Verteilte Algorithmen und Datenstrukturen

Kapitel 2: Netzwerktheorie

Prof. Dr. Christian Scheideler

Institut für Informatik

Universität Paderborn

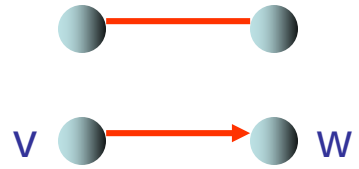
Übersicht

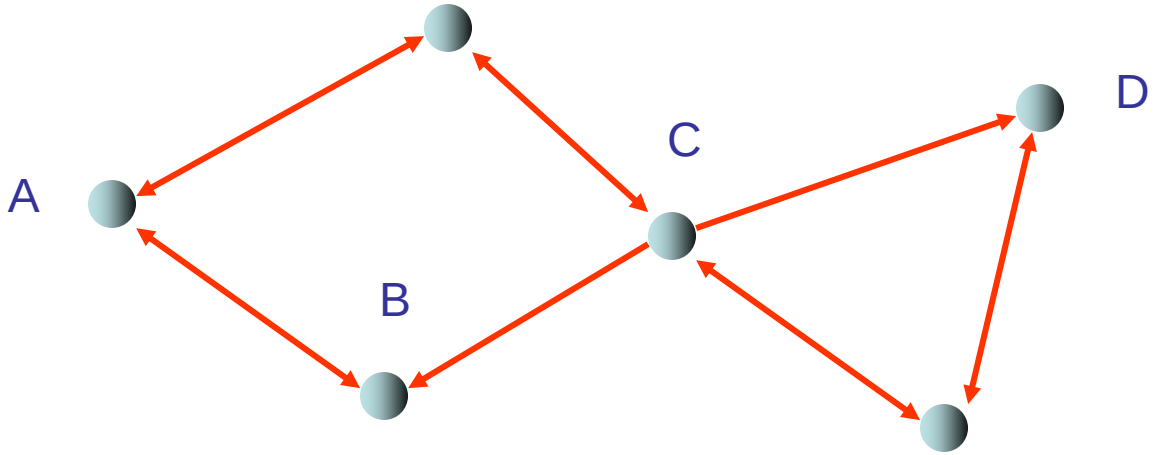
- Grundlagen
- Grundlegende Graphparameter
- Klassische Graphfamilien
- Skip Graphen
- Routing

Grundlagen

Definition 2.1: Ein Graph G=(V,E) besteht aus einer Knotenmenge V und Kantenmenge E.

- G ungerichtet: $E \subseteq \{ \{v,w\} \mid v,w \in V\}$
- G gerichtet: $E \subseteq \{ (v,w) \mid v,w \in V\}$

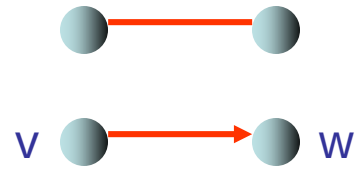




Grundlagen

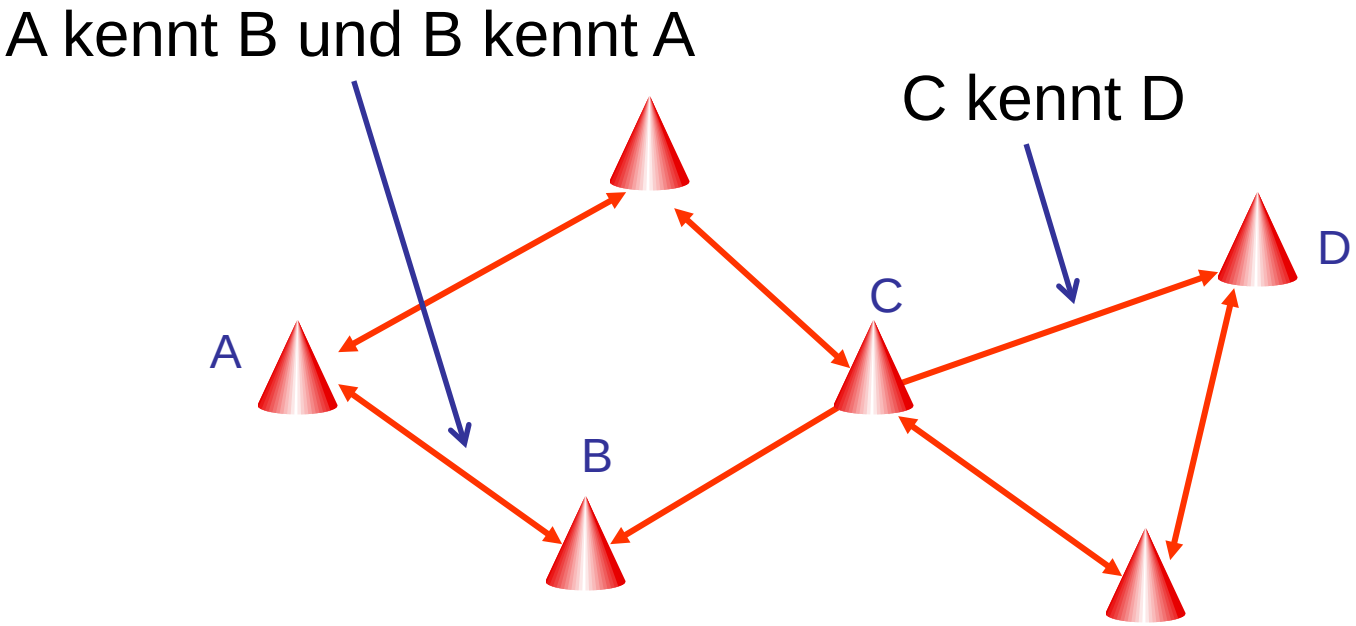
Definition 2.1: Ein Graph $G=(V,E)$ besteht aus einer Knotenmenge V und Kantenmenge E.

- G ungerichtet: $E \subseteq \{ \{v,w\} \mid v,w \in V\}$
- G gerichtet: $E \subseteq \{ (v,w) \mid v,w \in V\}$
- G bigerichtet: für alle $(v,w) \in E$ ist auch $(w,v) \in E$



Grundlagen

Graph: repräsentiert Wissen über bzw. Verbindungen zwischen Prozessen

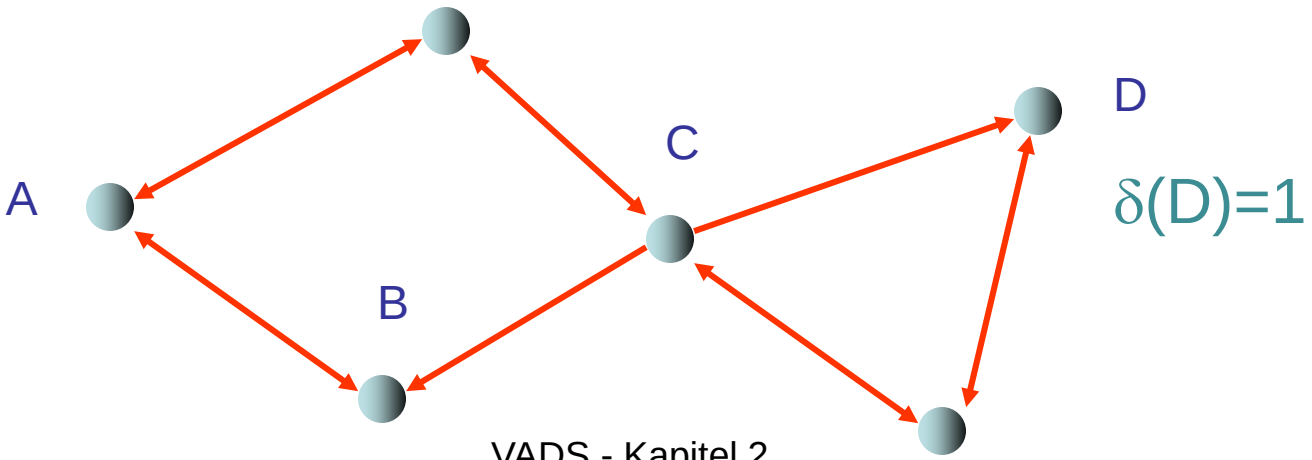


Grundlagen

Definition 2.2: Sei $G=(V,E)$ ein Graph.

- G ungerichtet: Grad von $v \in V$:
$$\delta(v)=|\{ w \in V \mid \{v,w\} \in E\}|$$
- G gerichtet: Grad von $v \in V$:
$$\delta(v)=|\{ w \in V \mid (v,w) \in E\}|$$

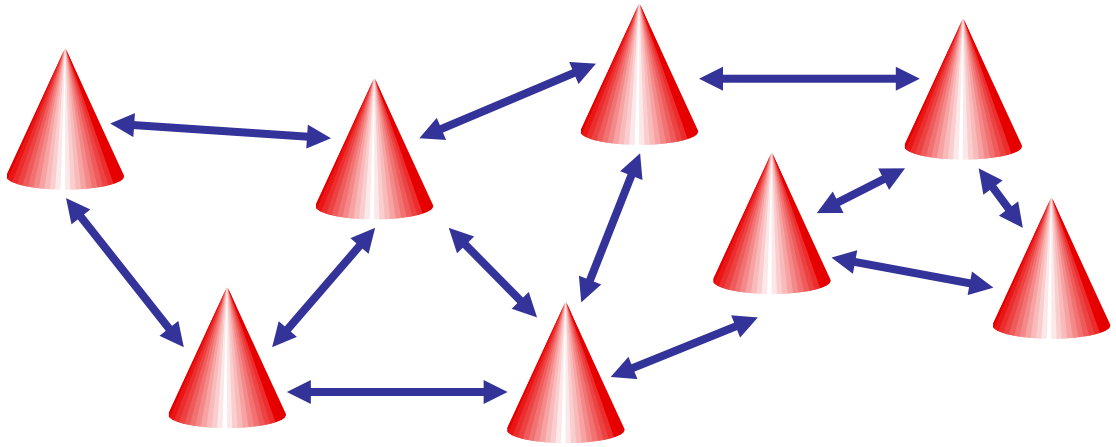
Grad von G: $\Delta = \max_{v \in V} \delta(v)$



Grundlagen

Grad:

- worst-case Aufwand pro Prozess für Kontrolle seiner Verbindungen
- in bigerichteten Graphen, worst-case update Kosten für Prozess, falls sich Menge der Prozesse verändert.

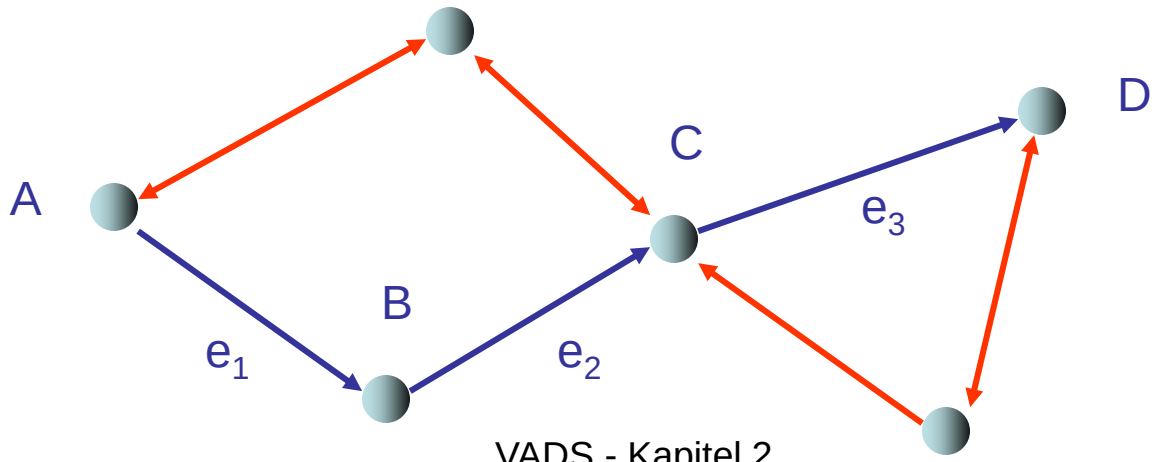


Grad sollte nicht zu hoch sein.

Grundlagen

Definition 2.3: Sei $G=(V,E)$ ein Graph. Eine Kantenfolge $p=(e_1,e_2,\ldots,e_k)$ in G heißt Weg/Pfad, falls es eine Knotenfolge $(v_0,\ldots,v_k)$ gibt mit

- G ungerichtet: $e_i=\{v_{i-1},v_i\}$ für alle $i\in\{1,\ldots,k\}$
- G gerichtet: $e_i=(v_{i-1},v_i)$ für alle $i\in\{1,\ldots,k\}$



Grundlagen

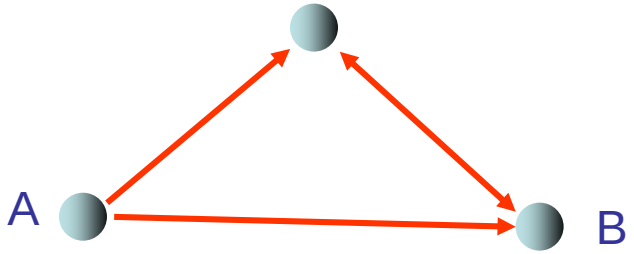
Definition 2.4: Ein Graph $G=(V,E)$ heißt

- zusammenhängend, wenn G ungerichtet ist und für jedes Knotenpaar $v,w \in V$ ein Pfad von v nach w in G existiert.
- schwach zusammenhängend, wenn G gerichtet ist und für jedes Knotenpaar $v,w \in V$ ein Pfad von v nach w in der ungerichteten Version von G existiert
- stark zusammenhängend, wenn G gerichtet ist und für jedes Knotenpaar $v,w \in V$ ein Pfad von v nach w in G existiert

Grundlagen

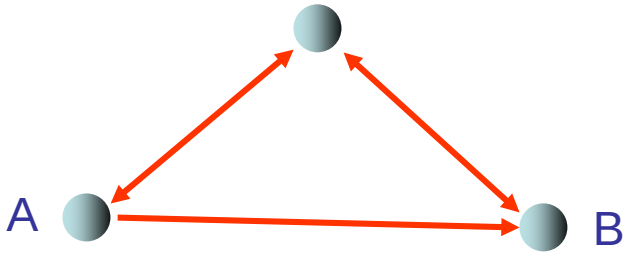
Beispiele:

(1) Graph nur schwach zusammenhängend



kein gerichteter Weg
von B nach A

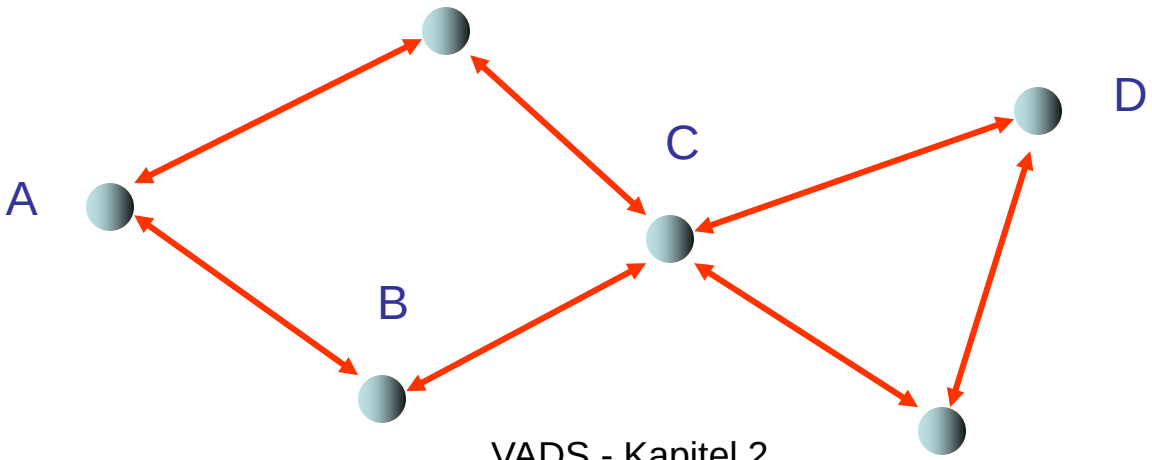
(2) Graph stark zusammenhängend



Grundlegende Graphparameter

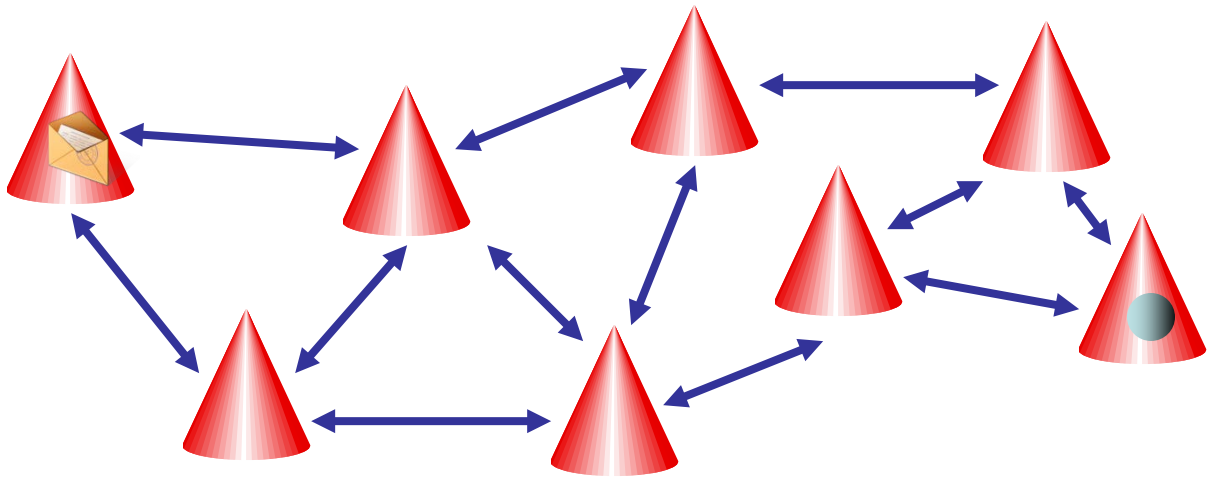
Definition 2.5: Sei $G=(V,E)$ ein Graph und $p=(e_1,e_2,\ldots,e_k)$ ein Weg von v nach w in G.

- Länge von p: $|p|=k$
- Distanz von w zu v: $d(v,w)$ = min. Weglänge von v nach w ($d(v,w) = \infty$ falls kein Weg von v nach w existiert)
- Durchmesser von G: $D(G)=\max_{v,w\in V} d(v,w)$



Grundlegende Graphparameter

Durchmesser: untere Schranke für worst-case Zeit (gemessen in Kommunikationsrunden) für Zugriff auf anderen Prozess

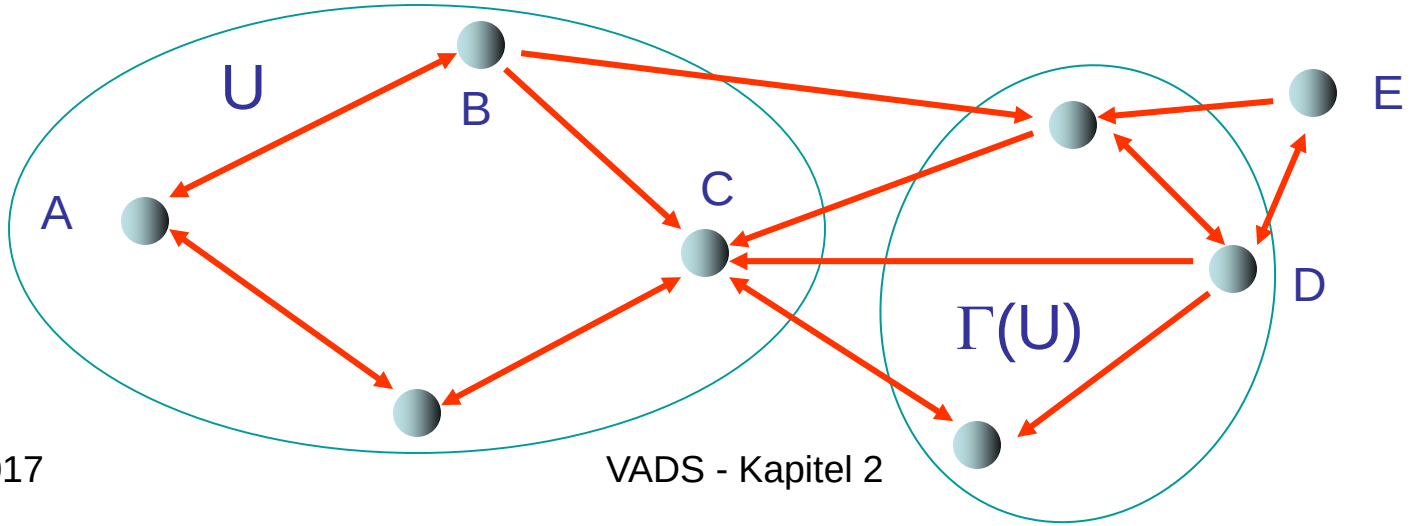


Durchmesser sollte nicht zu hoch sein.

Grundlegende Graphparameter

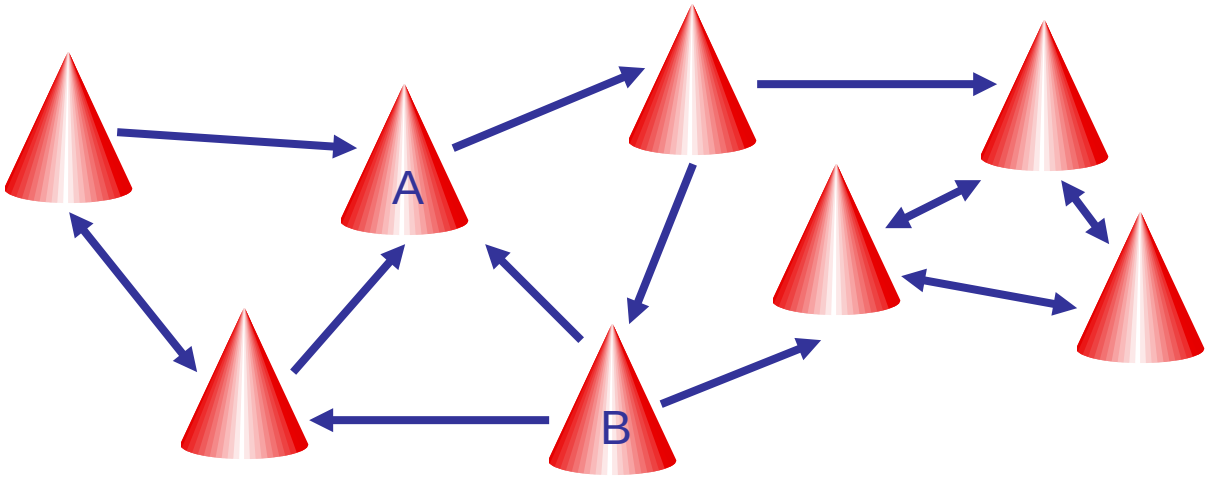
Definition 2.6: Sei $G=(V,E)$ ein Graph.

- $\Gamma(U)$: Nachbarmenge einer Knotenmenge $U \subseteq V$, d.h.
  $\Gamma(U) = \{ w \in V \setminus U \mid$ es gibt $v \in U$ mit $\{v,w\} \in E$ (bzw. $(v,w) \in E$ oder $(w,v) \in E$ im gerichteten Fall) $\}$
- $\alpha(U) = |\Gamma(U)| / |U|$: Expansion von U
- $\alpha(G) = \min_{U,|U| \leq \lceil |V|/2 \rceil} \alpha(U)$: Expansion von G



Grundlegende Graphparameter

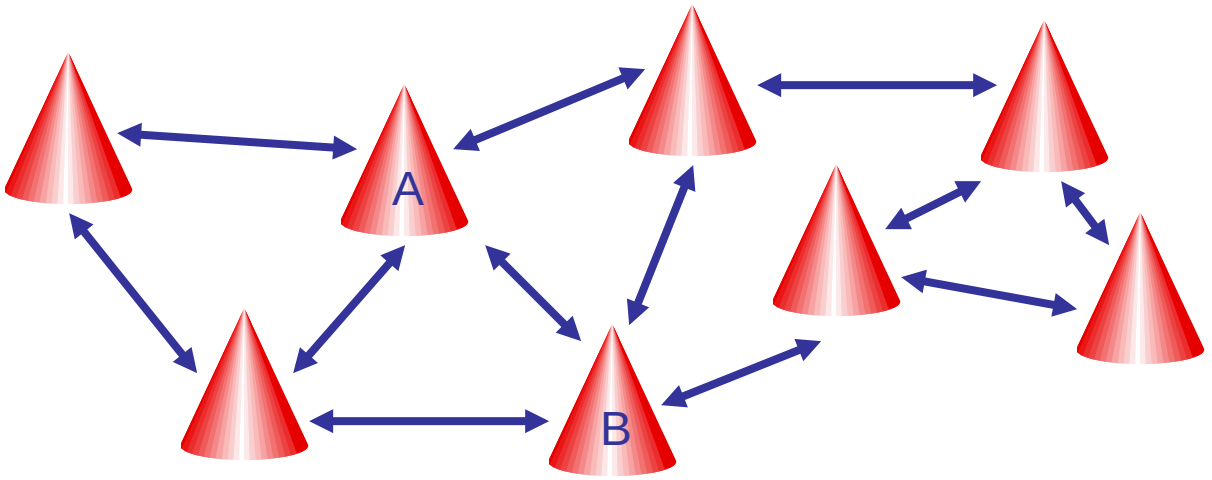
Expansion: k Ausfälle $\Rightarrow$maximal $k/\alpha(G)$ Knoten werden vom Graphen abgetrennt



Beweis: Sei U eine Menge nicht augefallener Knoten, die durch Ausfälle abgetrennt werden. Dann müssen alle Knoten in $\Gamma(U)$ ausfallen, d.h. $|\Gamma(U)| \leq k$. Weiterhin ist $\alpha(U) \geq \alpha(G)$ und $\alpha(U)=|\Gamma(U)|/|U|$, also $|U| \leq k/\alpha(G)$.

Grundlegende Graphparameter

Expansion: k Ausfälle $\Rightarrow$maximal $k/\alpha(G)$ Knoten werden vom Graphen abgetrennt

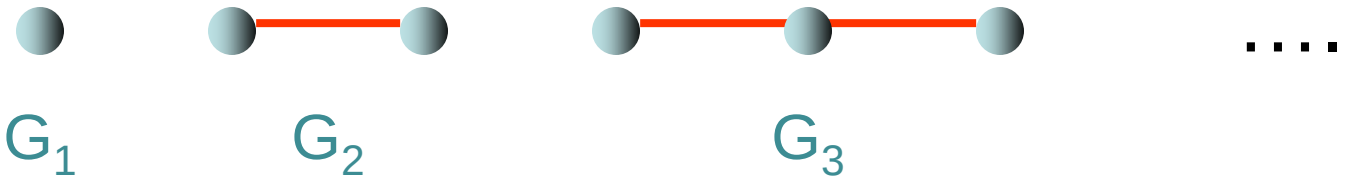


Expansion sollte möglichst groß sein

Klassische Graphfamilien

Im folgenden betrachten wir klassische Familien ungerichteter Graphen $\mathcal{G}=\{G_1, G_2, \ldots\}$.

Beispiel: Familie der linearen Listen

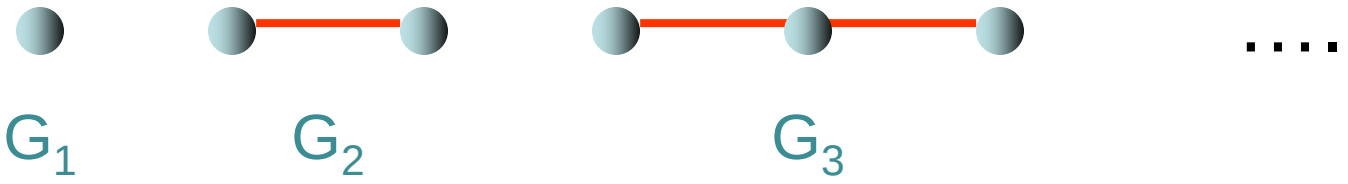


Wir sagen: Graph G aus einer Familie $\mathcal{G}$ hat konstanten Grad, falls der Grad aller Graphen in $\mathcal{G}$ durch eine Konstante beschränkt ist.

Klassische Graphfamilien

Im folgenden betrachten wir klassische Familien ungerichteter Graphen $\mathcal{G}=\{G_1, G_2, \ldots\}$.

Beispiel: Familie der linearen Listen

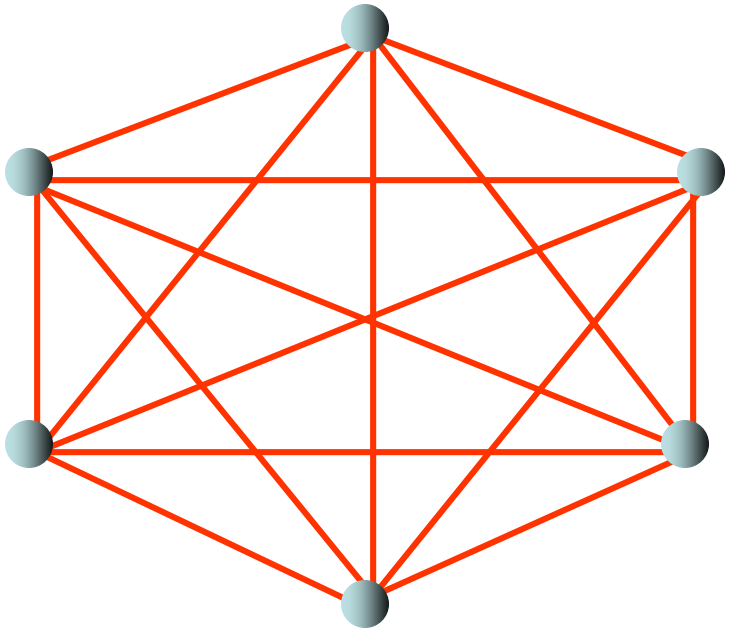


Für einen Graphen G aus $\mathcal{G}$ bezeichnen wir mit

- n: Anzahl der Knoten (bzw. Größe) von G
- m: Anzahl der Kanten von G

Clique

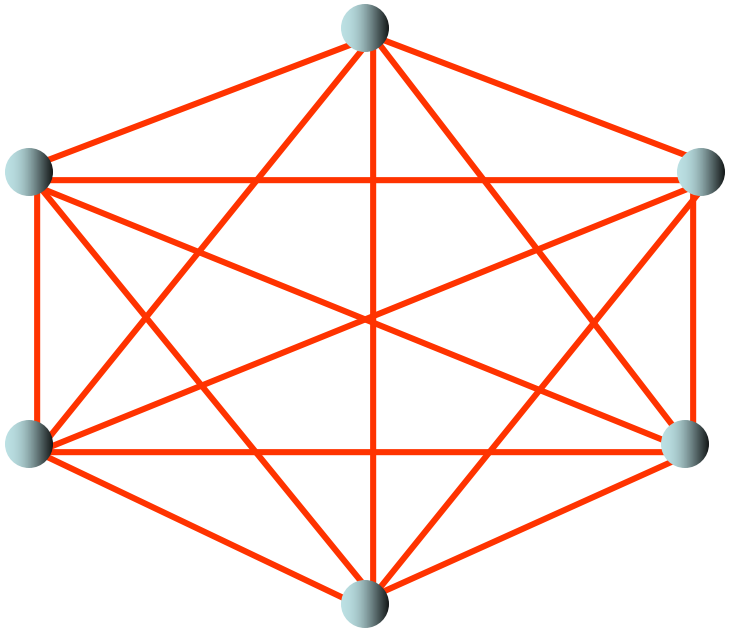
Vollständiger Graph / Clique: jeder Knoten ist mit jedem anderen verbunden



Vorteil: niedriger Durchmesser, hohe Expansion

Clique

Vollständiger Graph / Clique: jeder Knoten ist mit jedem anderen verbunden



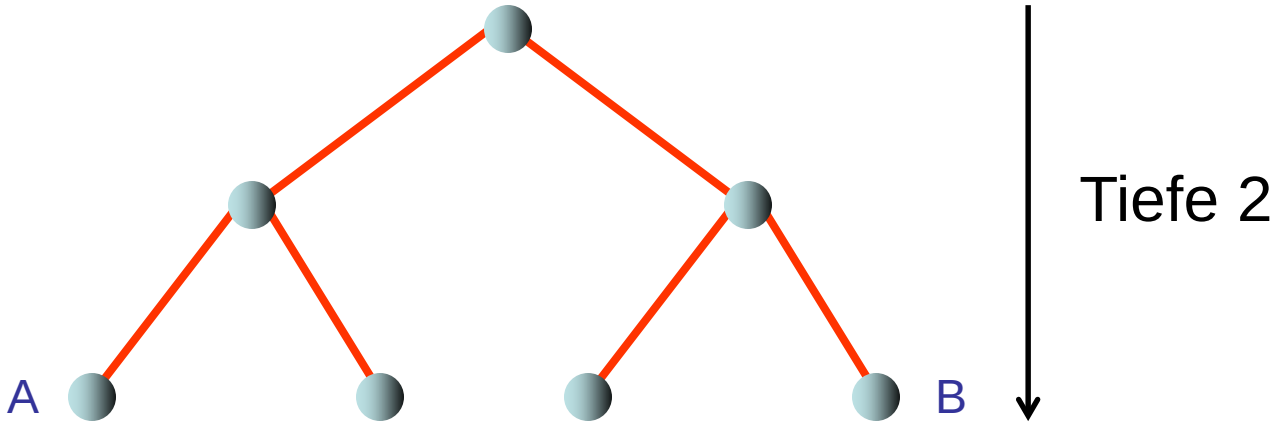
Problem: hoher Grad! ($\delta(v)=n-1$ für alle v)

Lineare Liste

- Grad 2 (minimal für Zusammenhang), ABER
- Durchmesser schlecht (D(Liste)=n-1)
- Expansion schlecht (α(Liste)$\approx$2/n)

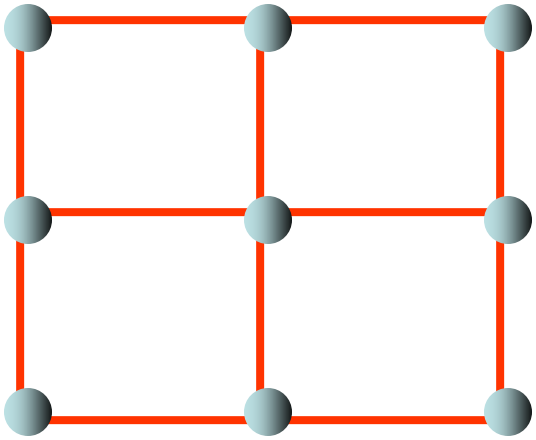
Wie erhält man kleinen Grad und Durchmesser?

Vollständiger binärer Baum



- $n=2^{k+1}-1$ Knoten bei Tiefe $k \in \mathbb{N}_0$, Grad 3
- Durchmesser ist $2k \approx 2 \log_2 n$, ABER
- Expansion schlecht ($\alpha(\text{Baum}) \approx 2/n$)

2-dimensionales Gitter



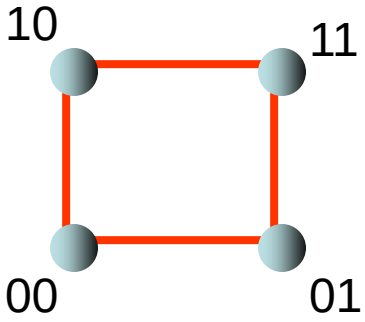
- $n = k^2$ Knoten bei k Knoten pro Seite, maximaler Grad 4
- Durchmesser ist $2(k-1) \approx 2\sqrt{n}$
- Expansion ist $\approx 2/\sqrt{n}$
- Nicht schlecht, aber geht es besser?

d-dimensionaler Hypercube

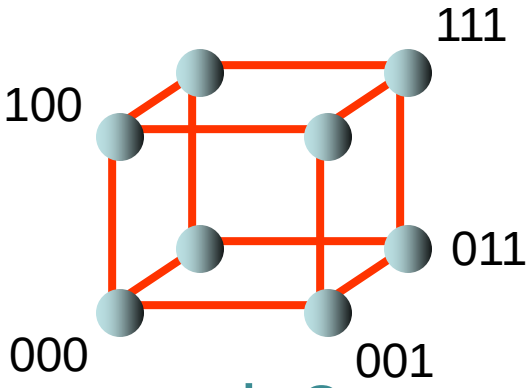
- Knoten: $(x_1,\ldots,x_d) \in \{0,1\}^d$
- Kanten: $\forall i: (x_1,\ldots,x_d) \rightarrow (x_1,..,1\text{-}x_i,..,x_d)$
- ungerichtete Hypercubes:

nur Bit i gedreht

d=1



d=2

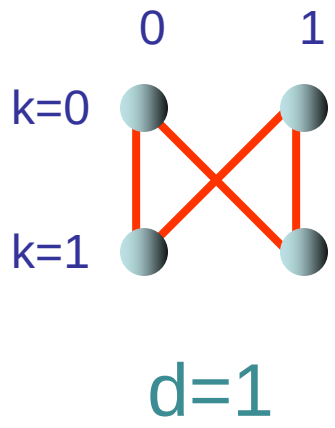


d=3

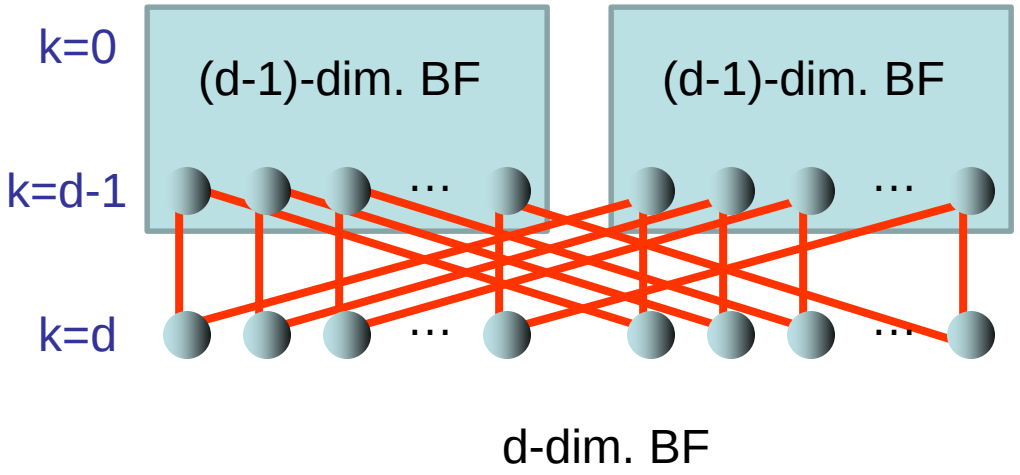
Grad d, Durchmesser d, Expansion $\approx 1/\sqrt{d}$

d-dimensionales Butterfly

- Knoten: $(k,(x_d,\ldots,x_1)) \in \{0,..,d\} \times \{0,1\}^d$
- Kanten: $(k,(x_d,\ldots,x_1)) \rightarrow (k+1,(x_d,..,x_{k+1},..,x_1)),$
  $(k+1,(x_d,..,1\text{-}x_{k+1},..,x_1))$
- ungerichtetes Butterfly:



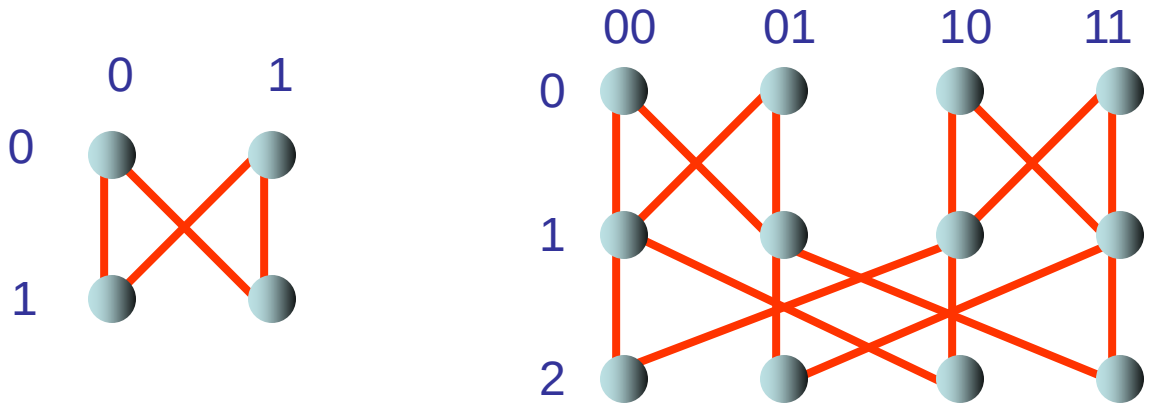
d=1



d-dim. BF

d-dimensionales Butterfly

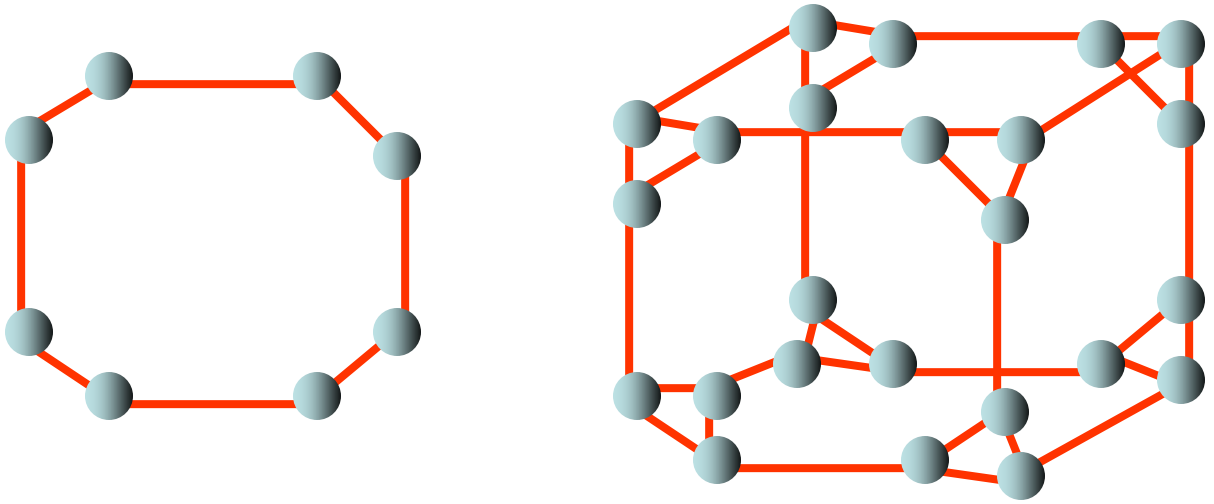
- Knoten: $(k,(x_d,\ldots,x_1)) \in \{0,..,d\} \times \{0,1\}^d$
- Kanten: $(k,(x_d,\ldots,x_1)) \rightarrow (k+1,(x_d,..,x_{k+1},..,x_1)),$
  $(k+1,(x_d,..,1\text{-}x_{k+1},..,x_1))$



Grad 4, Durchmesser 2d, Expansion ~1/d

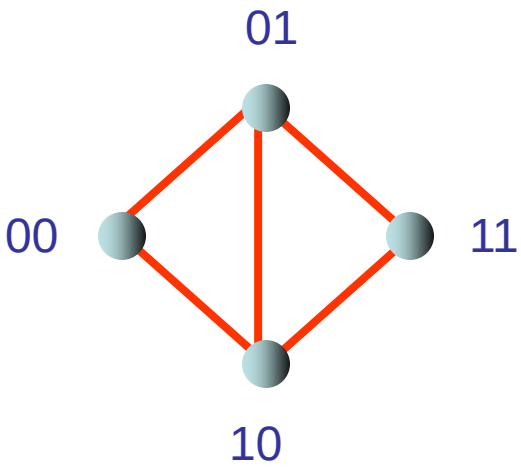
Cube-Connected-Cycles

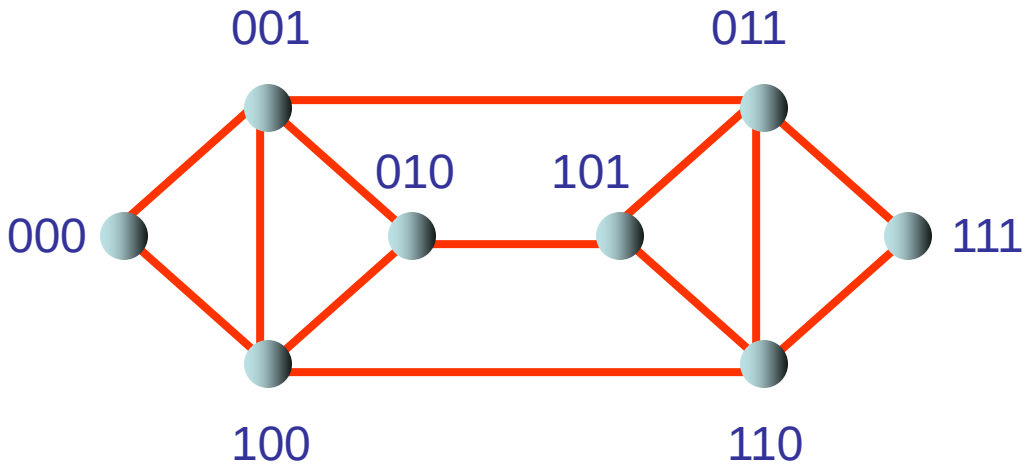
- Knoten: $(k,(x_1,\ldots,x_d)) \in \{0,..,d\text{-}1\} \times \{0,1\}^d$
- Kanten: $(k,(x_1,\ldots,x_d)) \rightarrow (k\text{-}1 \pmod d,(x_1,..,x_d))$,
  $(k+1 \pmod d,(x_1,..,x_d))$,
  $(k,(x_1,..,1\text{-}x_{k+1},..,x_d))$



d-dimensionaler De Bruijn Graph

- Knoten: $(x_1,\ldots,x_d) \in \{0,1\}^d$
- Kanten: $(x_1,\ldots,x_d) \rightarrow (0,x_1,x_2,\ldots,x_{d-1})$
  $(1,x_1,x_2,\ldots,x_{d-1})$
- ungerichteter de Bruijn Graph:





Durchmesser

Satz 2.7: Jeder Graph mit maximalem Grad $\delta \geq 4$ und Größe n muss einen Durchmesser von mindestens $(\log n)/(\log(\delta-1))-1$ haben.

Beweis: Übung

Satz 2.8: Für jedes gerade $\delta \geq 4$ gibt es eine Familie von Graphen mit maximalem Grad δ und Größe n mit Durchmesser höchstens $(\log n) / (\log \delta - 1)$.

Beweis: Übung

Expansion

Satz 2.9: Für jeden Graph G ist die Expansion $\alpha(G) \in [0,1]$.

Beweis: siehe Definition von $\alpha(G)$.

Definition 2.10: Ein Graph heißt Expander, wenn er eine konstante Expansion besitzt.

Satz 2.11: Es gibt Familien von Graphen mit konstantem Grad und konstanter Expansion.

Beispiel: Gabber-Galil Graph

- Knotenmenge: $(x,y) \in \{0,\ldots,k-1\}^2$
- $(x,y) \rightarrow (x,x+y),(x,x+y+1), (x+y,y), (x+y+1,y) \pmod k$

Noch bessere Expander bekannt als Ramanujan Graphen.

Skip Graphen

Betrachte eine beliebige Menge V an Knoten mit totaler Ordnung (d.h. die Knoten können bzgl. einer Ordnung ´<´ sortiert werden).

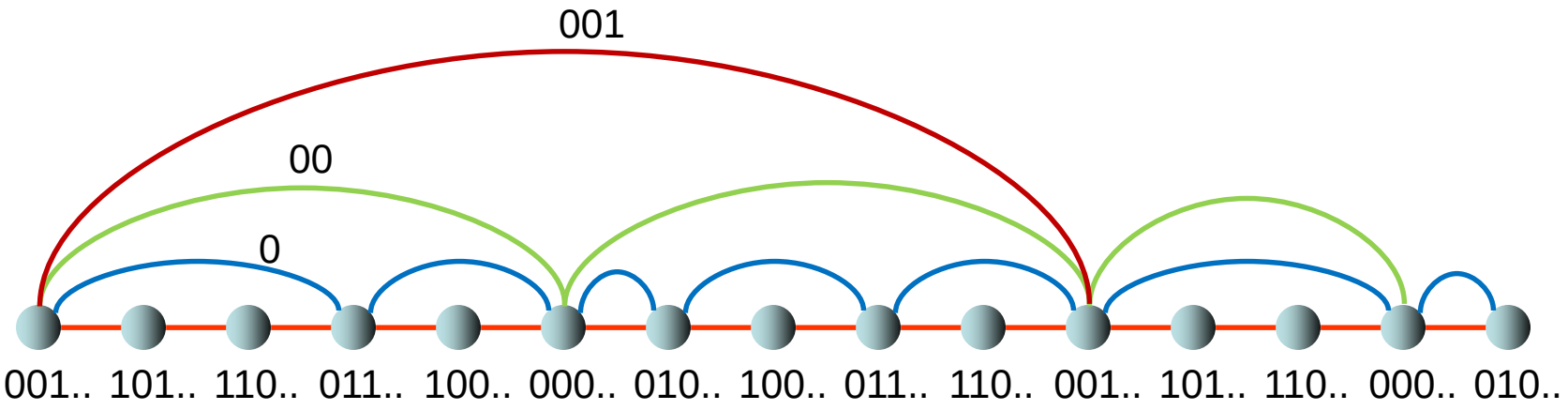
- Jeder Knoten v sei assoziiert mit einer genügend langen zufälligen Bitfolge $r(v)$ (so dass $r(v) \neq r(w)$ für alle $v,w \in V$).
- $prefix_i(v)$: erste i Bits von $r(v)$
- $succ_i(v)$: nächster Nachfolger w von v (bzgl. der Ordnung ´<´) mit $prefix_i(w)=prefix_i(v)$.
- $pred_i(v)$: nächster Vorgänger w von v mit $prefix_i(w)=prefix_i(v)$.

Skip Graph Regel:
Für jeden Knoten v und jedes $i \in \mathbb{N}_0$:

- v hat eine Kante zu $pred_i(v)$ und $succ_i(v)$

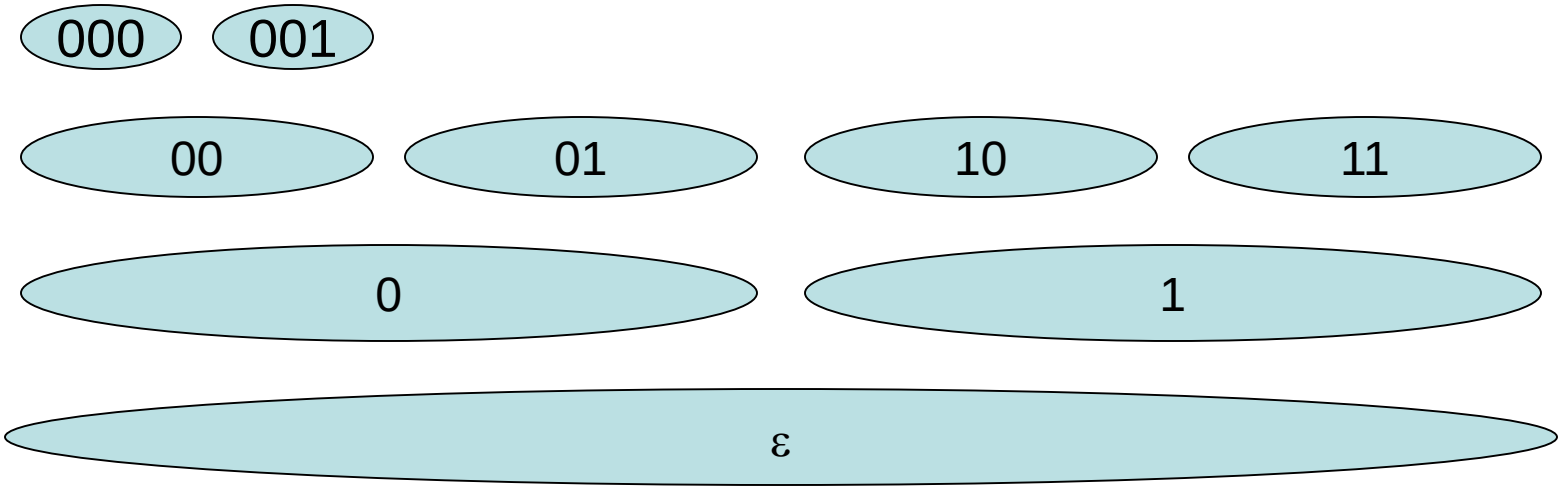
Skip Graphen

Beispiel einiger Teillisten für verschiedene Präfixlängen im Skip Graphen:



Skip Graphen

Hierarchische Sicht: geordnete Listen von Knoten mit demselben Präfix.

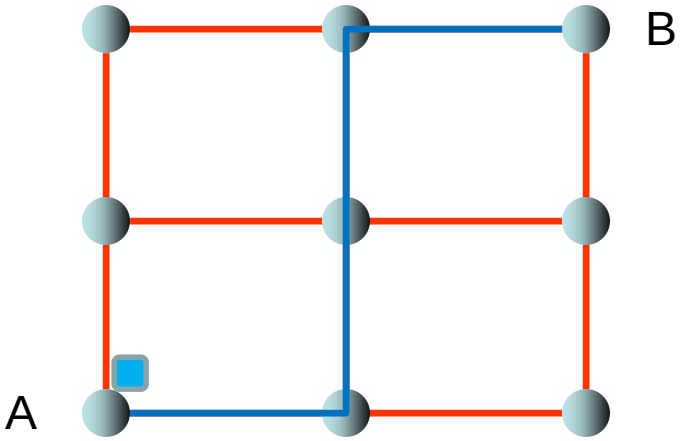


$\Theta(\log n)$ Grad, $\Theta(\log n)$ Durchmesser, $\Theta(1)$ Expansion (mit hoher Wahrscheinlichkeit)

Übersicht

- Grundlagen
- Grundlegende Graphparameter
- Klassische Graphfamilien
- Skip Graphen
- Routing

Routing



- Routing: finde Weg von A nach B

Routing

Ziel: Gegeben ein Graph $G=(V,E)$ und eine Menge $R=\{(s_1,t_1),\ldots,(s_k,t_k)\}\subseteq V\times V$ von Quell-Ziel-Paaren, finde einen Weg für jedes dieser Quell-Ziel-Paare, so dass

- die Weglänge so kurz wie möglich ist (um Nachrichten möglichst schnell auszuliefern) und
- möglichst wenige Wege über denselben Knoten wollen (um die Belastung der Knoten möglichst gering zu halten)

Wie finden wir solche Wege, und wie messen wir die Qualität solcher Wege?

Routing

Gängige Maße für die Qualität von Wegen:

Definition 2.12: Gegeben eine Menge an Wegen $P=\{p_1, p_2, p_3,\ldots\}$ mit Gewichten $w:P\rightarrow\mathbb{R}$ in einem Graphen $G=(V,E)$, dann ist

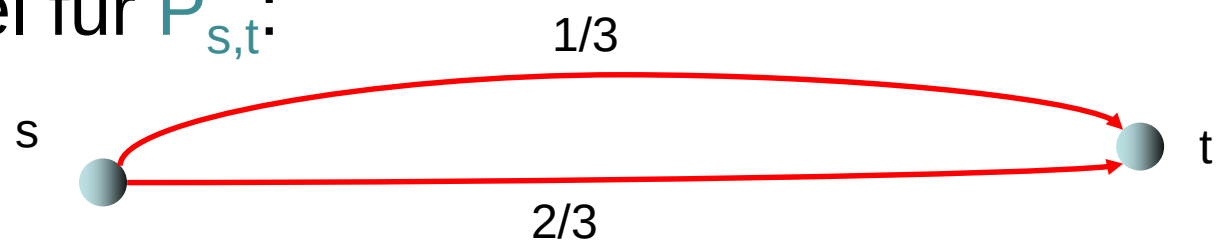
- Congestion von P (max. Knotenbelastung): $C(P) = \max_{v\in V} \Sigma_{p\in P_v} w(p)$, wobei $P_v\subseteq P$ die Menge aller Wege in P über v ist
- Dilation von P (max. Weglänge): $D(P) = \max_{p\in P} |p|$

Routing

Definition 2.13: Routingproblem: Gegeben ein Graph $G=(V,E)$ und eine Menge $R=\{(s_1,t_1),\ldots,(s_k,t_k)\}\subseteq V\times V$ von Quell-Ziel-Paaren, finde ein

- Wegesystem $P = \cup_{s,t} P_{s,t}$ mit nichtleerer Wegemenge $P_{s,t}$ für alle Quell-Ziel-Paare $(s,t)\in R$ und eine
- Gewichtsfunktion: $w:P\rightarrow[0,1]$,so dass für alle $(s,t)\in R$: $\Sigma_{p\in P_{s,t}} w(p) = 1$.

Beispiel für $P_{s,t}$:



Oblivious Routing

Einfachste Strategie zur Lösung von Routingproblemen: oblivious Routing (verwende vorberechnete Wege, um ein beliebiges Routingproblem zu lösen)

Definition 2.14: Sei $G=(V,E)$ ein Netzwerk. Ein oblivious Routing-schema (P,w) ist spezifiziert durch:

- Wegesystem: $P = \cup_{s,t} P_{s,t}$ mit nichtleerer Wegemenge $P_{s,t}$ für alle Quell-Ziel-Paare $(s,t) \in V^2$
- Gewichtsfunktion: $w: P \rightarrow \mathbb{R}_+$, so dass für alle $(s,t) \in V^2$: $\Sigma_{p \in P_{s,t}} w(p) = 1$.

Lösung eines Routingproblems mittels oblivious Routingschema (P,w):

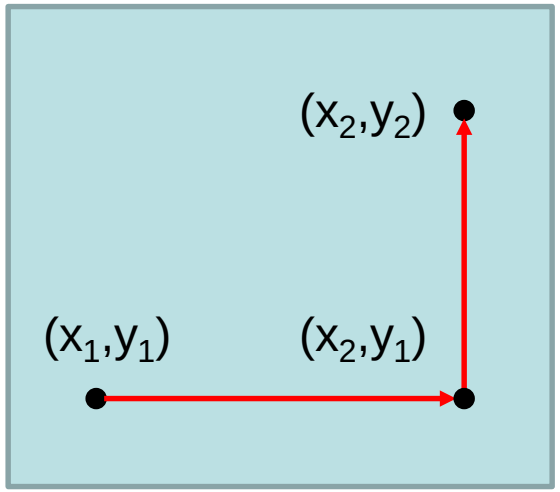
Gegeben eine Menge $R=\{(s_1,t_1),\ldots,(s_k,t_k)\}$, wähle für jedes $(s,t) \in R$ die Wegemenge $P_{s,t}$ aus P mit den vorgegebenen Gewichten in w.

Wie gut ist das?

Oblivious Routing im Gitter

Oblivious Routingschema P für $k \times k$-Gitter:

- Wegesystem P: Für jedes Paar $(x_1,y_1),(x_2,y_2) \in [k]^2$, route erst von (x_1,y_1) nach (x_2,y_1), dann von (x_2,y_1) nach (x_2,y_2).
- D.h. eindeutiger Weg p ($w(p)=1$) pro Quell-Ziel-Paar.



x-y-Routing Strategie

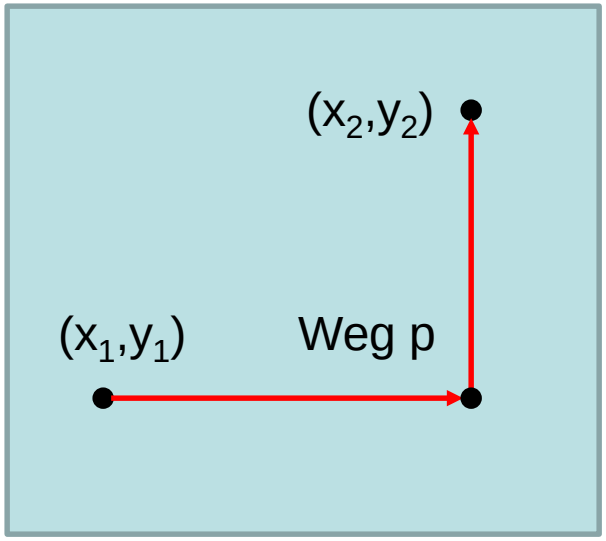
Oblivious Routing im Gitter

„Benchmark“ für Routingstrategie: kann die Strategie beliebige Permutationen $\pi:V\rightarrow V$ mit geringer Dilation und Congestion routen? (D.h. das Routingproblem R_π zu π ist definiert als $R_\pi=\{ (v,\pi(v)) \mid v\in V \}$).

Satz 2.15: Die x-y-Routingstrategie kann jede Permutation im $k\times k$-Gitter mit Congestion höchstens $4d$ und Dilation höchstens d routen, wobei d die maximale Distanz eines Quell-Ziel-Paares ist.

Oblivious Routing im Gitter

Beweis:

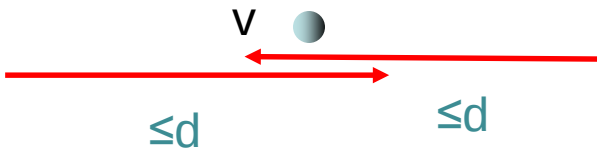


Dilation:

- p hat Länge $d((x_1,y_1),(x_2,y_2))$
- also ist max. Weglänge d

Congestion:

- betrachte Knoten v in x-Richtung



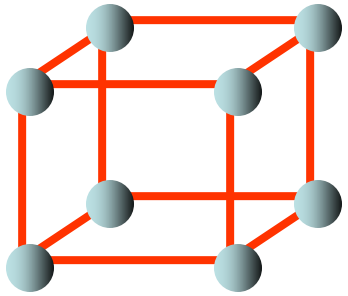
- maximal 2d Quellen haben Wege, die in x-Richtung über v laufen, also ist Congestion in x-Richtung $\leq 2d$
- dasselbe gilt auch für Ziele in y-Richtung

Oblivious Routing im Hypercube

Bitanpassungsstrategie:
Weg von $(x_1,\ldots,x_d)$ nach $(y_1,\ldots,y_d)$ führt über $(y_1,x_2,\ldots,x_d)$, $(y_1,y_2,x_3,\ldots,x_d)$, $(y_1,y_2,y_3,x_4,\ldots,x_d)$, …, $(y_1,y_2,y_3,\ldots,y_{d-1},x_d)$, $(y_1,\ldots,y_d)$

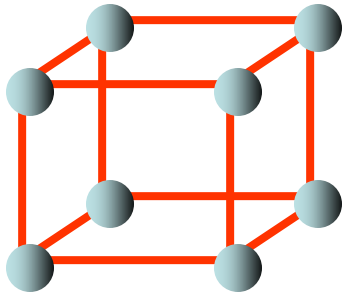
- Dilation: optimal, da Weglänge gleich Distanz
- Congestion: es gibt Permutationen, die sehr hohe Congestion haben!

Beispiel: sei $\pi(x_1,\ldots,x_d) = (x_d,\ldots,x_1)$
für alle $(x_1,\ldots,x_d)$



Oblivious Routing im Hypercube

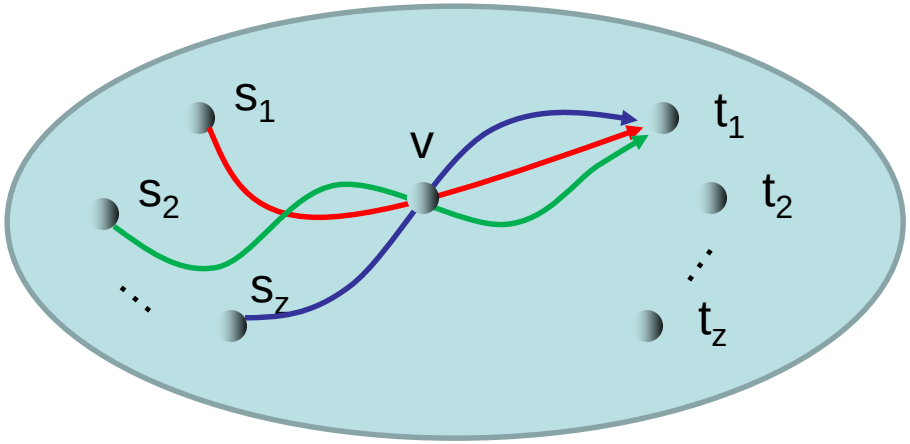
Beispiel: sei $\pi(x_1,\ldots,x_d) = (x_d,\ldots,x_1)$ für alle $(x_1,\ldots,x_d)$



- Betrachte Knotenmenge $M=\{ (x_1,\ldots,x_{d/2},0,\ldots,0) \mid x_i \in \{0,1\}$ für alle $i \in \{1,\ldots,d/2\} \}$
- Nach $d/2$ Routingschritten gemäß π sind alle Pakete mit Quelle in M im Knoten $(0,\ldots,0)$
- Da $|M|=2^{d/2} = \sqrt{2^d} = \sqrt{n}$ ist, kann Congestion wesentlich größer als Dilation (maximal $\log n$) sein. Bestmöglich wäre Congestion $O(\log n)$!

Borodin-Hopcroft Schranke

Satz 2.16: Für jeden ungerichteten Graphen G der Größe n mit Grad δ und jedes oblivious Routing-schema mit nur einem Pfad pro Quell-Ziel-Paar gibt es eine Permutation π, für die ein Knoten von mindestens $\sqrt{n/\delta}$ Pfaden durchlaufen wird.



Es gibt v und $t_1,\ldots,t_z$,so dass jedes t_i mind. z Quellen hat,deren Wege durch v führen, $z = \sqrt{n/\delta}$.

Valiants Trick

Frage: gibt es oblivious Routingschemen, die für alle Permutationen eine niedrige Congestion garantieren?

Antwort: ja, aber wir brauchen mehrere Pfade pro Quell-Ziel-Paar.

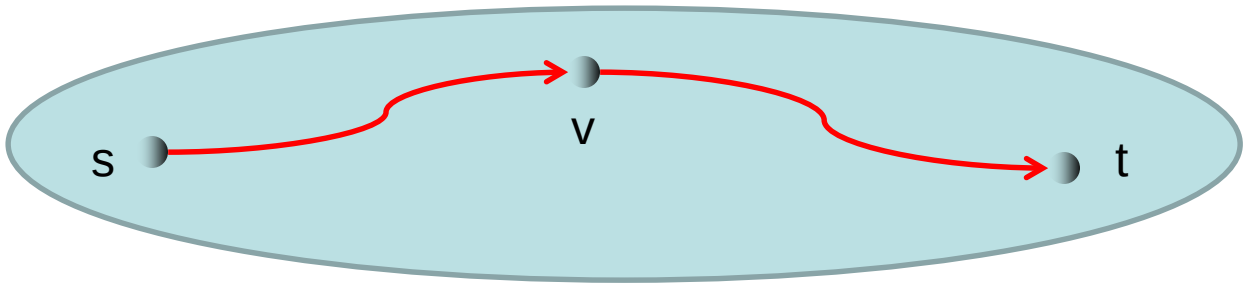
Strategie:

- Konstruiere zunächst ein Wegesystem $P' = \bigcup_{s,t} P'_{s,t}$ für alle $(s,t) \in V^2$ mit minimaler Congestion C_{OPT} (kann in polynomieller Zeit berechnet werden).
- Konstruiere aus P' mittels Zwischenzielen ein Wegesystem $P = \bigcup_{s,t} P_{s,t}$ für alle $(s,t) \in V^2$, so dass die Congestion in etwa gleich bleibt zu P'.

Valiants Trick

Konstruktion von P für ein bel. Netzwerk G=(V,E):

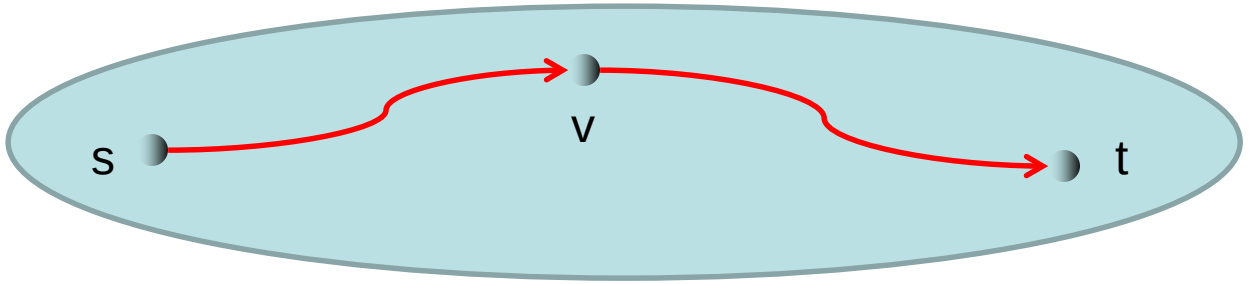
- $P' = \cup_{s,t} P'_{s,t}$: System mit minimaler Congestion C_{OPT}
- $P_{s,t} = \{ p=q_1 \circ q_2 \mid q_1 \in P'_{s,v}$ und $q_2 \in P'_{v,t}$ für ein $v \in V \}$
  ($\circ$: Konkatenation)
- Für alle $p=q_1 \circ q_2$ in $P_{s,t}$ setzen wir $w(p)=w(q_1) \cdot w(q_2)/n$
- Damit ist $\Sigma_{p \in P_{s,t}} w(p) = 1$, d.h. $P = \cup_{s,t} P_{s,t}$ ist ein gültiges Wegesystem
- Congestion von P: $2C_{OPT}$ (Beweis: Übung)



Valiants Trick

Eigenschaften von P:

- Betrachte eine beliebige Permutation π
- Betrachte nur die erste Hälfte aller Wege in P für R_π
- Für alle $(s,\pi(s)) \in R_\pi$ werden für die erste Hälfte alle Wege $q_1 \in P'_{s,v}$ für alle $v \in V$ mit Gewicht $w(q_1)/n$ verwendet.
- Insgesamt werden also alle Wege $q \in P'$ mit Gewicht $w(q)/n$ verwendet.
- Also ist die Congestion C_{OPT}/n.
- Das gleiche gilt für zweite Hälfte.



Valiants Trick

Satz 2.17: Mit Valiants Trick (d.h. einem festen oblivious Routingschema mit Zwischenzielen) kann in jedem Netzwerk jede Permutation mit Congestion höchstens $2C_{OPT}/n$ geroutet werden.

Satz 2.18: In jedem Netzwerk benötigt eine Permutation im Durchschnitt mindestens C_{OPT}/n Congestion, um geroutet zu werden.

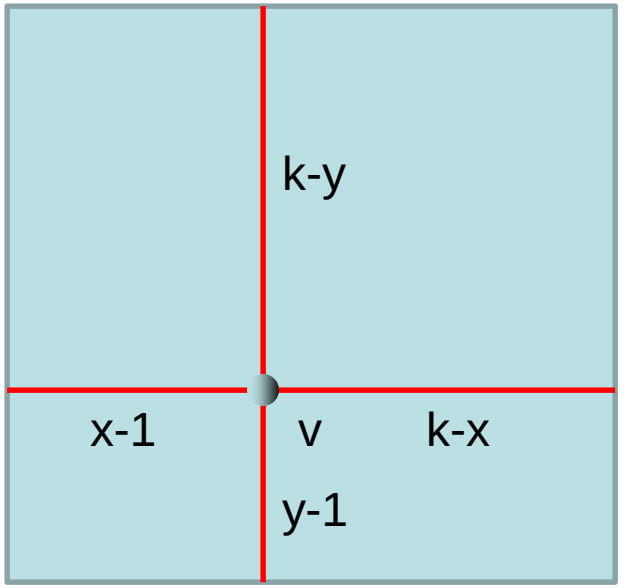
Beweis: Übung

Valiants Trick

Obere Schranke für C_{OPT} für k×k-Gitter:

Für das x-y-Routing gilt für jeden Knoten v an Position $(x,y) \in \{1,\ldots,k\}^2$:

- Es gibt maximal
  $(x-1)(k-x+1)k + k^2 + (k-x)x \cdot k$
  Quell-Ziel-Paare, deren Pfade v in x-Richtung durchlaufen
- Entsprechend gibt es maximal
  $(y-1)(k-y+1)k + k^2 + (k-y)y \cdot k$
  Quell-Ziel-Paare, deren Pfade v in y-Richtung durchlaufen
- Also ist $C_{OPT}=O(k^3)$ und damit $C_{OPT}/n = O(k)$.

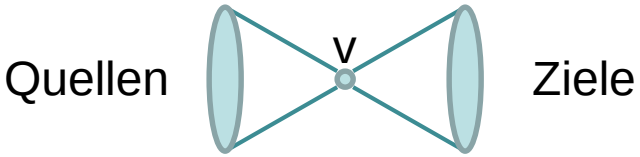


Valiants Trick

Obere Schranke für C_{OPT} für d-dim. Hypercube:

Für die Bitanpassungsstrategie gilt für jeden Knoten v:

- Für jedes $i \in \{0,\ldots,d\}$ hat v 2^i Quellen, die v in i Schritten erreichen können, und die von v aus noch 2^{d-i} Ziele erreichen können.



- Es können also maximal

$$\sum_{i=0}^{d} 2^i \cdot 2^{d-i} = (d+1)2^d$$

  Pfade über v laufen.
- Also ist $C_{OPT} \leq (d+1)2^d$ und damit $C_{OPT}/n = O(d)$.

Valiants Trick

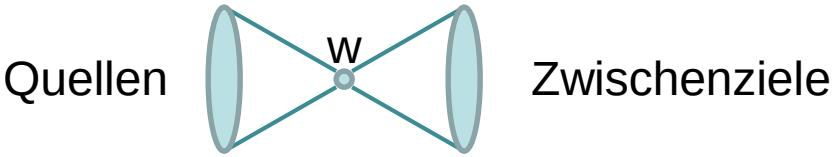
Valiants Trick für d-dimensionalen Hypercube:

- Bitanpassungsstrategie ist tatsächlich sogar optimales Wegesystem für P'.
- D.h. $P'_{s,t}$ besteht lediglich aus einem Weg für jedes Knotenpaar (s,t) im Hypercube, d.h. $P'_{s,t} = \{p_{s,t}\}$ für den Bitanpassungsweg $p_{s,t}$ und $w(p_{s,t})=1$.
- Also ist $P_{s,t} = \{ p_{s,v,t} = p_{s,v} \circ p_{v,t} \mid v \in V \}$ mit $w(p_{s,v,t})= 1/n$.

Valiants Trick

Ist das Wegesystem wirklich gut?

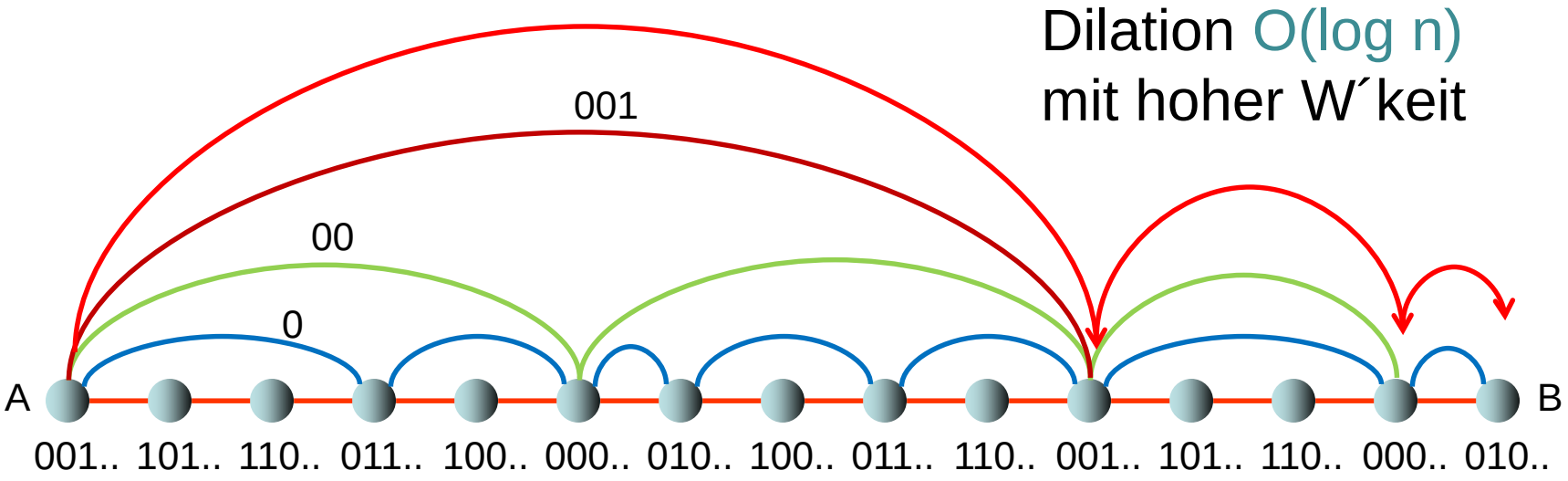
- Betrachte eine beliebige Permutation π (d.h. wir haben Quell-Ziel-Paare $(v, \pi(v))$ für alle $v \in V$).
- Zunächst streuen sich die Wege von s über alle n Zwischenziele v
- Dilation: maximal d bis v, also insgesamt maximal $2d$.
- Congestion: betrachte einen festen Knoten w und für ein festes $i \in \{0,\ldots,d\}$ alle Quell-Zwischenziel-Paare, deren Wege w in i Schritten erreichen.
- Mit der Bitanpassungsstrategie gibt es 2^i mögliche Quellen für w.
- Weiterhin können 2^{d-i} Zwischenziele von w aus erreicht werden.



- D.h. die Anzahl der Quell-Ziel-Paare, dessen Pfade w durchlaufen, ist max.
$$\Sigma_{i=0}^{d} 2^i \cdot 2^{d-i} = (d+1)\cdot 2^d = (d+1)\cdot n.$$
- Da jeder dieser Pfade ein Gewicht von $1/n$ hat, ist die Congestion maximal
$$(d+1)\cdot n \cdot 1/n = d+1.$$
- Da diese Congestion zweimal entsteht (von der Quelle zum Zwischenziel, und vom Zwischenziel zum Ziel), ist die Gesamtcongestion $\leq 2(d+1)$.

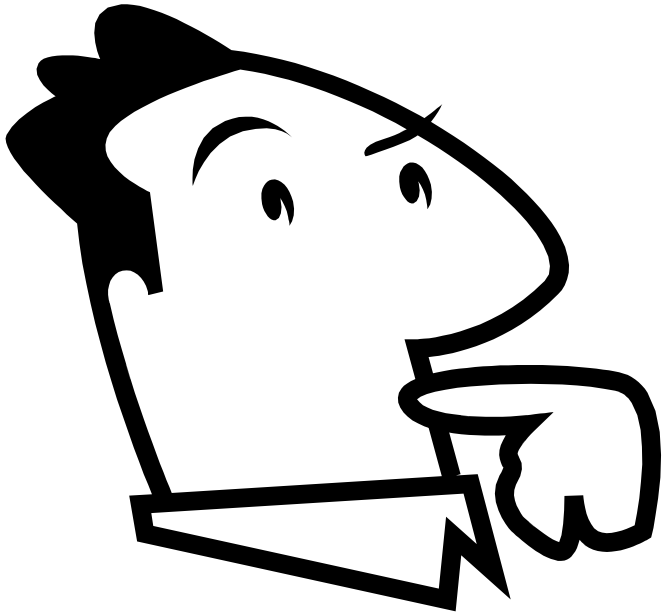
Oblivious Routing im Skip Graph

Geeignete Routingstrategie (A kennt ID von B): wähle möglichst lange Kante in der Richtung des Ziels B (Greedy Routing)



Referenzen

- James Aspnes, Udi Wieder: The expansion and mixing time of skip graphs with applications. *Distributed Computing* 21(6): 385-393 (2009).
- O. Gabber and Z. Galil. Explicit constructions of linear-sized superconcentrators. *Journal of Computer and System Sciences, 22:407–420, 1981.*
- F. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes. Morgan* Kaufmann Publishers, 1992.
- A. Lubotzky, R. Phillips, and R. Sarnak. Ramanujan graphs. *Combinatorica, 8(3):261–277, 1988.*
- C. Scheideler. *Universal Routing Strategies for Interconnection Networks.* Lecture Notes in Computer Science 1390. Springer, 1998.



Fragen?