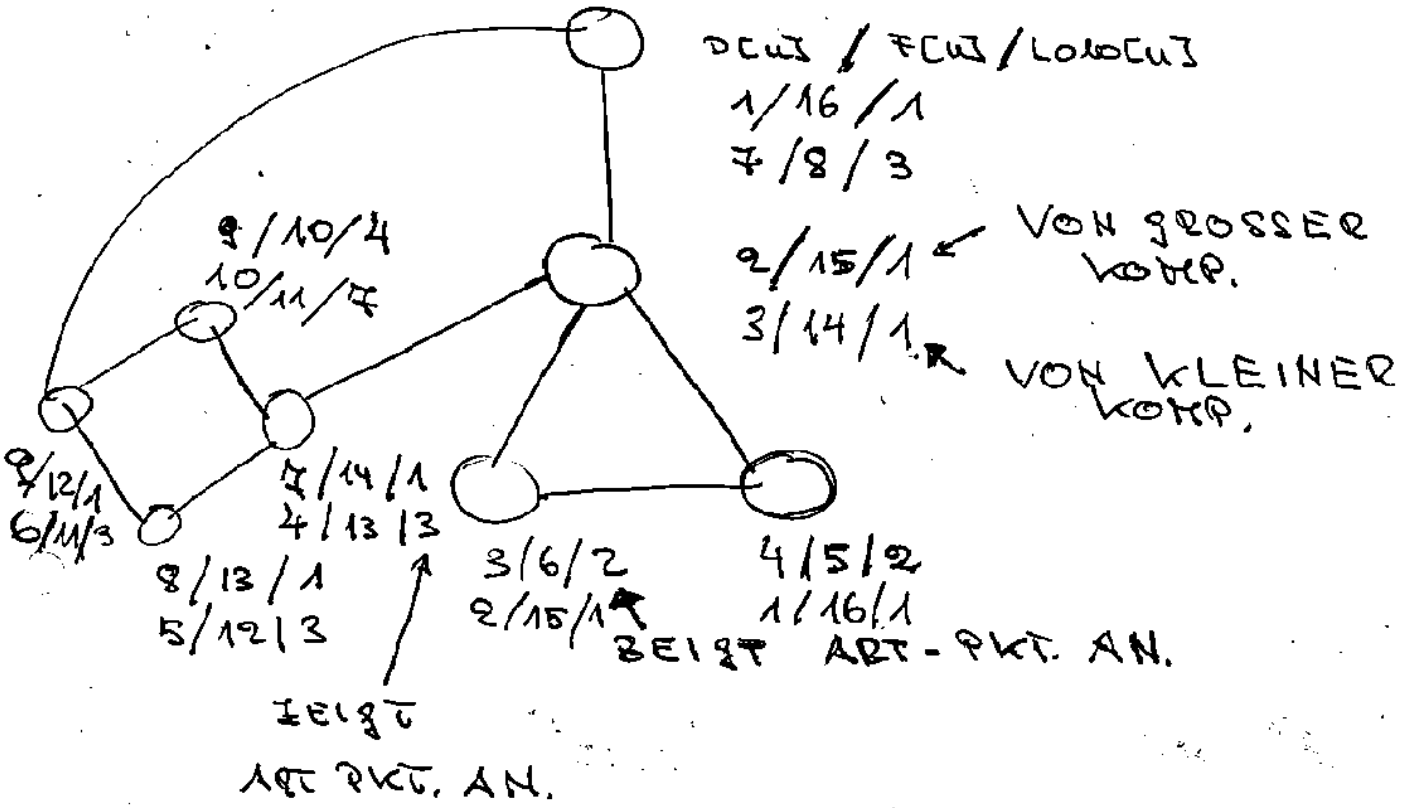


DFS-VISIT(g) FERTIG ⇒ 1. KOMPO.

DFS-VISIT(d) FERTIG ⇒ 2. KOMPO.

DFS-VISIT(a) FERTIG ⇒ 3. KOMP.

LOW-WERT UND DURCHLAUFREIHENFOLGE



Zu §. 8.21

Die Rechnung in der Mitte muß folgendermaßen lauten:

$$\frac{2^{O(n)}}{2^{\Omega(n \cdot \log n)}} = \frac{1}{2^{\Omega(n \cdot \log n) - O(n)}} \geq \frac{1}{2^{\varepsilon \cdot n \cdot \log n - cn}} \geq \frac{1}{2^{Q(n \log n)}}$$

$\geq \varepsilon \cdot n \cdot \log n$

$\leq c \cdot n$

reelle andere Konstante.

Also: # Verschiedene Aufrufe ist nur ein verschwindend kleiner Anteil an allen Aufrufen!

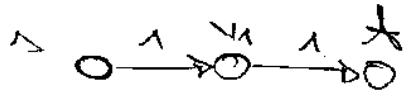
Das

$$\frac{2^{O(n)}}{2^{\Omega(n \cdot \log n)}} = \frac{1}{2^{\Omega(\log n)}}$$ ist falsch!

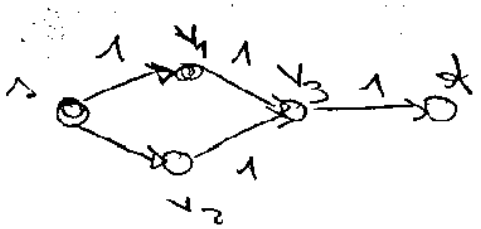
Seite 9.5 überarbeitet neu (9.5)

3. Erhöhe den Fluß entlang des Weges soweit es geht. Danach bei 2. weiter.

Dieses Problem ist von ganz anderem Charakter als bisher, da die Anzahl der zunächst zulässigen Flüsse unendlich ist. Ganz einfach in



Ist $\rho(s,v_1) = \rho(v_1,t) < 1$ so haben wir einen erlaubten Fluß. Aber sogar maximale Flüsse gibt es unendlich viele:



Jeder Fluß mit $\rho(s,v_1) + \rho(s,v_2) = 1$ ist maximal.

Zu S. 211

Erste Zeile: Es ist q_f mit der Restkapazität $\overline{u}_f$ ein Flußnetzwerk. Analog $\overline{u}_f(u,v)$ ist für alle Paare $(u,v) \in V \times V$ definiert:

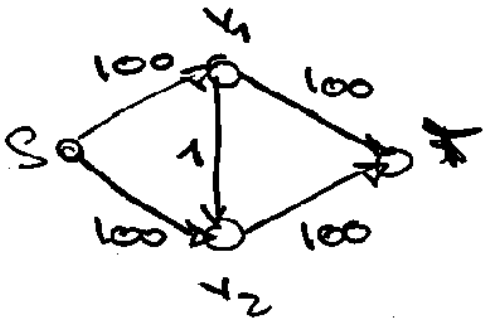
Zu S. 221

Termination zunächst nur bei ganzzahligen Kapazitäten. Erhöhung jedesmal um eine ganze Zahl ≥ 1. Also ist irgendwann ein maximaler Fluß erreicht!

Diese Seite mußte aus rechtlichen Gründen entfernt werden!

BEISPIEL LANGE LAUFZEIT

9.23



$f_{MAXIMAL} \Rightarrow |f| = 200.$

ERWEITERUNGSWEGE:

$(s, v_1, v_2, t) + 1$

$(s, v_2, v_1, t) + 1$

$(s, v_1, v_2, t) + 1$

$\vdots$

BESSER

$(s, v_1, t) + 100$

$(s, v_2, t) + 100.$

200 EINZELNE ERHÖHUNGEN

BEACHTE DAS RESTNETZWERK

„OSZILLIEREN" VON $v_1 \xrightarrow[1]{} v_2$.

Man sieht, daß Laufzeiten
der Größe $\Omega(|E| \cdot |P^*|)$ durchaus
auftreten können. Es lassen sich
die Kapazitäten so wählen, daß
$|P^*|$ exponentiell in der Länge
(in Bits) von $\mathcal{Z}$ und der Kapazitäts-
funktion ist. Abhilfe schafft
hier: Es wird im Restgraph
immer ein kürzester Erweiterungspfad,
d.h. minimaler Kantenanzahl
gewählt.

9.24a

Algorithmus (Edmonds Karp)

1. $f(u,v) = 0$
2. while t in V_f von s erreichbar
3. $B_f(x_f, s)$ // Breitensuche
4. $v_k := t$, $v_{k-1} := p[v_k]$, ...
   $v_1 := p[v_2]$, $v_0 := v_0 := p[v_1]$ // p Breitensuchbaum
5. $W := (s = v_0, v_1, \dots, v_k = t)$ // W. ist eine // k. W. bezogen
7. $g(v_i, v_{i+1}) := M_f(W)$ // auf die Kanten // Kanten
8. $f := f + g$; V_f berechnen

}

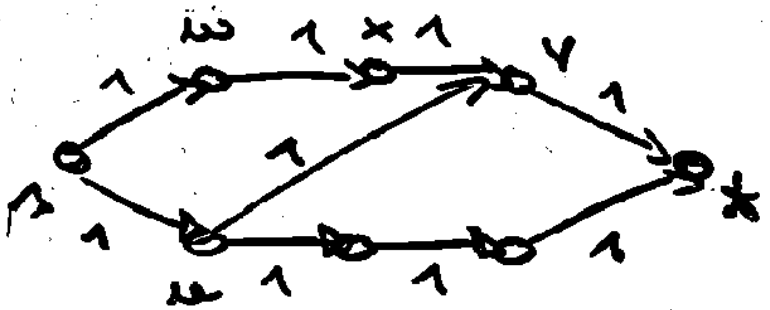
2. Ausgabe von f als maximalen Fluß.

Beachte nach Algorithmus:

Erweiterungsweg = Weg $s \to t$ mit minimaler Kantenzahl. Vgl. bsp. S. 9.23, dort gerade nicht minimale Kantenzahl.

Zu EDMONDS KARP

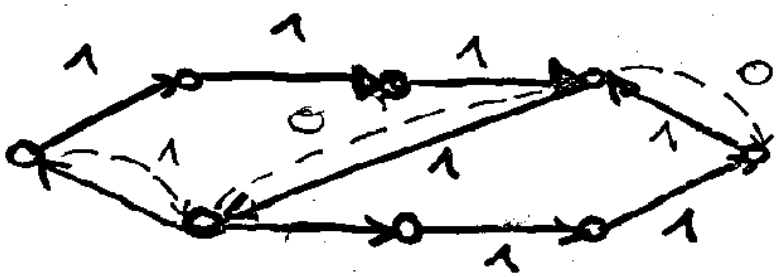
- NEGATIVER FLUSS
  (⇔ FLUSS UMLENKEN)



FLUSSNETZ-
WERK.

KÜRZESTER E-WEG

$(s, u, v, t) + 1$



RESTNETZ-
WERK

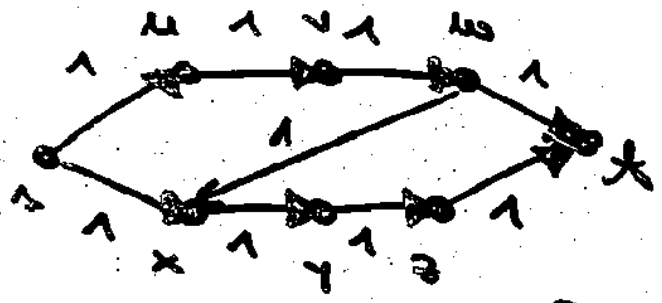
E-WEG

$(s, w, x, v, u, \ldots t) + 1.$

MAX. FLUSS GEFUNDEN.

○ VERKÜRZUNG KÜRZESTER WEGE FALLS NICHT (!) E.K.

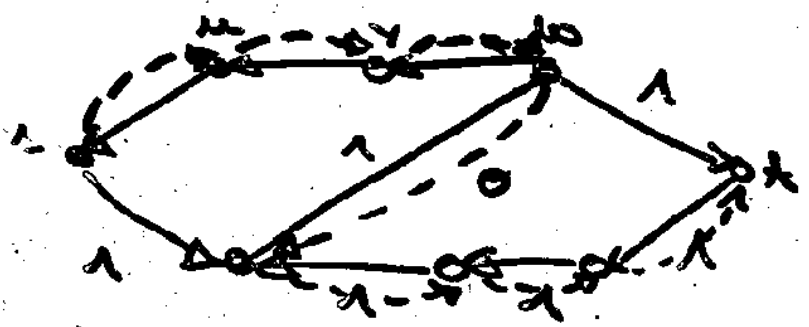
↗ EDMONDS KARP



= 9

DIST (s, w) = 3
DIST (s, t) = 4
E-WEG (s, u, v, w, x, .. t) + 1

↳ NICHT E.K.



DIST (s, w) = 2
DIST (s, t) = 3

E-WEG (s, x, w, t) + 1.

Wir zeigen im restlichen Verlauf dieses Abschnitts folgendes:

Satz

Sind $w_1, w_2, w_3, \ldots, w_n$ die von Edmonds Kap betrachteten ξ-Wege, dann gilt { # Knoten des betrachteten Flußnetzes }

(a) $L_g(w_1) \leq L_g(w_2) \leq \ldots \leq L_g(w_n) \leq |V|$

(b) $L_g(w_1) \neq L_g(w_{|E|+1}) \neq L_g(w_{2|E|+1})$

$\neq L_g(w_{3|E|+1}) \neq \ldots \neq$

Aus diesem Satz folgt { Heißt: Maximal $|E|$ ξ-Wege von gleicher Länge. }

Folgerung

Bei Edmonds Kap gilt:
(a) $|E| \cdot |V|$ Runden der Schleife { in OM zu bearbeiten sind. }
(b) $O(|E|^2 \cdot |V|)$ Zeit, sofern Kapazitäten

Beweis (der Folgerung)

(a) Mit dem letzten Satz ist für

$$w_1, w_2, \ldots, w_{|E|+1}, \ldots, w_{2|E|+1}, \ldots$$

Länge ≥ 1 $\quad \geq 2$ $\quad \geq 3$

also sicher ist beim Index $(|V|-1)|E|+1$ der Länge $|V|$ erreicht. Wir haben maximal $|E|$ Wege der Länge $|V|$ und bei $|E| \cdot |V|$ ist ganz sicher der letzte Weg erreicht und die Schleife hat maximal $|E| \cdot |V|$ Durchläufe.

(b) Ein Durchlauf durch die Schleife braucht eine Zeit von $O(|V|+|E|)$. Und für $|E| \geq |V|$ gilt die Behauptung. □

(2.29)

Betrachten uns einmal einen Lauf der Schleife, zunächst vom allgemeinen Fall Fulkerson.

Sei dazu

f = Fluß vor dem Lauf der Schleife

f' = Fluß nach dem Lauf der Schleife

und g_f, $g_{f'}$ sind die zugehörigen

Restnetzwerke. Es gibt, f_f entsteht aus f_f indem eine (oder mehrere) Kanten $u \to v$ gelöscht werden und durch $v \to u$ ersetzt werden. Dabei gilt für $u \to v$:

- Die Kanten $u \to v$ liegen auf dem gewählten E-Weg.
- Es ist möglich, daß eine Umkehrung $v \to u$ bereits in f_f vorhanden ist. (Macht nichts)

Bei Edmonds Karp gilt zusätzlich:

- Die Kanten $u \to v$ wie oben liegen auf einem kürzesten (!) Weg $s \to t$.

Wir betrachten einmal alles ganz allgemein.

Lemma

Sei G ein gerichteter Graph mit Knoten s und t. #Kanten eines k.W.

sei $k = \mathrm{Dist}(s,t)$. Ist

$u \to v$ Kante auf irgendeinem k.W. $s \to t$, so ist auch die Umkehrkante $v \to u$ in G enthalten.

(a) Ist $u \to v$ Kante eines k.W. $s \to t$, so kommt nie $v \to u$ auf einem k.W. $s \to t$ vor.

(4.39)

(b) Enthält ein einfacher Weg W

$\overset{v_0}{\circ} \rightarrow \overset{v_k}{\circ}$ l Umkehrkanten, dann

$$L_2(W) \geq k + 2l.$$

Beweis

(a) Angenommen auf den kürzesten Wegen kommen irgendwelche dieser Umkehrkanten vor. Wir betrachten einen k.W. auf dem eine dieser Umkehrkanten am weitesten vorne steht. Sei

$$W = (v = v_0, v_1, \dots, v_k = t)$$

und $\overset{v_i}{\circ} \leftarrow \overset{v_{i+1}}{\circ}$ sei die erste Umkehrkante und kein k.W. benützt auf seinem

(9.1/39)

ersten i Schritten eine Kante.

Also gibt es einen k-W.

$$W = (s = u_0, \dots, u_j, u_{j+1}, \dots, u_k = t)$$

mit $u_{j+1} = v_i$ und $u_j = v_{i+1}$. Dann

ist $j \geq i$, da sonst Kante

auf U auf den ersten i Schritten.

Dann ist aber

$$W' = (\underbrace{s = v_0, v_1, \dots, v_i}_{\text{von } W} = \underbrace{u_{j+1}, \dots, u_k = t}_{\text{von } U})$$

ein Weg mit

$$Lg(W') = i + k - (j+1) \overset{j \geq i}{\leq} k - 1,$$

da $j \geq i$. Das kann aber nicht

sein, da $Dist(s,t) = k$!

(b) Einige Spezialfälle zuvächst.

$l = 0$, v da $\mathrm{Dist}(1, k) = k$.

$l = 1$, dann betrachten wir

$$W = (1 = u_0, u_1, \dots, \underbrace{u_i, u_{i+1}}_{\text{1 Schritt}}, \dots u_n = k)$$

$W_1 =$ bis u_i, $W_2 =$ ab u_{i+1}

und sei $u_i \to u_{i+1}$ Nebenkante aus einem k. W. Sei also W k. W. mit

$$W = (1 = v_0, v_1, \dots, v_j, v_{j+1}, \dots, v_b = k)$$

mit $v_j = u_{i+1}$, $v_{j+1} = u_i$. Dann ist

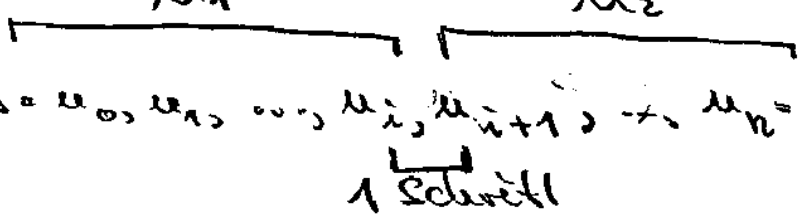
$$Lg(W_1) = i \geq j+1$$

da W k. W. ist! Ebenso

$$Lg(W_2) = k - i \geq k - j$$

Dann $Lg(W) \geq (j+1) + 1 + k - j = k + 2$.

Für beliebiges $l \geq 2$ gehen wir analog induktiv vor. Setze die Behauptung für alle Wege mit $l-1$ Umkehrkanten. Habe



$$W = (s = u_0, u_1, \dots, u_i, u_{i+1}, \dots, u_n = t)$$

genau l Umkehrkanten vor k. W's.

Sei $u_i \to u_{i+1}$ die erste Umkehrung auf W.

Sei also

$$W' = (s = v_0, \dots, v_j, v_{j+1}, \dots v_k = t)$$

k-W. mit $v_j = u_{i+1}$, $v_{j+1} = u_i$.

Wir betrachten

$$u' = (\underbrace{v_0 = s, \dots, v_j = u_{i+1}}_{\text{Von } u} , \underbrace{\dots, u_k = t}_{\text{Von } v})$$

u': ohne Wiederholung gem. (a). v: $\ell - 1$ Wiederholungen

Schauen uns die Längen an, dann ist nach Ind.-Vor.

$$L_g(u') \geq k + 2(\ell - 1)$$

Da u k-Wdh. ist (!) ist

$$L_g(u_1) = i \geq j + 1 .$$

Schließlich

$$\begin{aligned} L_g(u) &= L_g(u') - j + i + 1 \\ &\geq k + 2(\ell - 1) - j + (j + 1) + 1 \\ &= k + 2\ell . \end{aligned}$$

□ 87

(9.37)

Nun folgt direkt der

Beweis (des Satzes auf S. 9.27).

In jedem Lauf der Schleife löscht Edmonds Kopf mindestens eine Kante auf einem k.W. $\overset{\circ}{\circ}\!\!\rightarrow\!\!\overset{\circ}{\circ}$ und hängt die Umkehrkante ein.

(a) Fangen wir mit W_1 an, für $L_q(W_1) = k$. Zunächst betrachtet E. Kopf alle E-Wege der Länge k. Solange es solche Wege gibt wird nach letzterem Lemma keine Umkehrkante betreten. Irgendwann sind die

Wege der Länge k nicht
mehr möglich. Dann haben
wir nun noch k.W.'s der Länge
$k' \geq k+1$. Entf. mit Umkehrkanten
von k.W.'s der Länge k. W.
bearbeiten all diese Wege
und machen so weiter.

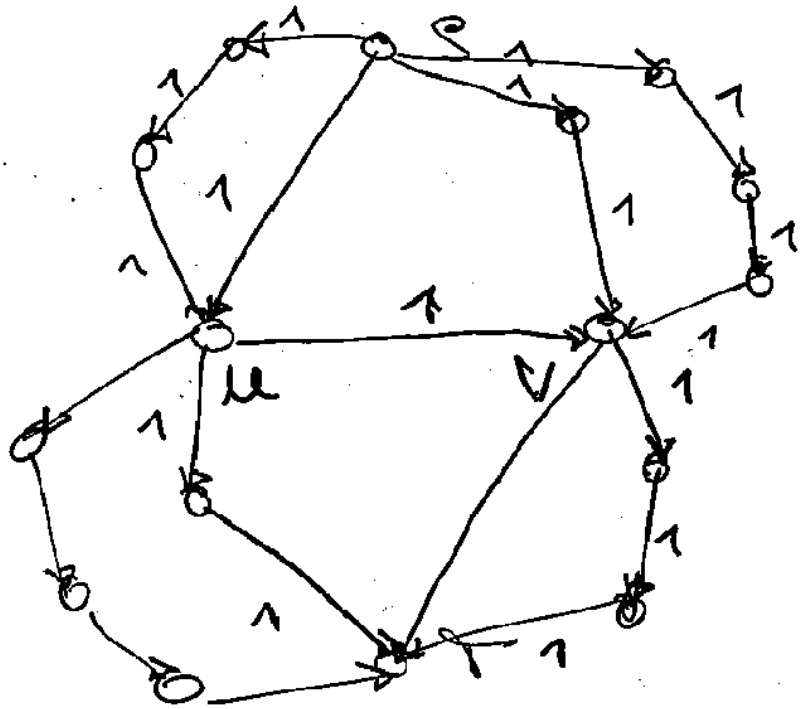
(b) Seien einmal
$W_1, \dots, W_l$ k.W.'s
in einem Restnetzwerk.
In jeder Runde nach Edmonds
Kap verschwindet mindestens
eine Kante auf den k.W.'s.
Nach $|E|$ Runden sind alle
Kanten auf den $W_1, \dots, W_l$ entfernt

Prinzip: Umkehrkanten von k.W.'s bringen keine Verbesserung k.W.'s ▽

(9.38)

Satz. Nach dem Lemma enthalten die U_δ jeweils keine Nullstellenketten. Also kommt in U_δ $|E|+1$ eine Nullstellenkette vor. Dann vergrößert sich die Länge. □

OSZILLATION BEI EDMONDS KARP



1. E-WEG (s, u, v, t) +1

2. E-WEG $(s, -, v, u, -, t)$ +1

3. E-WEG $(s, -, -, u, v, -, -, t)$ +1

4. E-WEG $(s, -, -, -, v, u, -, -, -, t)$ +1

⋮

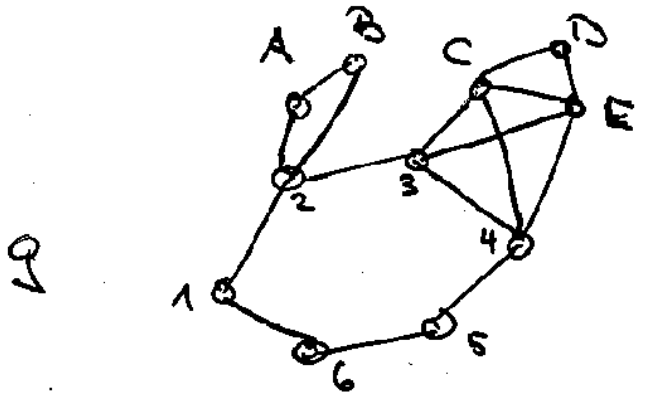
VERGLEICHE FOLIEN S. S. 23,

10. 111

NACHTRAG ZUM BEWEIS

AUF S. 10.92 / 93

GERADE GRADE ⇒ EULERSCHER KREIS



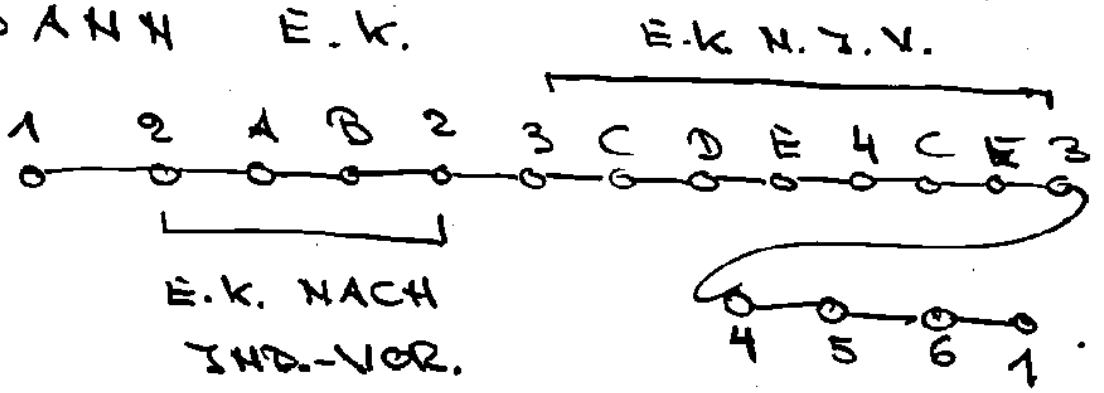
G

O.B.d.A. KEIN E.K

WEG ω ETWA

$$\omega = (1, 2, 3, 4, 5, 6, 1)$$

DANN E.K.



UNGLEICHE BRÜDER (10.12)

POLYNOMIAL

KÜRZESTER WEG (OHNE NEG. KREISE)

EULER KREIS

2-FÄRB-KEIT

2-SAT

MIN. SCHNITT

EXPONENTIELL

KÜRZ. WEG MIT NEG. KREISE IM GRAPH

LÄNGSTER KREISFREIER WEG

HAMILTON KREIS

3-FÄRB-KEIT

≥ 3-SAT

MAX-2-SAT

MAX. SCHNITT

- EXPO. ⇔ DURCHPROBIEREN UNVERMEIDBAR.
- INEINANDER ÜBERSETZBAR
- KEIN BEWEIS, DASS NUR EXPO (ABER VERMUTET)

MASTER THEOREM (11.88)

(KORREKTUR)

SEI

$$T(N) = \sum_{j=1}^{k} T(\alpha_j \cdot N) + \Theta \cdot N^L$$

$0 \leq \alpha_j \leq 1$. ALLES KONST,

AUSSER N. DANN

$$T(N) \begin{cases} \Theta(N^L) & \sum \alpha_j^L < 1 \\ \Theta(N^L \log N) & \sum \alpha_j^L = 1 \\ \Theta(N^C) & \sum \alpha_j^L > 1 \end{cases}$$

$\sum \alpha_j^C = 1$ BESTIMMT C.

M.89

BEISPIELE ZUM MASTER THEOREM

LINEARES SELECT : $L = 1$ $\Theta(N)$, $\Sigma = 2$

MERGESORT : $L = 1$, $\alpha_1 + \alpha_2 = 1$, $\Sigma = 2$

$$\Theta(N \cdot \log N)$$

MATRIXMULT: $L = 2$, $\Sigma = 7$

$$\alpha_j = 1/2 \cdot \sum_j \alpha_j^q = 1$$

$$\Rightarrow \alpha_j^c = 1/7$$

$$\Rightarrow 2^q = 7 \Rightarrow q = \log_2 7$$

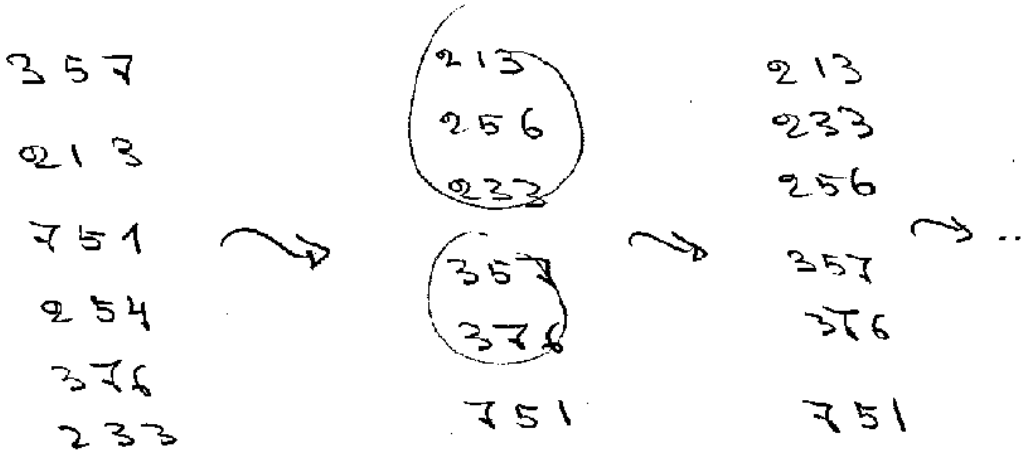
$$\Theta(N^{\log_2 7}).$$

LITERATUR : SCHÖNING

13.18

Bei Counting Sort werden die zu sortierenden Zahlen als Indices eines Arrays verwendet. Das geht bei größeren Zahlen nicht. Deshalb Radixsort (verwendet Counting Sort auf den Ziffern).

Ziffernweises Sortieren ist bekannt als Sortieren von der größten zur kleinsten Stelle:



(13.13)

Man muß die Stücke einzeln
sortieren. Dazu braucht man
#Stellen. #Ziffern viele pointer
in das zu sortierende array hinein.

Radixsort: Fängt auf der letzten
Stelle an und bearbeitet immer
hin zu den ganzen array.
Dabei wichtig: Stabilität
des Verfahrens für die
Ziffern (Counting Sort)
Also ein Beispiel:

Globel!

13.20

357
213
751
254
376
233

751
213
233
254
376
357

213
233
751
254
357
376

Globel

213
233
254

357
376
751

Invariante: Nach i-tem Lauf
kommt sortiert bzgl. der
letzten i Ziffern.

Algorithmus (Radixsort)

Eingabe: array $A[1..m]$ of subtype
und d = # Ziffern.

Ausgabe: A sortiert.

1. for $i := 1$ to d // Letzte Stelle 1.
// d Stelle 1.
2. Sortiere A auf Stelle i
mit Sortierung Sort □

Zeit: Ziffern aus $\mathbb{Q}, -, k$
dann $O(d \cdot (m+k)) = O(d \cdot m)$
bei k konstant (etwa $k = 2$)
sind die Zahlen groß, also von
der Größe $O(n)$, ist $d = O(\log m)$
und wir haben auch nur $O(m \cdot \log m)$. 33