Theoretische Informatik I

Nachfolgerveranstaltung zu Algorithmen und Programmierung im 1. Semester und Datenstrukturen im 2. Semester. Inhalt der Vorlesung Theoretische Informatik I ist das Gebiet der effizienten Algorithmen, insbesondere Graphalgorithmen und algorithmische Techniken. Gemäß dem Titel Theoretische Informatik I liegt der Schwerpunkt der Vorlesung auf beweisbaren Aussagen über Algorithmen.

Der Vorlesung liegen die folgenden Lehrbücher zu Grunde:

Uwe Schöning	Algorithmik, Spektrum Verlag, 2001
Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest	Algorithms *), The MIT Press, 1990
Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman	The Design and Analysis of Computer Algorithms, Addison Wesley, 1974
Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman	Data Structures and Algorithms, Addison Wesley, 1983

*) Auf deutsch im Oldenburg Verlag.

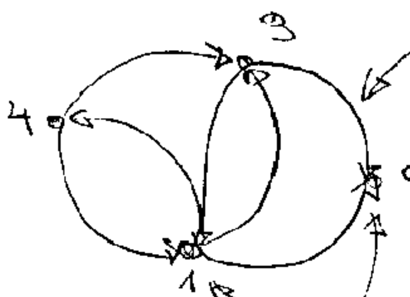
Besonders die beiden erstgenannten Bücher sollte jeder Informatiker einmal gesehen (=teilweise durchgearbeitet) haben. Weitere deutschsprachige Bücher sind:

Thomas Ottmann, Peter Widmayer	Algorithmen und Datenstrukturen BI Wissenschaftsverlag 1993
Volker Heun	Grundlegende Algorithmen Vieweg Verlag 2000

Inhaltsübersicht

1. Grundlagen: Gerichtete Graphen

Ein typischer gerichteter Graph:

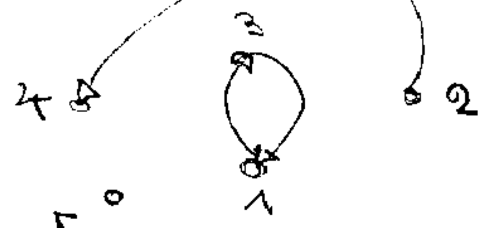


Kante (directed edge, directed arc)

G_1

Knoten (vertex, node)

Ein anderer

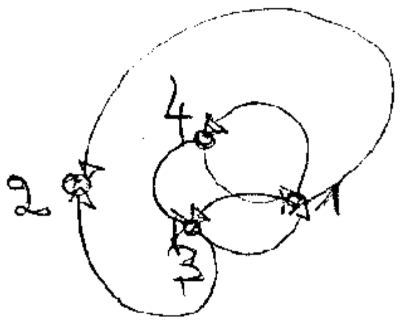


G_2

$V = \{1, 2, 3, 4, 5\}$

$E = \{(3,1), (1,3), (2,4)\}$

Derselbe, wie der erste



G_1

$V = \{1, 2, 3, 4\}$

$E = \{(4,1), (1,4), (3,1), (1,3), (4,3), (1,2), (2,3)\}$

Edge = Kante!

<u>Definition</u>

Ein gerichteter Graph G besteht aus 2 Mengen

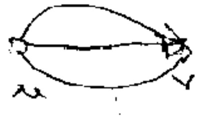
- V eine beliebige, endliche Menge von Knoten
- E eine Menge von Kanten, wobei

$$E \subseteq \{(u,v) \mid u,v \in V,\ u \neq v\}$$

Schreibweise $G = (V, E)$. □

$(u,v) \rightsquigarrow \underset{u}{\circ} \longrightarrow \underset{v}{\circ}$, $(v,u) \rightsquigarrow \underset{v}{\circ} \longrightarrow \underset{u}{\circ}$

Hier einmal nicht(!) betrachtet wollen Schleifen $(u,u) \rightsquigarrow$  und

Mehrfachkanten  etwa $(u,v), (u,v), (u,v)$.

Folgerung

Ist $|V| = n$, so hat jeder gerichtete Graph mit Knotenmenge V $\leq n(n-1)$ Kanten.

gerichtete Graphen bei n Knoten $= 2^{n(n-1)}$. □

$n(n-1) = n^2 - n$ ist $O(n^2)$.

Erinnerung

$f(n)$ ist $O(n^2)$ bedeutet:

Es gibt eine Konstante C, so daß

$$f(n) \leq C \cdot n^2$$

(für alle hinreichend großen n).

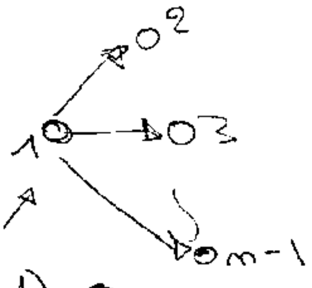
(1.4)

Ist die Kante (u,v) in E vorhanden,
so sagt man:

- v ist adjazent zu u.
- (u,v) ist eine Kante inzident mit u und inzident mit v.

- $\text{Ausgangsgrad}(v) = |\{(v,u) \mid (v,u) \in E\}|$
- $\text{Eingangsgrad}(u) = |\{(v,u) \mid (v,u) \in E\}|$

Immer $0 \leq \text{Agrd}(v) \leq n-1$,
$0 \leq \text{Egrd}(u) \leq n-1$.



$\text{Egrd}(1) = 0$
$\text{Agrd}(1) = n-1$.

Definition (Weg)

Eine Folge von Knoten $(v_0, v_1, \dots, v_k)$

ist ein Weg in $G = (V, E)$

gdw.

$v_0 \to v_1 \to v_2 \to \dots \to v_k$

Kanten in E sind. □

Ist $E = \{(u,v), (v,w), (w,v), (v,u)\}$,

so ist auch

$u \to v \to u \to v \to w \to v$

ein Weg.

$$\text{Länge}(v_0 \to v_1 \to v_2 \to \dots \to v_k) = k$$

Also Länge = # Schritte. Länge $v_0 = 0$.

Weg $(v_0, v_1, \dots, v_k)$ einfach

gdw. $|\{v_0, v_1, \dots, v_k\}| = k+1$

(alle v_i verschieden). Weg

$(v_0, \dots, v_k)$ geschlossen gdw. $v_0 = v_k$

Weg $(v_0, v_1, \dots, v_k)$ Kreis

gdw.

- $k \geq 1$
- $(v_0, v_1, \dots, v_{k-1})$ einfach
- $v_0 = v_k$.

$(1, 2, 1)$
$v_0 \; v_1 \; v_2 = v_0$

Sogar $k \geq 2$, da bei $k = 1$ (v_0, v_0), was nicht zugelassen ist

Beachte noch einmal Länge: Ein Weg

$(v_0, \dots, v_k)$ haben nur $k+1$ v_i's

aber nur k Schritte $(v_i \to v_{i+1})$.

(1.7)

Die Kunst des Zählens:

Auswahl von Elementen

1. Bei $|V| = n$ haben wir genau $n \cdot (n-1)$ gerichtete Kanten.

Eine Kante = ein Paar (v, w)

$v, w \in V$, $v \neq w$.

Auswahl

Möglichkeiten für v: n.

Ist v gewählt, dann

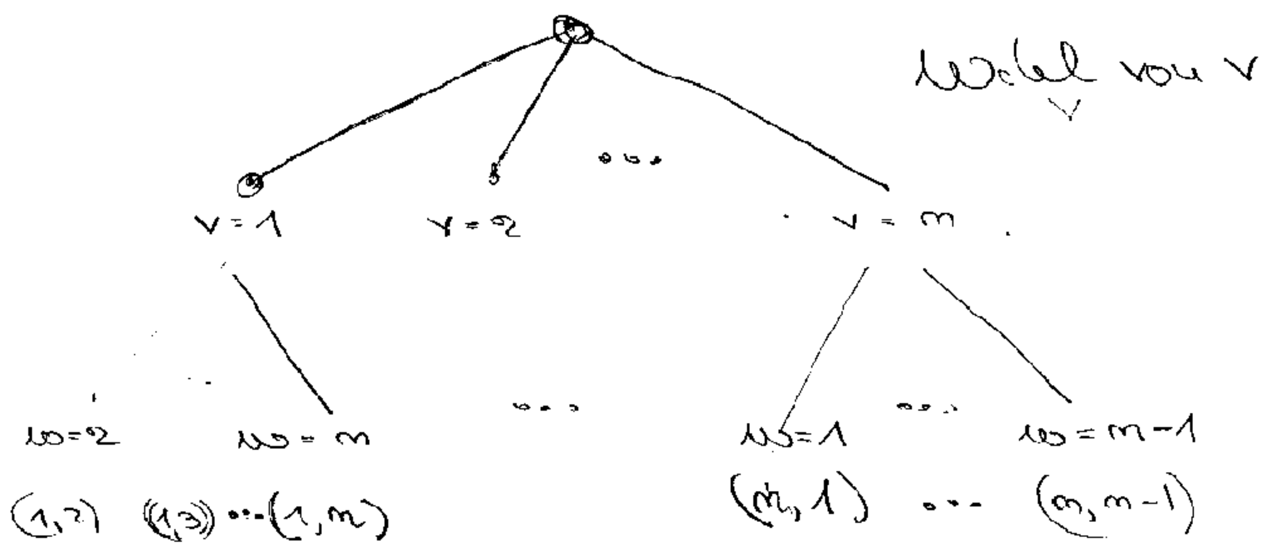
Möglichkeiten für w: $n-1$ (!)

Jede Wahl erzeugt genau eine Kante. Jede Kante wird genau einmal erzeugt.

Multiplikation der Möglichkeiten gibt $n(n-1)$.

(1.8)

Veranschaulichung durch „Auswahlbaum" für Kante (v, w)



jedes Blatt = genau eine Kante

$\#$ Blätter $= n(n-1) = n^2 - n$.

Interpretation von $n^2 - n$:

$n^2 = \#$ alle Paare (v, w), $v, w \in V$ (auch $v = w$)

$n = \#$ Paare (v, v), $v \in V$. → Werden von n^2 abgezogen.

(1.2)

2. Bei einer Menge M mit $|M| = m$ haben wir genau 2^m Teilmengen.

Teilmenge von M = Bitstring der Länge m

$0\ 0\ 0 \ldots 0 \quad \emptyset \quad M = \{a_1, \ldots, a_m\}$

$a_1 \qquad a_m$

$0\ 0\ 0 \ldots 1 \quad \{a_m\}$

$0\ 0\ 0 \ldots 1\ 0 \quad \{a_{m-1}\}$

$\vdots$

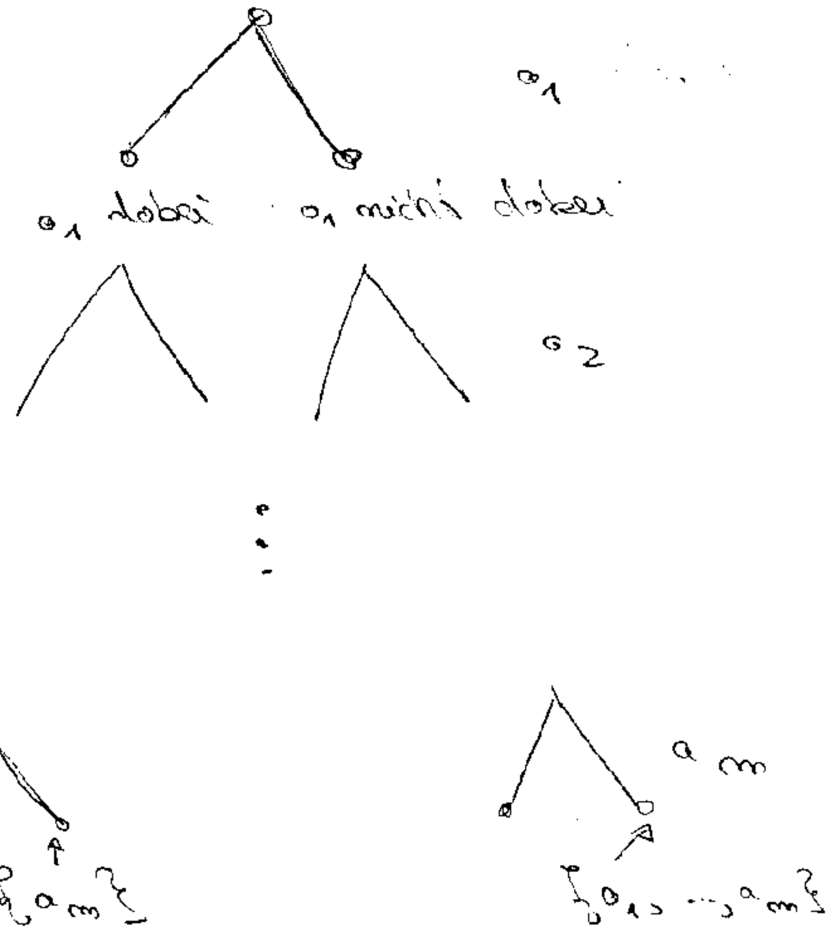
$1\ 1\ 1 \ldots 1\ 1 \quad \{a_1, \ldots, a_m\} = M.$

2^m Bitstrings der Länge m.

Also # gerichtete Graphen bei $|V| = n$ ist genau $2^{n(n-1)}$.

(1.10)

Nach dem Auswahlbaum für die Teilmengen von M:



Tiefe m, # Blätter $= 2^m$.

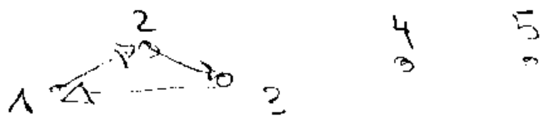
Datenstrukturen (Adjazenzmatrix)

Darstellung von $G = (V, E)$ mit $V = \{1, \dots, n\}$. Matrix

$$A = (a_{u,v})_{\substack{1 \leq u \leq n \\ 1 \leq v \leq n}}, \quad a_{u,v} \in \{0,1\}$$

$$a_{u,v} = 1 \quad \text{gdw.} \quad (u,v) \in E. \quad \square$$

- Implementierung durch array A[][].



$$A = \begin{array}{c|cccc} & 1 & 2 & 3 & 4 \\ \hline 1 & 0 & 1 & & \\ 2 & & 0 & 1 & 0 \\ 3 & 1 & 0 & 0 & \\ 4 & & & & 0 \end{array}$$

$(1000)^2 = 1000000$

(1-12)

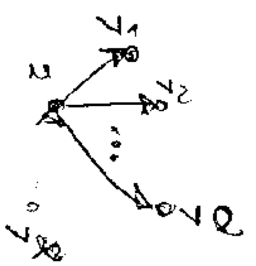
A hat immer n^2 Speicherplätze, egal wie groß $|E|$ ist. Jedes $|E| \geq n-1$ ist sinnvoll.

Datenstrukturen (Adjazenzlisten)

Sei $G = (V, E)$ ein gerichteter Graph.

- Adjazenzliste von u

  = Liste der direkten Nachbarn von u.



dann $v_1 \to v_2 \to v_3 \to \dots \to v_l$

Nicht v_k, jede Reihenfolge von $v_1, \dots, v_l$.

• Adjazenzlistendarstellung von G:

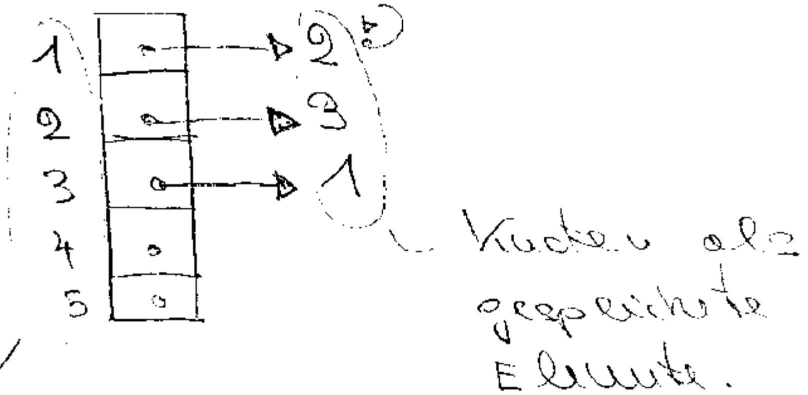
= array Adj, dessen Indices für V stehen,

Adj[v] zeigt auf Adjazenzliste von v.

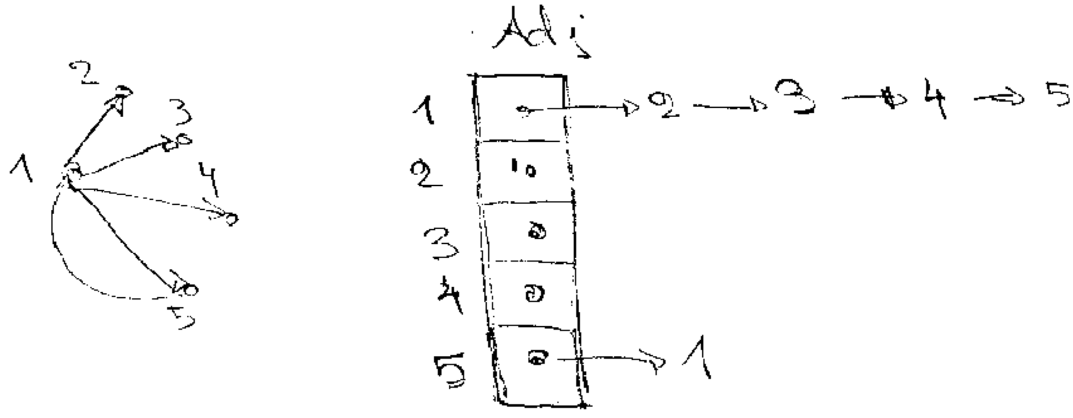
□

Beispiel mit dem

Array Adj



Knoten als Indices (also Adressen!)



(1.14)

Platz zur Darstellung von

$G = (V, E)$: $c + n + d \cdot |E|$

Initialisierung   Array A   Jede Kante einmal, d etwa 2

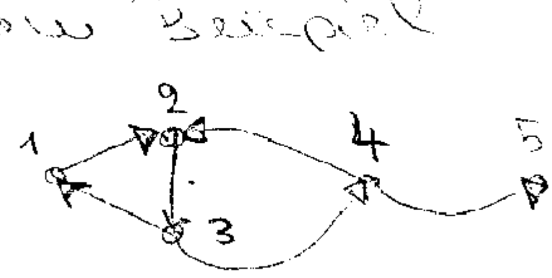
$\mathcal{O}(n + |E|)$

$\mathcal{O}(|E|)$ wenn $|E| \geq n-1$.

Was, wenn keine pointer?

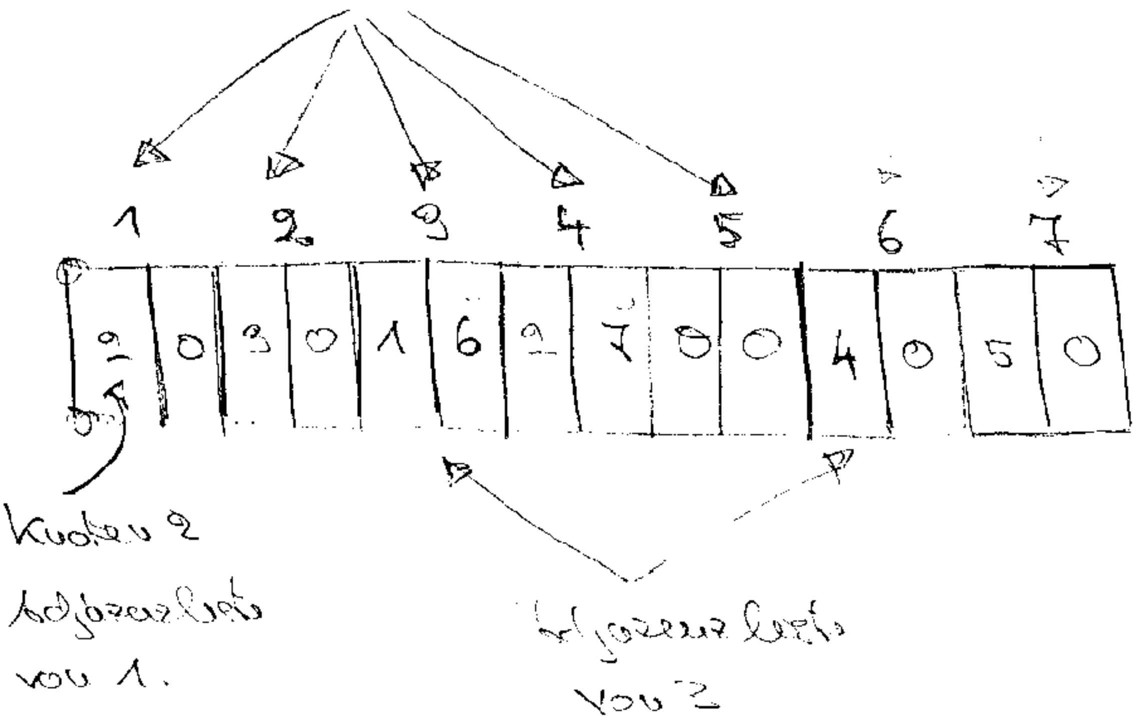
Adjazenzlistendarstellung in arrays!

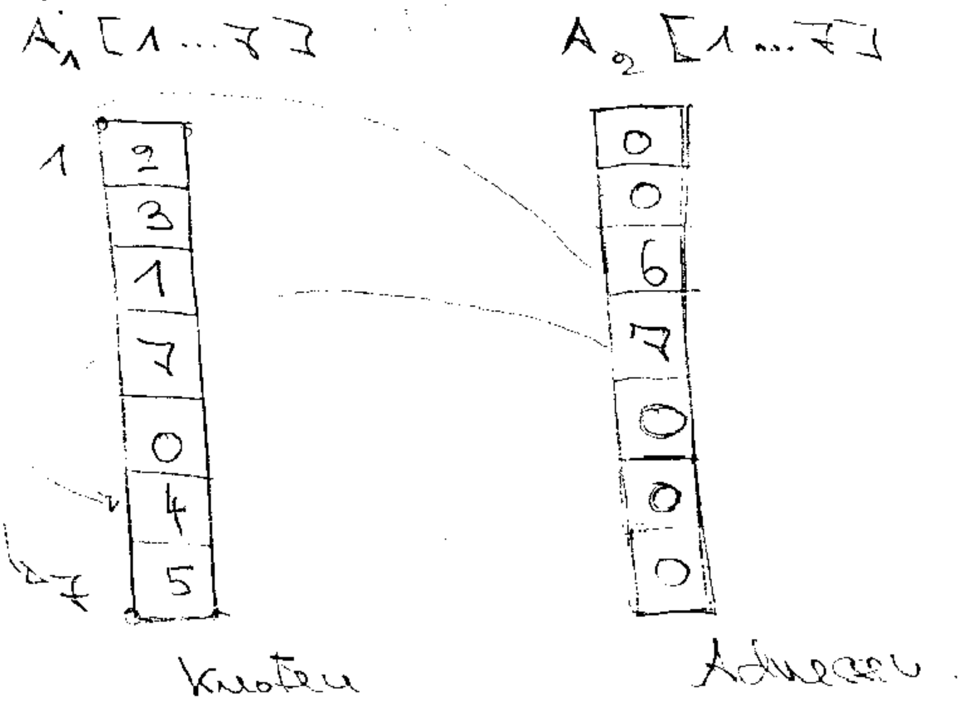
An einem Beispiel

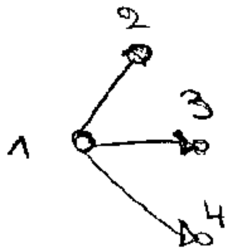
$G =$ 

6 Kanten, also array $A[1 \dots 6]$

1.15


Knoten.
1 2 3 4 5 6 7
2 0 3 0 1 6 2 7 0 0 4 0 5 0
Knoten 2
Adjazenzliste von 1.
Adjazenzliste von 2


A1 [1 ... 7]
A2 [1 ... 7]
1
7
2 3 1 7 0 4 5
0 0 6 7 0 0 0
Knoten
Adressen.



1		2		3		4		5	6		
2	5	0	0	0	0	0	0	4	6	3	0

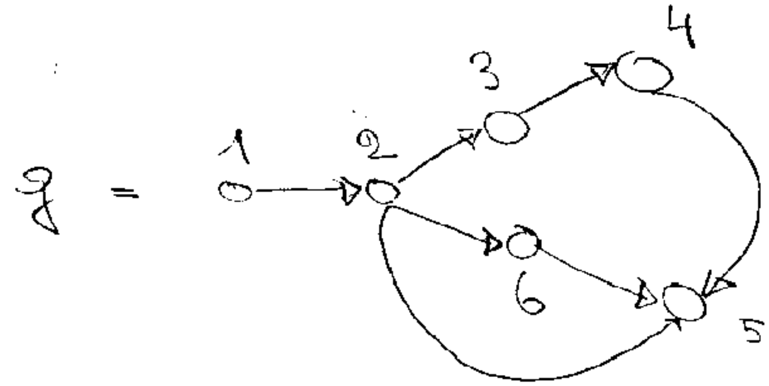
Platz $\leq 2 \cdot |V| + 2 \cdot |E|$

also $O(|V| + |E|)$ reicht auf.

Gibt es einen Weg von u nach v im gerichteten Graphen $G = (V, E)$?

Vorgehensweise: Schrittweises Entdecken der Struktur.

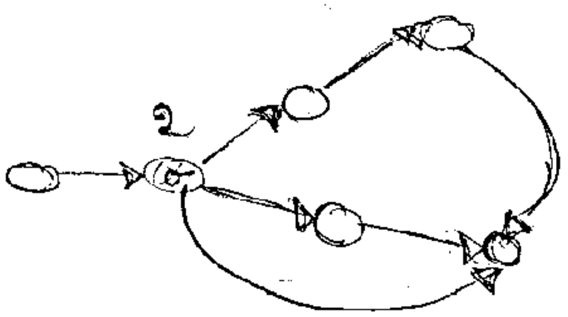
Dazu ein Beispiel



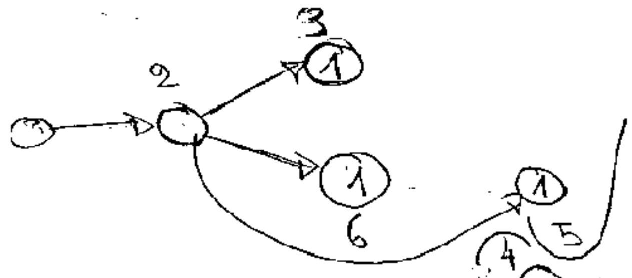
Weg von 2 nach 4 (4 von 2 erreichbar)?

Entdecktiefe

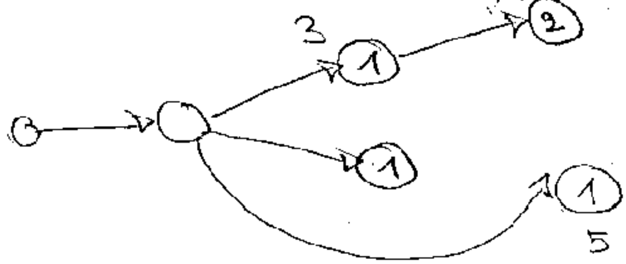
0



1



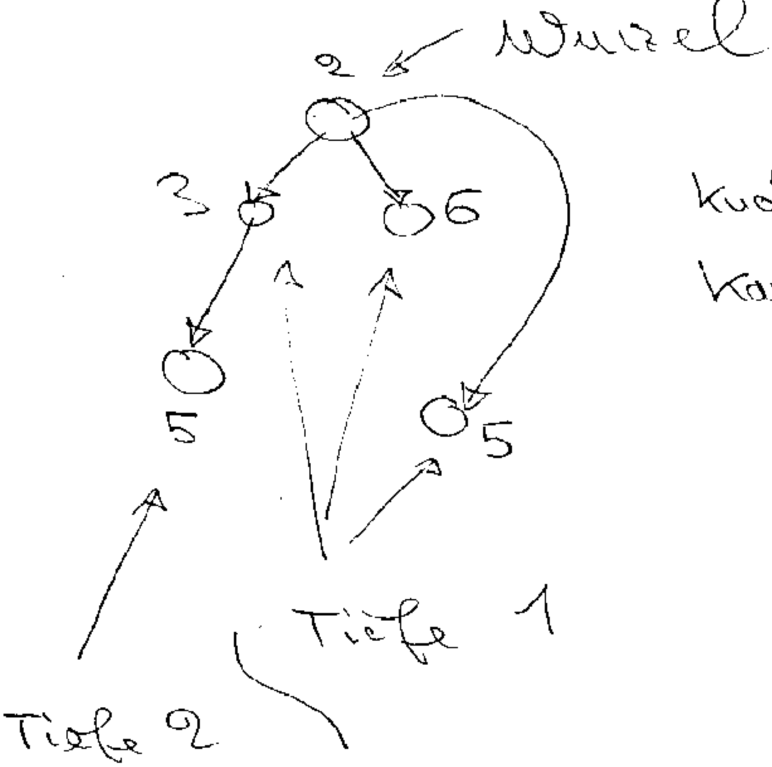
2



Breitensuchbaum

= Kanten, über die entdeckt wurde

Teilgraph von G



Knoten Teilmenge.

Kanten Teilmenge.

1.13

Implementierung mit einer Schlange.

<u>Datenstruktur</u> (Schlange)

Array Q[1...n] mit Zeigern in Q hinein, head, tail.

Einfügen am Ende (tail),
Löschen vorne (head)

1	2	3	4
5			

head = 1, tail = 2

7, 5 einfügen

5	7	5	

head = 1, tail = 4

1.20

Löschen; 8 einfügen

	7	5	8

head = 2, tail = 1

Lösche

tail geht neu

		5	8

head = 3, tail = 1

Löschen 2-mal

head = 1, tail = 1

Schlange leer!

Beachte: Beim Einfügen immer head ≠ tail, sonst geht es nicht mehr. tail ist immer der nächste freie Platz.

tail < head ⟺ Schlange geht ums Ende.

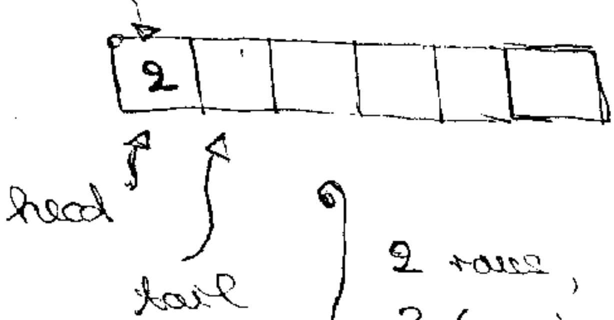
First-in First-out = Schlange

II

Am Beispiel von 2 oben

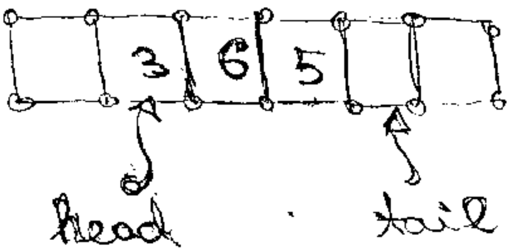
die Schlange im Verlauf der Breitensuche.

Knoten 2

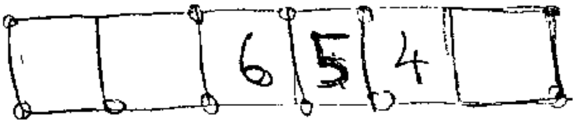


Bei $|V| = n \geq 2$ reichen n Plätze. Eins bleibt immer frei

2 raus,
3, 6 rein

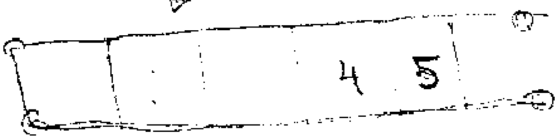


3 raus
4 rein



5, 6 hat Entdecktiefe 1
4 Entdecktiefe 2

6 raus
nichts rein



4 raus, 5 raus,
Schlange leer. head = tail = 6.