Informatik III

CoNe
Freiburg

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Arne Vater

Wintersemester 2006/07
25. Vorlesung
01.02.2007

Approximation

- **Viele wichtige Probleme sind NP-vollständig**
  - (also nicht effizient lösbar unter der Annahme P $\neq$ NP)
- **Diese sind zu wichtig um sie zu ignorieren**
- **Mögliche Lösungen:**
  - Für kleine n ist exponentielle Laufzeit OK
  - Spezialfälle vielleicht in polynomieller Zeit lösbar
  - Vielleicht tritt worst-case Laufzeit extrem selten auf
  - Möglicherweise kann eine beweisbar gute Näherungslösung in polynomieller Zeit berechnet werden

Konzepte und Terminologie (1)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Wir betrachten Optimierungsprobleme**
- **Problem X hat viele Lösungen**
- **Wir suchen Lösung S für X, die eine Kostenfunktion c(S) minimiert oder maximiert.**
  - Beispiele für Graphen:
    - finde einen minimalen Spannbaum eines Graphen
    - finde einen minimalen Hamiltonkreis
- **Seien C, C* Kosten der approximierten bzw. optimalen Lösung.**
- **Ein Approximationsalgorithmus hat Approximationsgüte ρ(n),**
  - falls für jede Eingabegröße n

$$\max\left(\frac{C}{C^*}, \frac{C^*}{C}\right) \leq \rho(n)$$

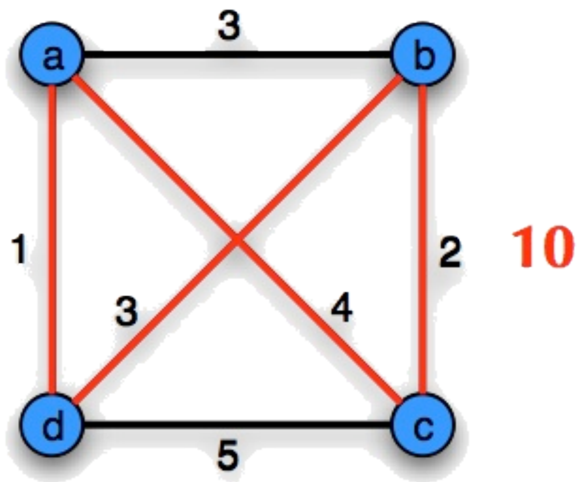
Konzepte und Terminologie (2)

- **Approximationsalgorithmus mit Güte ρ(n) wird als ρ(n)-Approximations-algorithmus bezeichnet**
- **Approximationsschema**
  – Approximationsalgorithmus mit zusätzlichem Parameter $\varepsilon > 0$ der eine $(1+\varepsilon)$-Approximation liefert
  – Falls Laufzeit polynomiell in n für jedes feste ε, spricht man von einem ***polynomiellen Approximationsschema*** (polynomial time approximation scheme, ***PTAS***)
  – Falls Laufzeit polynomiell in n und ε (z.B. $(1+\varepsilon)^2 n^2$), spricht man von einem ***streng polynomiellen Approximationsschema***

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Traveling Salesman Problem (TSP)

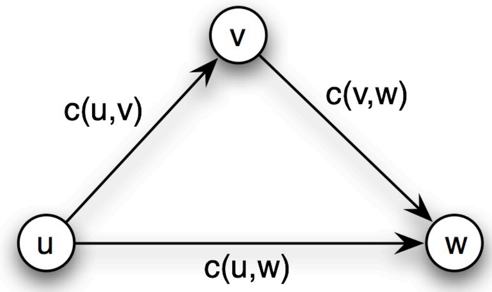
- **Gegeben:**
  - vollständiger Graph G = (V,E)
  - Kostenfunktion c(u,v) für alle (u,v) $\in$ E
- **Gesucht:**
  - Hamiltonkreis (Tour) mit minimalen Kosten



- **Theorem:**
  - Falls P $\neq$ NP existiert kein polynomieller Approximationsalgorithmus für TSP mit konstanter Güte
- **Beweis:**
  - Annahme es existiert ein Algorithmus, der TSP in pol. Zeit löst
  - Man kann zeigen: Hamiltonkreis $\leq_{m,p}$ TSP
  - Wir wissen: Hamiltonkreis ist NP-vollständig
  - Also kann A nicht existieren, falls P $\neq$ NP

Traveling Salesman Problem

- **TSP mit Einschränkung: Δ-TSP**
- **Gegeben:**
  - vollständiger Graph G = (V,E)
  - Kostenfunktion c(u,v) für alle (u,v) ∈ E



$$\forall u, v, w \in V \ : \ c(u,w) \leq c(u,v) + c(v,w)$$

- **Gesucht:**
  - Hamiltonkreis (Tour) mit minimalen Kosten

- **Beschränkung der Gewichte durch Dreiecksungleichung ist „natürlich“:**
  - ist in vielen Anwendungsfällen automatisch erfüllt
  - z.B. wenn Gewichte Entfernungen im euklidischen Raum repräsentieren

- **Aber: Auch Δ-TSP ist NP-schwierig!**
  - wird hier nicht bewiesen

Approximation für Δ-TSP

- **Lösungsansatz:**
  - Gibt es ein ähnliches/verwandtes Problem?
  - Ist es einfacher zu berechnen?

- **Minimale Spannbäume**
  - MST: Minimal Spanning Tree
  - Aber: wie kann ein MST in eine kürzeste Tour umgeformt werden?

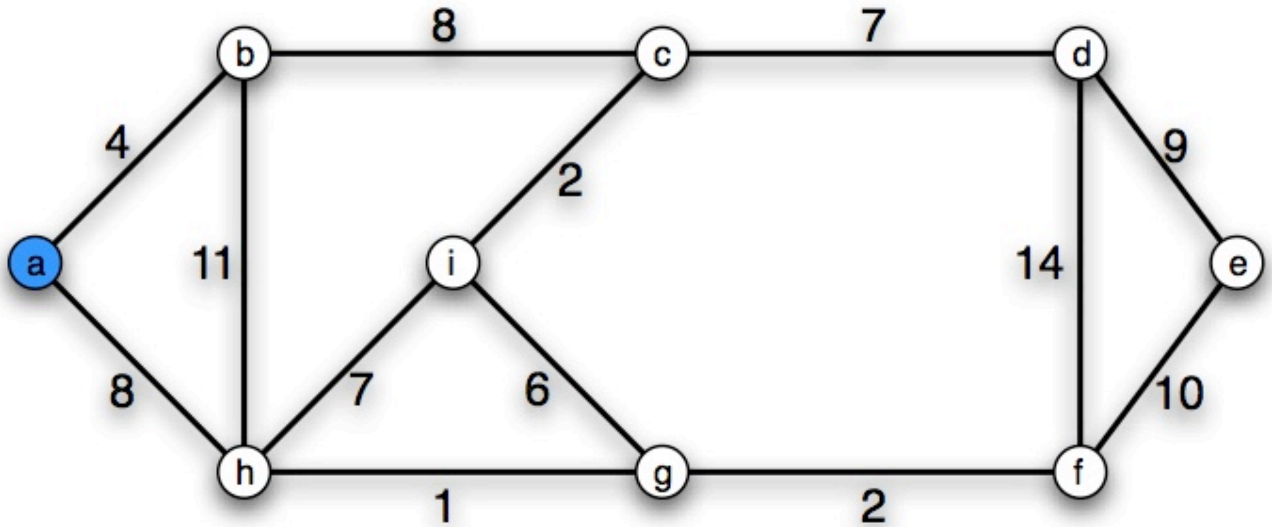
- **Idee:**
  - Bei Tiefensuche im MST wird jede Kante zweimal traversiert
  - Besuche alle Knoten in der Reihenfolge eines Pre-Order Tree-Walks

Prims Algorithmus zur Berechung des MST

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

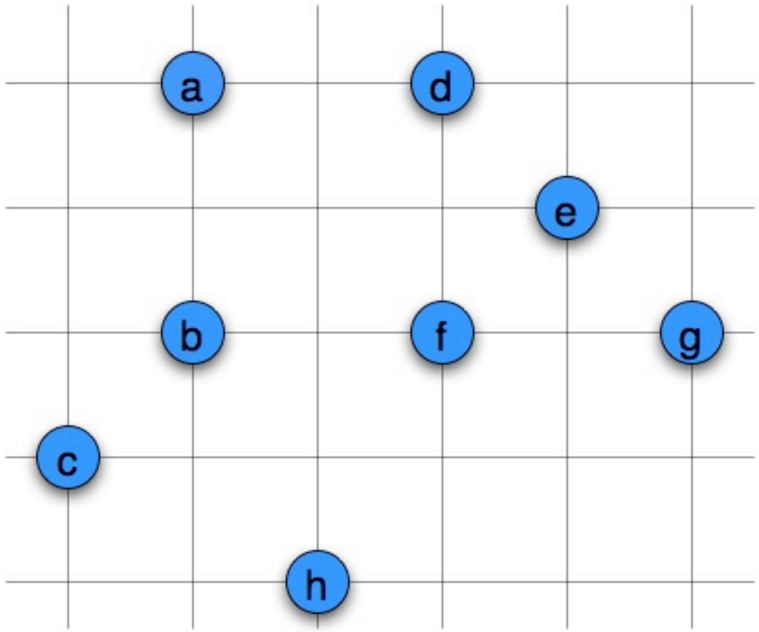
MST-PRIM(G)

1. **Initialisiere Baum B mit beliebigem Knoten**
2. **Wiederhole bis B alle Knoten enthält oder nicht erweitert werden kann**
   - Erweitere B mit Kante mit geringstem Gewicht, die B mit dem Rest von G verbindet

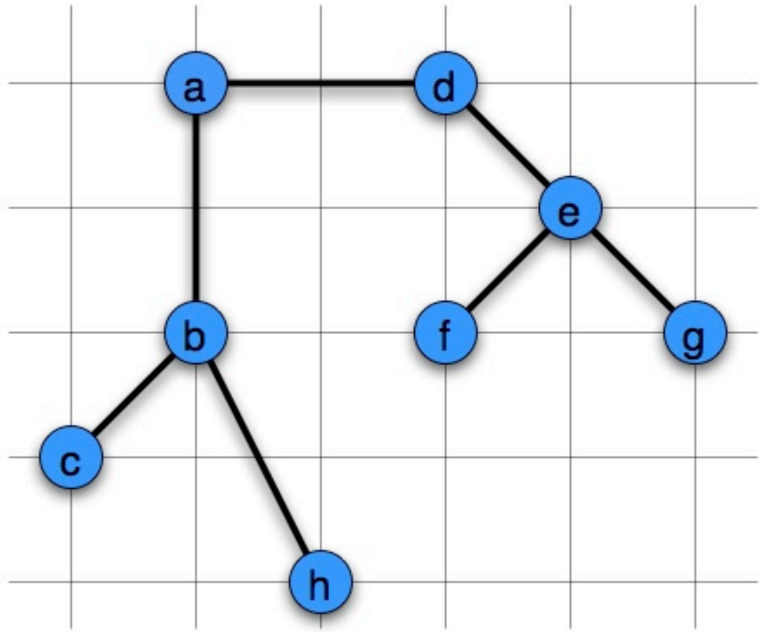


APPROX-Δ-TSP(G, c)
1 wähle Knoten $v \in V$
2 berechne minimalen Spannbaum T für G mit Wurzel v
3 konstruiere Liste L der Knoten die durch pre-order tree-walk in T entsteht
4 Gebe den durch L definierten Hamiltonkreis aus

Graph G
(gew. gemäß euklidischer Distanz)



Spannbaum T mit Wurzel A



Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Approximation für Δ-TSP

APPROX-Δ-TSP(G, c)

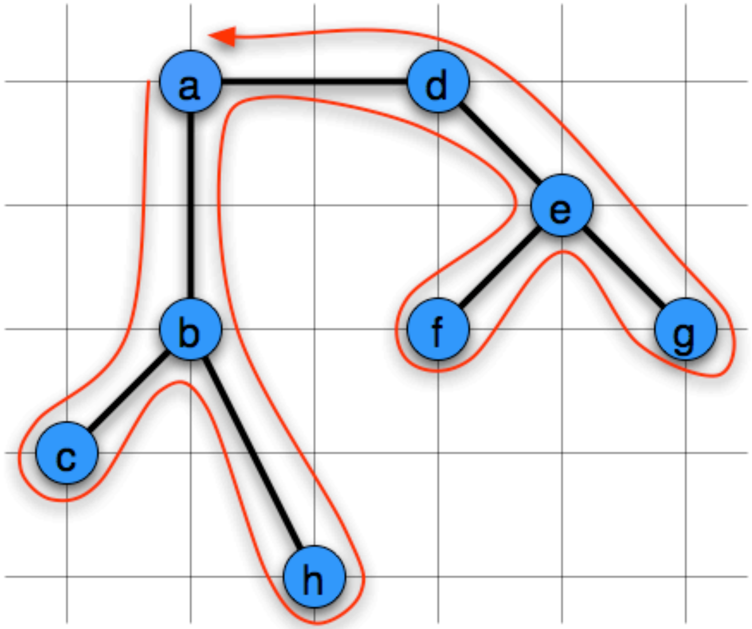
1 wähle Knoten $v \in V$

2 berechne minimalen Spannbaum T für G mit Wurzel v

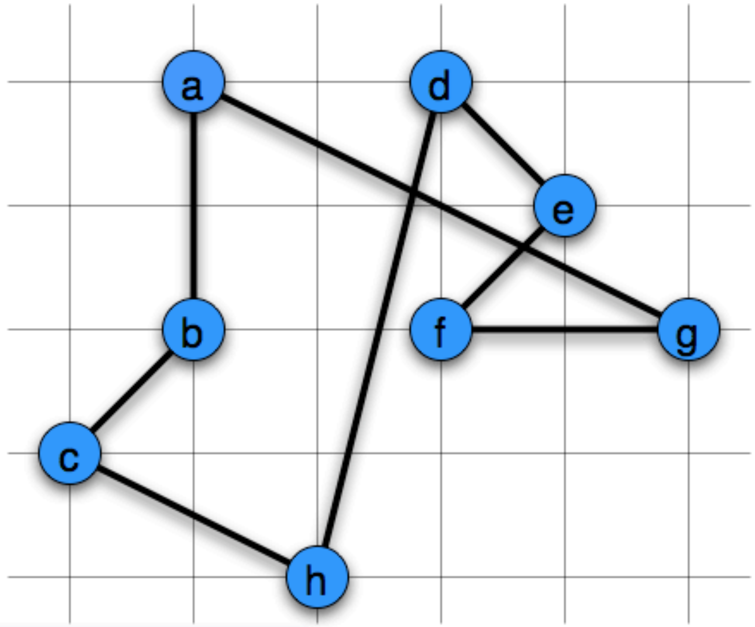
3 konstruiere Liste L der Knoten die durch pre-order tree-walk in T entsteht

4 Gebe den durch L definierten Hamiltonkreis aus

pre-order tree-walk:
(a, b, c, h, d, e, f, g)



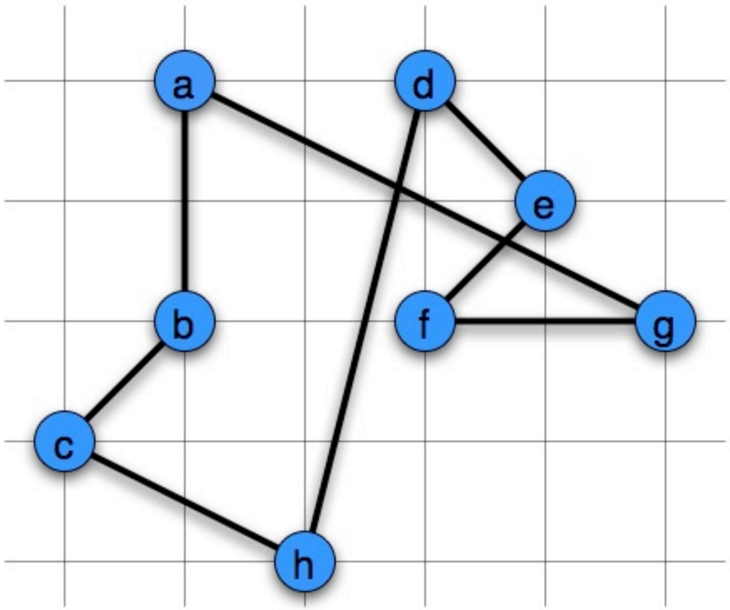
durch *L* definierter Hamiltonkreis:



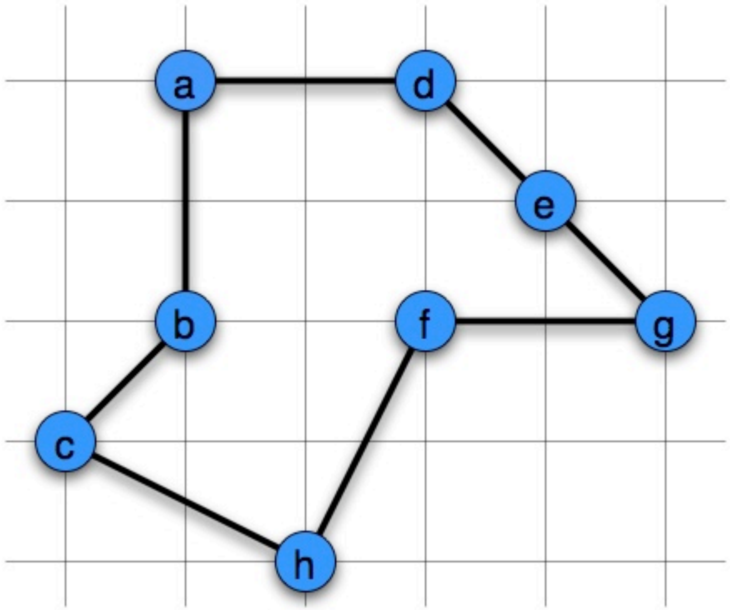
Approximation für Δ-TSP

APPROX-Δ-TSP(G, c)
1 wähle Knoten $v \in V$
2 berechne minimalen Spannbaum T für G mit Wurzel v
3 konstruiere Liste L der Knoten die durch pre-order tree-walk in T entsteht
4 Gebe den durch L definierten Hamiltonkreis aus

durch L definierter Hamiltonkreis:



minimaler Hamiltonkreis:



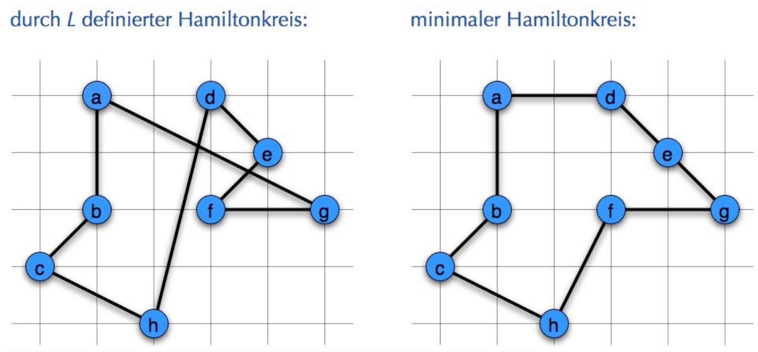
Approximation für Δ-TSP

APPROX-Δ-TSP(G, c)
1 wähle Knoten $v \in V$
2 berechne minimalen Spannbaum T für G mit Wurzel v
3 konstruiere Liste L der Knoten die durch pre-order tree-walk in T entsteht
4 Gebe den durch L definierten Hamiltonkreis aus

- **Theorem:**
  - Approx-Δ-TSP hat Laufzeit $\Theta(E) = \Theta(|V|^2)$
- **Theorem:**
  - Approx-Δ-TSP ist Approximationsalgorithmus für Δ-TSP mit Güte 2
- **Beweis:**
  - Sei H Ergebnis von Approx-Δ-TSP und H* optimale Lösung
  - Seien Kosten einer Tour A definiert durch:

$$c(A) = \sum_{(u,v) \in A} c(u,v)$$

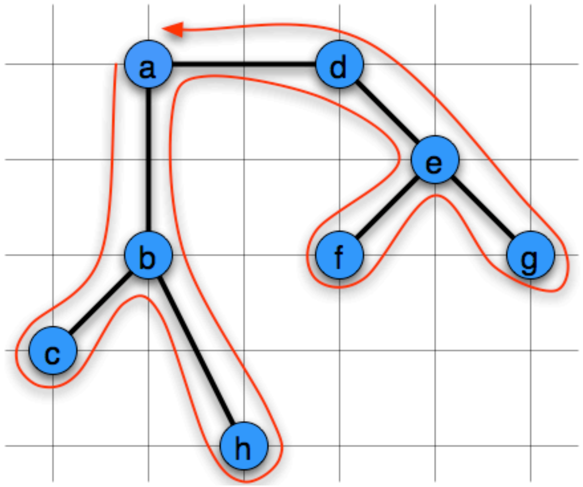
  - Theorem besagt: $c(H) \leq 2\, c(H^*)$



Approximation für Δ-TSP

- **Theorem:**
  - Approx-Δ-TSP ist Approximationsalgorithmus für Δ-TSP mit Güte 2
- **Beweis (Fortsetzung):**
  - Sei T der erzeugte Spannbaum. Es gilt: c(T) ≤ c(H*)
    - Streichen einer Kante in H* liefert Spannbaum mit Kosten ≤ c(H*)
  - Betrachte Folge F der Kanten die beim pre-order walk besucht werden:
    - z.B. F= (a,b,c,b,h,b,a,d,e,f,e,g,e,a)
    - F besucht jede Kante zweimal, somit gilt:
      - c(F) = 2c(T)
  - Daraus folgt c(F) ≤ 2c(H*)
  - H ensteht durch Streichen von Knoten in F. Wegen Dreiecksungleichung folgt
    - c(H) ≤ c(F)
  - Nun folgt:
    - c(H) ≤ 2 c(H*)

pre-order tree-walk:
(a, b, c, h, d, e, f, g)



Anmerkungen zum TSP

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Der angegebene Algorithmus kann leicht verbessert werden**

- **Algorithmus von Christofides liefert Güte 3/2**

- **Weiterhin hat Sanjeev Arora 1996 ein streng polynomielles Approximations-Schema vorgestellt:**
  - Eingabe:
    - eine Instanz des Δ-TSP im d-dimensionalen euklidischen Raum
    - und $\varepsilon > 0$
  - Erzeugt in Zeit $n^{O(d/\varepsilon)}$ eine Rundreise mit Güte $1 + \varepsilon$

Vertex Cover Problem

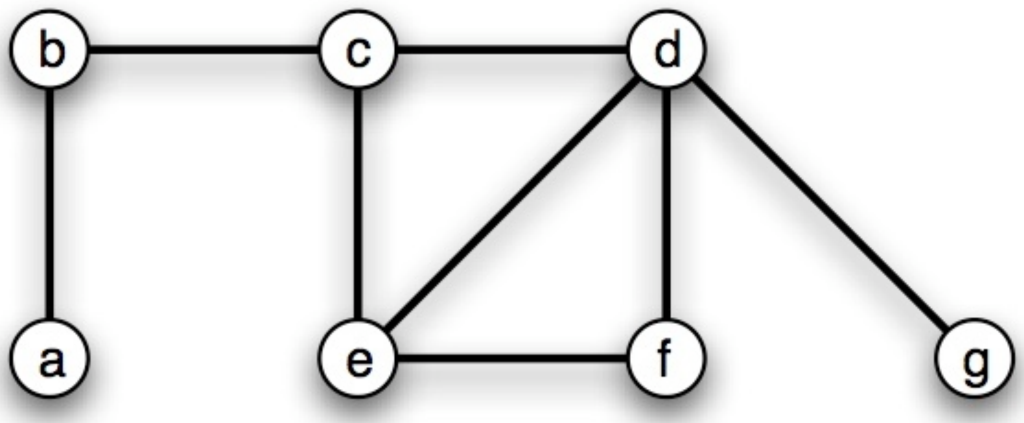
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Szenario:**
  - Alte Netzwerk-Router (Knoten) sollen gegen neue ausgetauscht werden, die Netzwerkverbindungen (Kanten) überwachen können.
  - Zum Überwachen einer Verbindung genügt es, wenn ein Router adjazent ist. Wieviele neue Router werden mindestens benötigt?
- **Formal:**
  - Gegeben $G = (V, E)$.
  - Finde minimale Teilmenge V' so dass
    - für alle $(u,v) \in E$ gilt:
      $u \in V$ oder $v \in V$.

Approximation für Vertex Cover

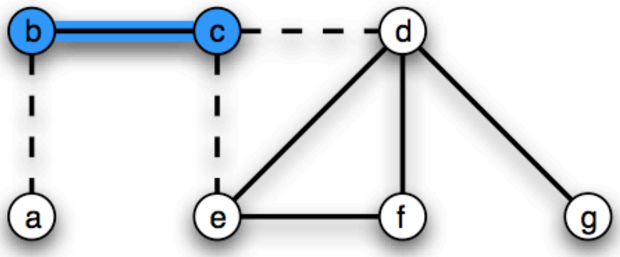
ApproxVertexCover(G(V,E))

1. C $\leftarrow \emptyset$
2. E' $\leftarrow$ E
3. solange E' $\neq \emptyset$
4. wähle $\{u,v\} \in$ E' zufällig
5. C $\leftarrow C \cup \{u,v\}$
6. entferne zu u oder v inzidente Kanten aus E'
7. gib C aus

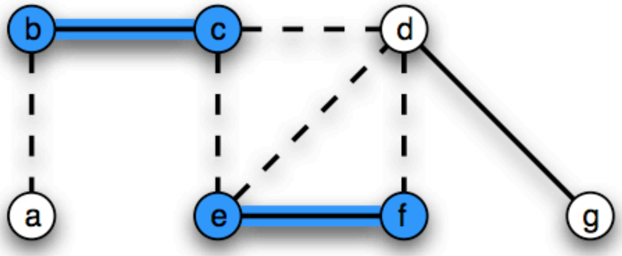


Beispiel

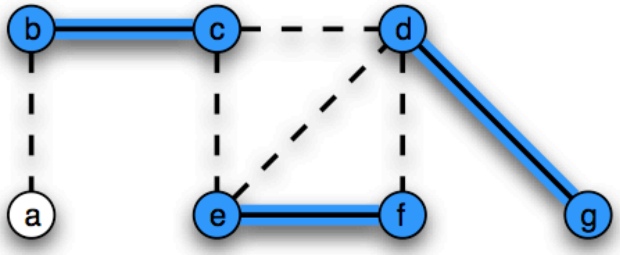
```
APPROXVERTEXCOVER(G(V,E))
C ← ∅
E' ← E
so lange E' ≠ ∅
    wähle {u,v} ∈ E' zufällig
    C ← C ∪ {u,v}
    entferne zu u,v inzidente Kanten aus E'
gebe C aus
```



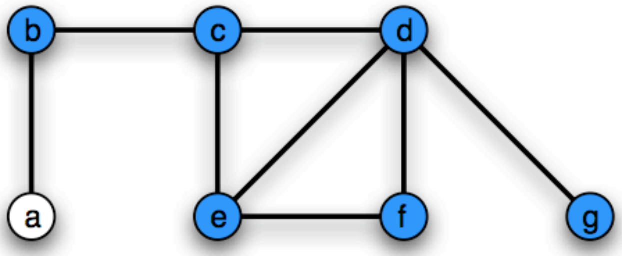
(1)



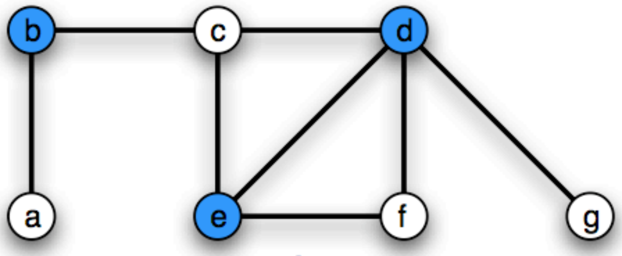
(2)



(3)



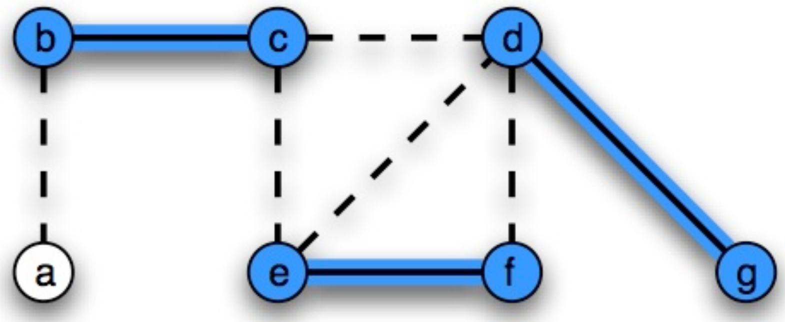
(4)



(minimale Lösung)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

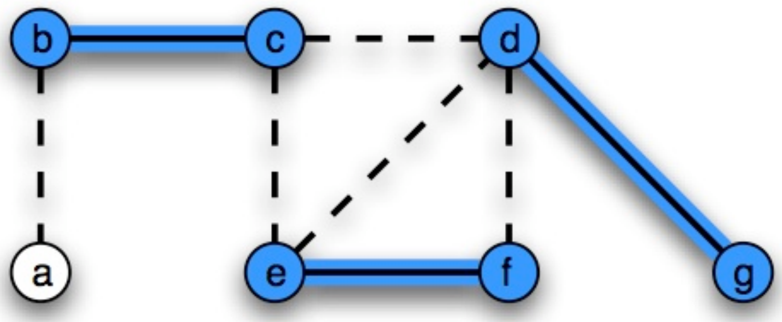
Analyse: ApproxVertexCover (1)



- **Theorem:**
  - ApproxVertexCover hat Güte 2 und Laufzeit O(E)
- **Beweis:**
  - Korrektheit, d.h. Lösung C ist Vertex Cover
    - Algorithmus läuft bis jede Kante in E zu Knoten in C inzident ist
  - Güte 2
    - Sei A Menge der in Zeile 4 gewählten Kanten (blau)
    - Keine 2 Kanten in A teilen einen Endpunkt
      - (sobald Kante gewählt, werden alle inzidenten Kanten entfernt)
    - Jede Iteration fügt 2 neue Knoten zu C hinzu, $|C| = 2\,|A|$
    - Minimales Vertex Cover C* muss wenigstens einen Knoten jeder Kante in A enthalten
    - Da keine Kanten in A Endpunkte teilen: $|A| \leq |C^*|$
    - somit gilt $|C| \leq 2\,|C^*|$

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Analyse: ApproxVertexCover (2)



ApproxVertexCover(G(V,E))

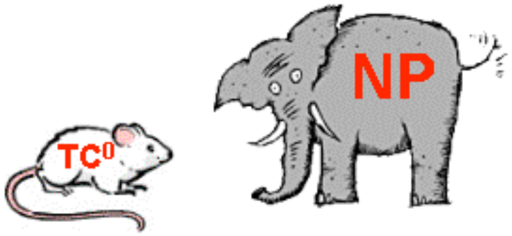
1. **C $\leftarrow \emptyset$**
2. **E' $\leftarrow$ E**
3. **solange E' $\neq \emptyset$**
4. **wähle {u,v} $\in$ E' zufällig**
5. **C $\leftarrow$ C $\cup$ {u,v}**
6. **entferne zu u oder v inzidente Kanten aus E'**
7. **gib C aus**

- **Theorem:**
  - ApproxVertexCover hat Güte 2 und Laufzeit O(E)
- **Beweis:**
  - Laufzeit O(E)
    - In jeder Iteration wird eine Kante aus E' entfernt
    - bei geeigneter Datenstruktur für E': Laufzeit O(E)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Komplexitätstheorie - Platzklassen

- **Platzkomplexität**
  - Definition
  - Simulation mehrerer Bänder
  - Savitchs Theorem
- **PSPACE**
  - PSPACE-schwierig
  - Das quantifizierte Boolesche Erfüllbarkeitsproblem
  - Gewinnstrategien für Spiele
- **Chomsky-Hierachien**
  - Lineare platzbeschränkte TM
  - Das Wortproblem linear platzbeschränkter TMs



What's your problem?
(aus: http://qwiki.caltech.edu/wiki/Complexity_Zoo)

Ein Platzkomplexitätsmaß

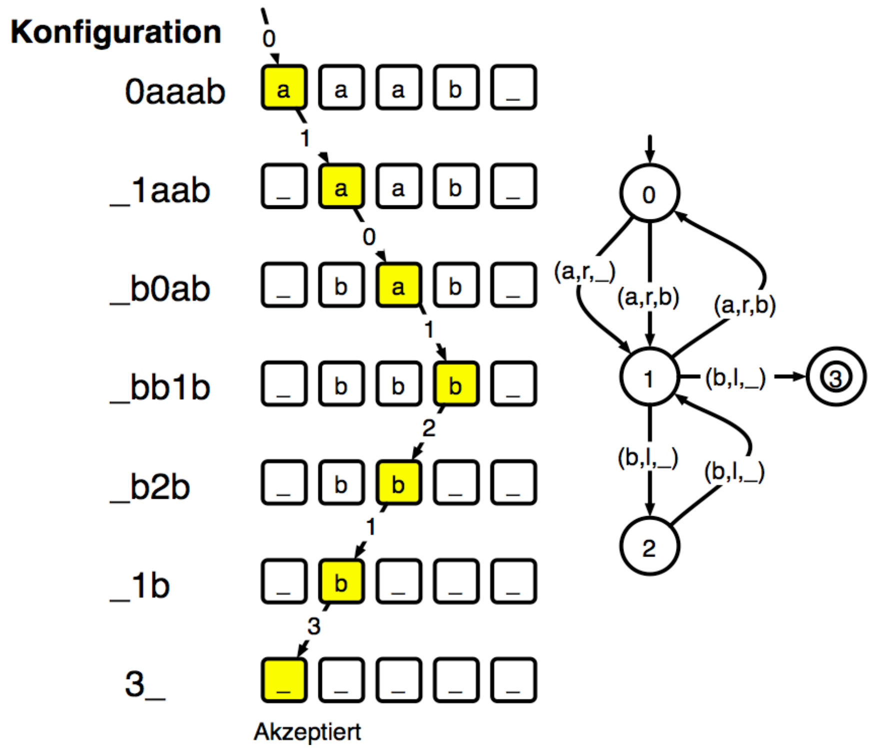
- **Definition**
  - Sei M eine deterministische Turing-Maschine, die auf allen Eingaben hält.
  - Der (Speicher-) **Platzbedarf (Platzkomplexität)** von M ist eine Funktion $f: N \rightarrow N$,
    - wobei f(n) die maximale Anzahl von Bandzellen von M ist, die M verwendet (ein Kopf der TM zeigt auf eine Bandzelle)
  - Falls f(n) der Platzverbrauch einer TM M ist, nennt man M auch eine **f(n)-Platz-Turing-Maschine**
    - z.B. Linear-Platz-TM für $f(n) = c\, n$ für eine Konstante c
    - z.B. Polynom-Platz-TM für $f(n) = c\, n^k$ für Konstanten c und k

- Zumeist beschreibt n die Eingabelänge.

Konfiguration einer DTM/NTM

- **Momentaufnahme einer TM**
  - –Bei Bandinschrift uv
    - dabei beginnt u am linken Ende des Bandes und
    - hinter v stehen nur Blanks,
  - –Zustand q,
  - –Kopf auf erstem Zeichen von v
- **Konfiguration C = uqv**

- **Bei Mehrband-TMs entsprechende Beschreibung aller Bänder**



Akzeptanz einer NTM

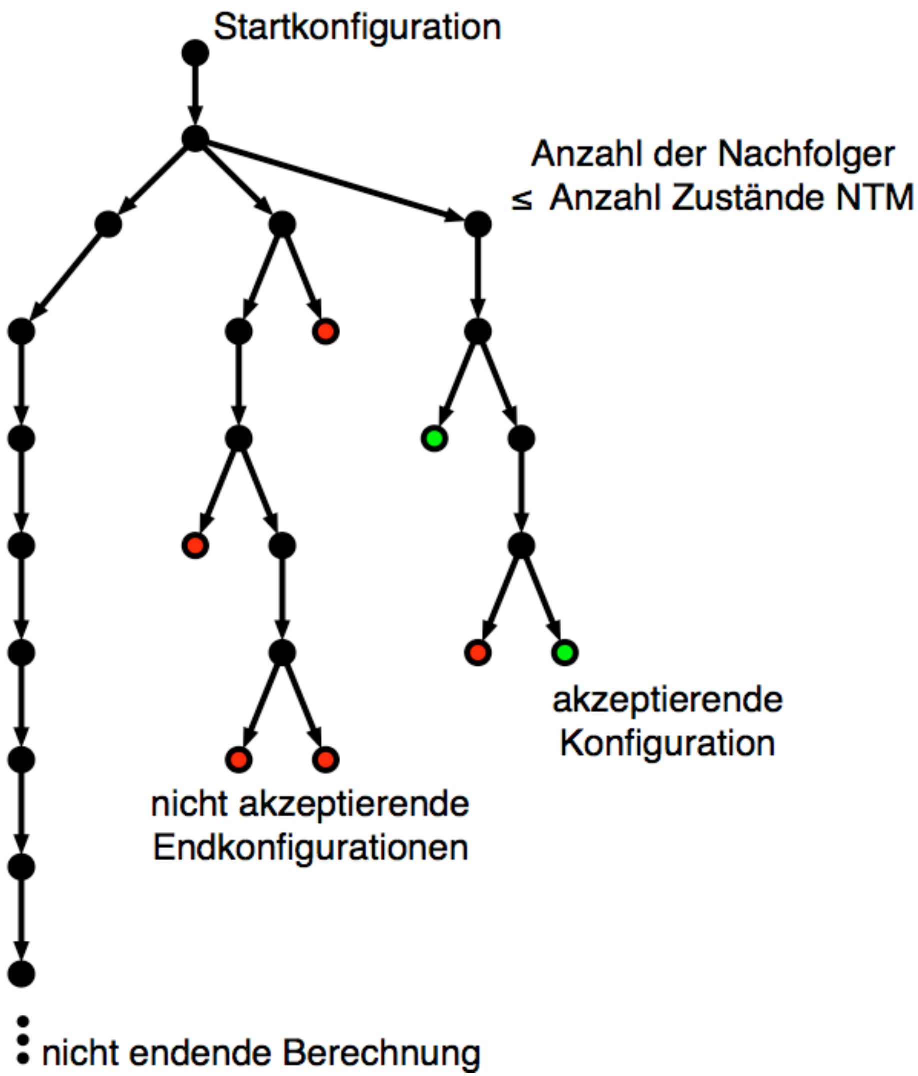
Eine Nichtdeterministische Turingmaschine M **akzeptiert** eine Eingabe w, falls es eine Folge von Konfigurationen C_1, C_2, …, C_k gibt, so dass

1. C_1 ist die Startkonfiguration von M bei Eingabe w
2. $C_{\mathbf{i}}$ kann zu C_{i+1} überführt werden
3. C_k ist eine akzeptierende Konfiguration

- Die von M akzeptierten Worte bilden die **von M akzeptierte Sprache** L(M)
- Eine NTM **entscheidet** eine Sprache, wenn jede Eingabe zu einem endlichen Berechnungsbaum der Konfigurationen führt.
- Der **Platzbedarf** s(n) einer Entscheider-NTM ist der maximale Platzbedarf über alle Pfade im Berechnungsbaums der NTM für alle Eingaben der Länge n.

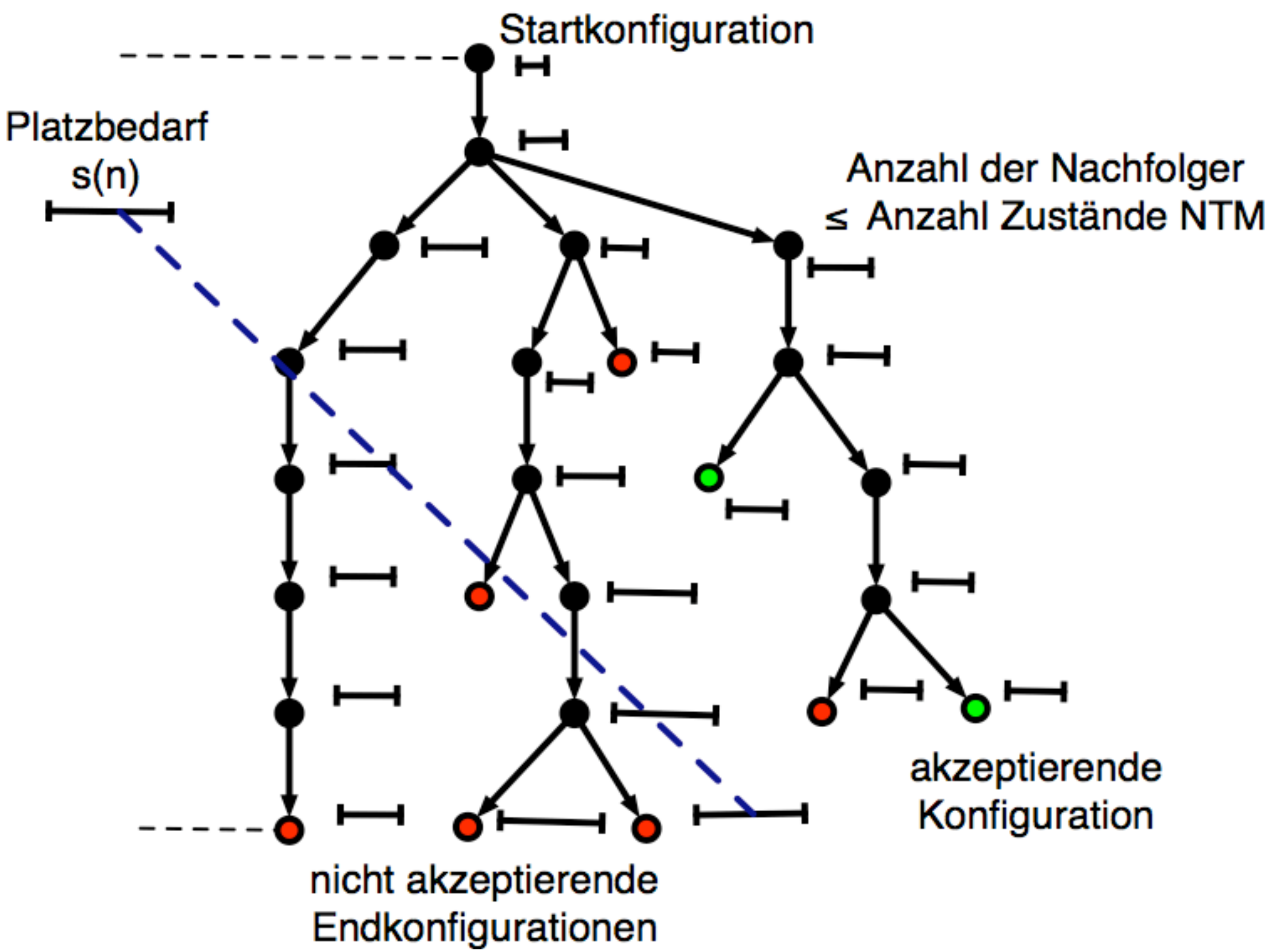
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Berechnungsbaum einer NTM



Nichtdeterministische Platzkomplexitätsklassen

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer



Platzkomplexitätsklassen

- **Definition**
  - Sei f: N → N eine Funktion. Die Platzkomplexitätsklasse SPACE(f(n)) und NSPACE(f(n)) sind wie folgt definiert
  - **SPACE(f(n))** = { L | L ist eine Sprache, die durch eine O(f(n))-Platz-**DTM** entscheiden wird }
  - **NSPACE(f(n))** = { L | L ist eine Sprache, die durch eine O(f(n))-Platz-**NTM** entscheiden wird }
  - Wird die Anzahl der Bänder auf k beschränkt, schreiben wir
    - $\mathbf{SPACE_{k\text{-}Band}(f(n))}$ oder einfach $\mathbf{SPACE_k(f(n))}$,
    - $\mathbf{NSPACE_{k\text{-}Band}(f(n))}$ oder einfach $\mathbf{NSPACE_k(f(n))}$.

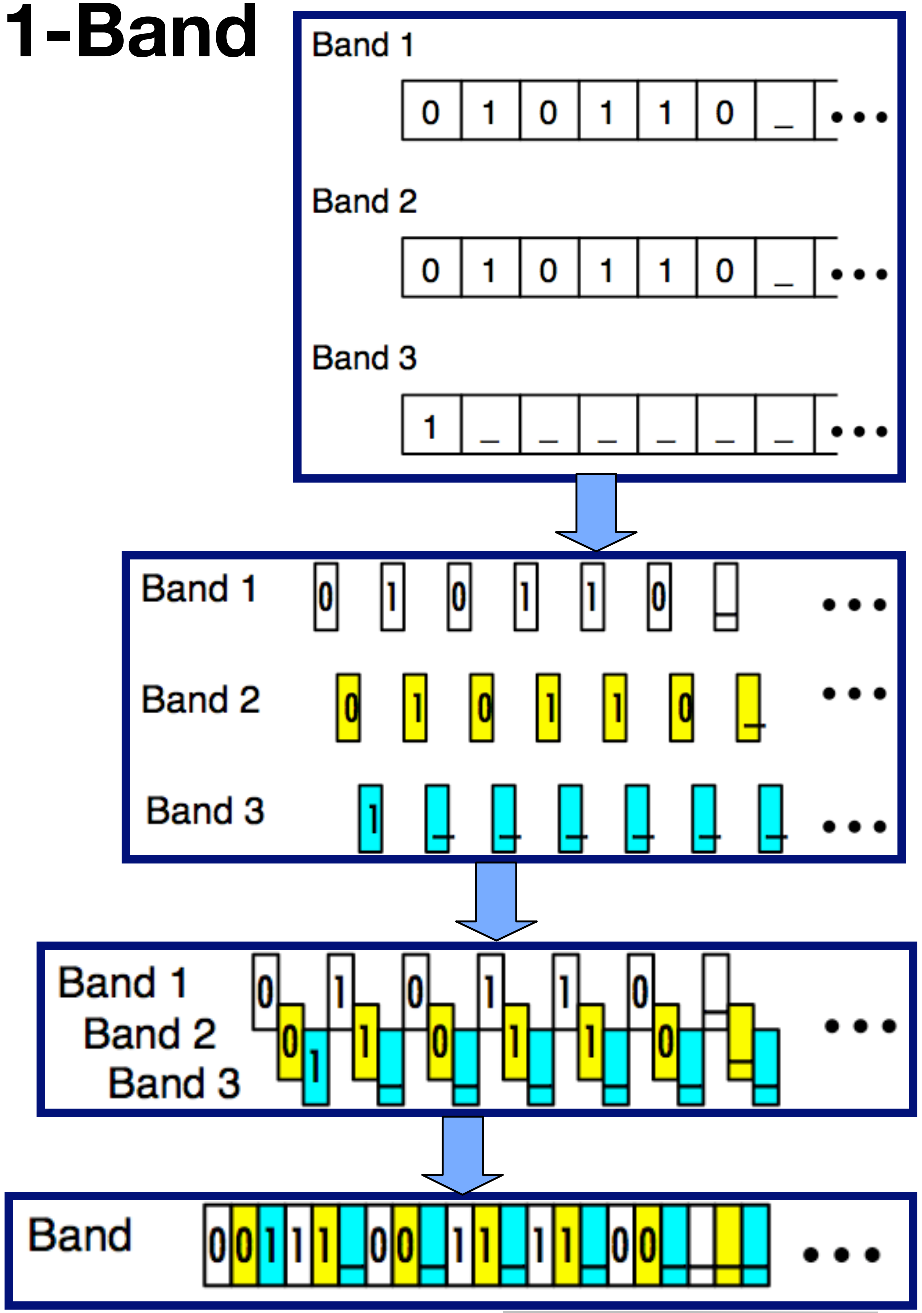
Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Beispiel: SAT ist in $SPACE_2(N)$

- **Betrachte folgende 2-Band-DTM:**
  - "Gegeben eine boolesche Formel F mit m Variablen $x_1, ..., x_m$
  - Für alle Belegungen von $z_1, ..., z_m \in \{0,1\}^m$
    - Setze Belegung $z_1, ..., z_m$ in F ein
      - Ist $F(z_1, ..., z_m)$ wahr, dann halte und akzeptiere
    - sonst verwerfe"
- **Laufzeitanalyse:**
  - Auswerten der Funktion: O(n)
  - Anzahl verschiedener Belegungen ($m \leq n$): 2^n
  - Insgesamt: $O(n\, 2^n)$
- **Platzbedarf:**
  - Auswerten der Funktion: O(n)
  - Speichern der Belegung: O(n)
  - Insgesamt: O(n)

k-Band-DTMs → 1-Band DTMs

- **Theorem**
  - **$SPACE_k(s(n)) \subseteq SPACE_1(O(s(n))$,** d.h.
  - Für $s(n) \geq n$ kann jede Berechnung einer **k-Band-s(n)-Platz-DTM** von einer **1-Band-(k s(n))-Platz DTM** berechnet werden.
- **Beweis(anfang):**
  - Betrachte k-Band-DTM M mit Arbeitsalphabet Γ
  - und konstruiere 1-Band-DTM mit Alphabet $\Gamma \times \{\_, \mathit{kopf}\}$,
    - Speichere das i-te Symbol von Band j an der Position $j + i\, k$.
    - Verwende Markierung *kopf* nur wenn der Kopf der k-Band-TM an der entsprechenden Stelle steht.
  - ...



k-Band-DTMs → 1-Band DTMs

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - **$\mathrm{SPACE}_k(s(n)) \subseteq \mathrm{SPACE}_1(s(n))$**
- **Beweis (Fortsetzung)**
  - Arbeitsweise: 1-Band-DTM
    1. Kodiere Eingabe passend um
    2. Für jeden Schritt der k-Band-DTM
    3. Für jedes Band j
    4. Suche Kopfposition an den Stellen {j,j+k,j+2k,...}
    5. Lies Eingabe
    6. Berechne den Folgezustand, neues Symbol und Bewegungsrichtung
    7. Für jedes Band
    8. Suche Kopfposition
    9. Schreibe neues Zeichen
    10. Bewege Kopfmarkierung um k Zeichen nach links oder rechts
    11. Falls M hält, halte und akzeptiere/verwerfe entsprechend

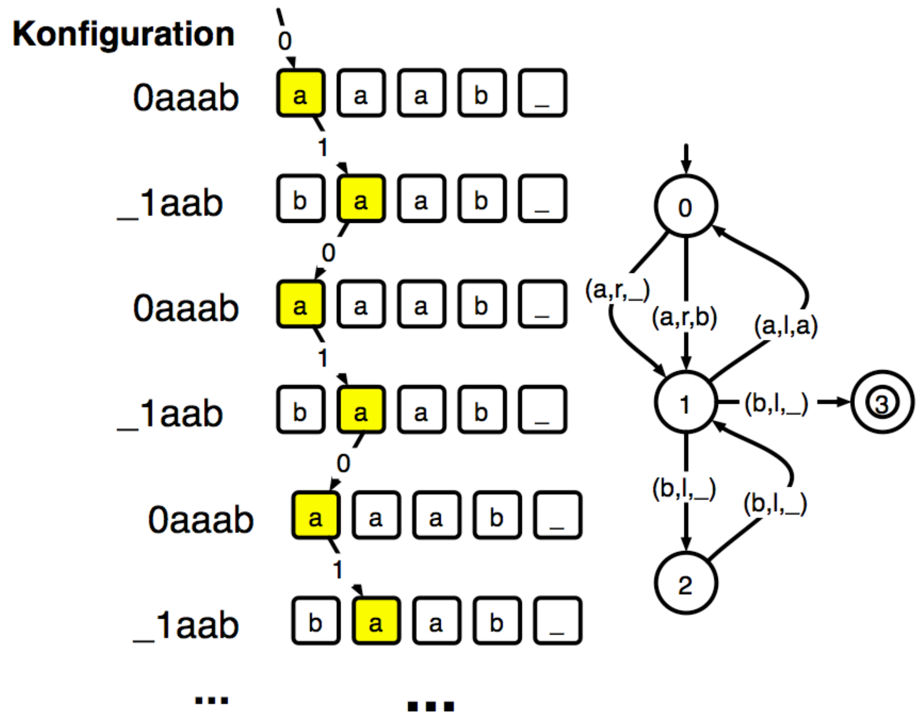
k-Band-DTMs → 1-Band DTMs

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem**
  - **$SPACE_k(s(n)) \subseteq SPACE_1(k\ s(n))$**
- **Beweis (Speicherplatz):**
  - Da die k-Band-DTM höchstens s(n) Zellen besucht, wird die 1-Band-DTM höchstens k s(n) Bandzellen besuchen.
- **Theorem**
  - **Jede Berechnung einer s(n)-Platz-DTM kann durch eine max{n,s(n)/k}-Platz-DTM durchgeführt werden.**
- **Beweis**:
  - Erweitere das Bandalphabet von Σ auf Σ^k.
  - Jeweils k benachbarte Zeichen $a_1,a_2,..,a_k$ werden in das Zeichen $(a_1,a_2,..,a_k)$ umgeformt
  - Die Zustandsmenge wird entsprechend vergrößert
  - und die Zustandsübergänge angepasst:
    - Zuerst kompriminiert die neue DTM die Eingabe um den Faktor k
    - Dann führt sie die Berechnung analog auf dem komprimierten Zeichensatz durch.

Maximale Berechnungszeit einer s(n)-Platz-DTM/NTM

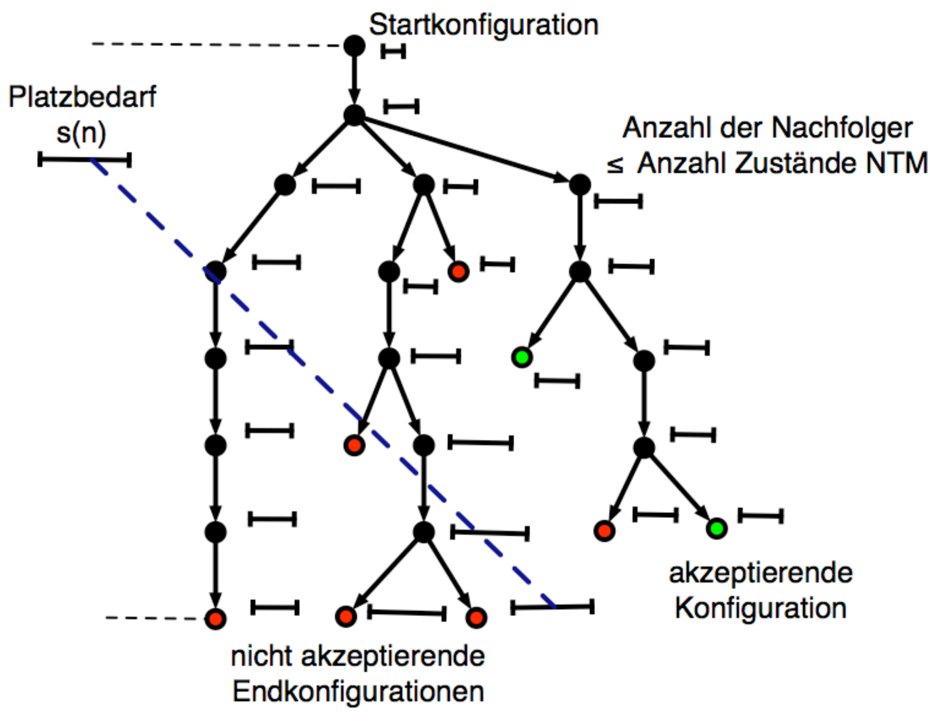
- **Lemma**
  - Jede s(n)-Platz-DTM hat eine Laufzeit von $2^{O(s(n))}$.
- **Beweis:**
  - Es gibt höchstens $2^{O(s(n))}$ verschiedene Konfigurationen der DTM
  - Wiederholt sich eine Konfiguration, so wiederholt sich diese immer wieder und die DTM hält niemals
- **Lemma**
  - Jede s(n)-Platz-NTM hat eine Laufzeit von $2^{O(s(n))}$.
- **Beweis**
  - Analog zur DTM:
  - Wenn sich auf einem Berechnungspfad eine Konfiguration wiederholt, dann wird sie sich immer wieder und wieder und wieder wiederholen.



Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Nicht der Satz von Savitch

- **Schwacher-Satz: Für $s(n) \geq n$**
  - **$NSPACE_1(s(n)) \subseteq SPACE_3(2^{O(s(n))})$, d.h.**
  - Jede Berechnung einer s(n)-Platz 1-Band-NTM kann von einer 3-Band DTM in Platz $2^{O(s(n))}$ durchgeführt werden.
- **Beweis:**
  - Simuliere alle Berechnungspfade der NTM durch
  - Dadurch wird die Berechnung deterministisch
- **Warum ist die Platzschranke so schlecht?**
  - Der Baum kann exponentiell tief sein: $2^{O(s(n))}$
  - Dann braucht man allein zum Abspeichern, welchen Ast man gerade berechnet ein exponentiell langes Wort



Der Satz von Savitch

- **Theorem: Für jede Funktion $s(n) \geq n$**
  - **$NSPACE_1(s(n)) \subseteq SPACE_3(s^2(n))$, d.h.**
  - Jede Berechnung einer t-Zeit 1-Band-NTM kann von einer 3-Band DTM in Zeit **$s^2(n)$** durchgeführt werden.
- **Beweis:**
  - Betrachte NTM für $L \in NSPACE_1(s(n))$, die mit einer **eindeutigen Konfiguration C_{akz}** akzeptiert, d.h.
    - Es gibt nur einen akzeptierenden Zustand
    - Am Ende der Berechnung wird das Band gelöscht
  - Betrachte das Prädikat **`Erreicht-Konf`(C,C',S,T):**
    - Dieses Prädikat ist wahr, wenn die S-Platz-NTM M ausgehend von der Konfiguration C die Konfiguration C' innerhalb von T Schritten erreicht.
  - **Lemma: `Erreicht-Konf`(C,C',S,T) kann von einer 2-Band-(S log T)-Platz-DTM entschieden werden**
    - noch zu beweisen
  - Nun ist $T \leq 2^{O(s(n))}$ und damit ist $\log T = O(s(n)) \leq c\, s(n)$ für eine Konstante $c>0$.
  - **Sei C_{start} die Startkonfiguration**
  - **Das Prädikat `Erreicht-Konf`(C_{start}, C_{akz}, $s(n)$, $2^{c\, s(n)}$) entscheidet L.**
  - Dann kann eine 3-Band-DTM L in Platz
    $c\, s(n)\, s(n) = c\, s(n)^2 = O(s^2(n))$ die Sprache L entscheiden

Beweis des Lemmas

- **Lemma: Das Prädikat Erreicht-Konf(C,C',S,T) kann eine DTM mit 2 Bändern mit Platzbedarf s(n) log T entscheiden.**
  - Diese Prädikat **Erreicht-Konf(C,C',S,T)** ist wahr, wenn die S-Platz-NTM M ausgehend von der Konfiguration C die Konfiguration C' innerhalb von T Schritten erreicht.
- **Beweis**
  - Betrachte folgende DTM M' auf Eingabe (C,C',S,T)
    - Falls T=0 dann
      - Akzeptiere falls C=C' und verwerfe falls $C \neq C'$
    - Falls T=1 dann
      - Akzeptiere falls C' eine erlaubte Nachfolgekonfiguration von C ist oder C=C', ansonsten verwerfe
    - Falls T>1 dann
      - Für alle Konfigurationen Z der Länge S
        - Berechne rekursiv r_1 = Erreicht-Konf(C,Z,S,$\lfloor T/2 \rfloor$)
        - Berechne rekursiv r_2 = Erreicht-Konf(Z,C',S,$\lceil T/2 \rceil$)
        - Falls r_1 und r_2 gilt, dann halte und akzeptiere
      - Verwerfe

- **Analyse Platzverbrauch**
- **Platzverbrauch = Eingabelänge = s(n)**
- **Platzverbrauch: zusätzlich s(n)+1 in jeder Rekursionstiefe**
- **Anzahl Rekursionstiefen: log T**

Der Satz von Savitch (Nachschlag)

Albert-Ludwigs-Universität Freiburg
Institut für Informatik
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

- **Theorem: Für jede Funktion s(n) $\geq$ n**
  - **$NSPACE_1(s(n)) \subseteq SPACE_3(s^2(n))$, d.h.**
- **Beweis:**
  - Zusammenfassung
    - Sei C_{start} die Startkonfiguration
    - Sei C_{akz} die eindeutige akzeptierende Endkonfiguration
    - Das Prädikat **`Erreicht-Konf`**(C_{start}, C_{akz}, s(n), $2^{c\,s(n)}$) entscheidet L in Platz $O(s^2(n))$
  - **Aber wie soll man $2^{c\,s(n)}$ berechnen?**

Ende der 25. Vorlesung

Albert-Ludwigs-Universität Freiburg
Rechnernetze und Telematik
Prof. Dr. Christian Schindelhauer

Arne Vater
Wintersemester 2006/07
25. Vorlesung
01.02.2007