Algorithmen und Datenstrukturen 2

Prüfungsausarbeitung

1 Inhalt

1 Inhalt 1
2 Vorwort 2
3 History 2
4 Prüfung 07.11.2008 3
4.1 Beispiel 1 3
4.2 Beispiel 2 4
4.3 Beispiel 3 5
4.4 Beispiel 4 5
4.5 Beispiel 5 5
5 Prüfung 10.10.2008 6
5.1 Beispiel 1 6
5.2 Beispiel 2 7
5.3 Beispiel 3 8
5.4 Beispiel 4 9
5.5 Beispiel 5 9
6 Prüfung 27.06.2008 10
6.1 Beispiel 1 10
6.2 Beispiel 2 11
6.3 Beispiel 3 11
6.4 Beispiel 4 11
6.5 Beispiel 5 12
7 Prüfung 25.04.2008 13
7.1 Beispiel 1 13
7.2 Beispiel 2 13
7.3 Beispiel 3 14
7.4 Beispiel 4 14
7.5 Beispiel 5 15
8 Prüfung 14.03.2008 16
8.1 Beispiel 1 16
8.2 Beispiel 2 16

8.3 Beispiel 3 17
8.4 Beispiel 4 18
8.5 Beispiel 5 18
9 Prüfung 30.01.2008 19
9.1 Beispiel 1 19
9.2 Beispiel 2 19
9.3 Beispiel 3 20
9.4 Beispiel 4 21
9.5 Beispiel 5 21

2 Vorwort

Diese Prüfungsausarbeitung soll als Vorbereitung für die Algorithmen und Datenstrukturen Prüfung 2 dienen. Sämtliche Prüfungen vom Jahr 2008 sind enthalten. Ältere Prüfungen wurden nicht berücksichtigt, da sich aufgrund der Studienplanänderung einige Themengebiete dieser Lehrveranstaltung geändert haben.

3 History

Datum	Änderung
16.01.2009	Erstellung des Dokuments
17.01.2009	Beispiel 4.1 hinzugefügt
18.01.2009	Beispiel 6.2b, 8.2c und 9.2a geändert
20.01.2009	Beispiel 4.1d, 4.3b,c, 8.5a,b, 9.5a,b eingefügt
21.01.2009	Beispiel 5.2d, 6.2a, 9.2a geändert
22.01.2009	Beispiel 9.2a geändert

4 Prüfung 07.11.2008

4.1 Beispiel 1

a.

Abbildung 1: Perfekte Skipliste

b. Suche nach 25

Anzahl	Vergleichswert	Anmerkung
1	27	27<25 i dekrementieren
2	13	13<25 p=p.next(2)
3	27	27<25 i dekrementieren
4	18	18<25 p=p.next(1)
5	27	27<25 i dekrementieren
6	23	23<25 p=p.next(0)
7	27	27<25 i dekrementieren ENDE
8	23	p !=x

c. Suche nach 18

Anzahl	Vergleichswert	Anmerkung
1	27	27<18 i dekrementieren
2	13	13<18 p=p.next(2)
3	27	27<18 i dekrementieren
4	18	18<18 i dekrementieren
5	15	15<18 p=p.next(0)
6	18	18<18 i dekrementieren ENDE
7	18	p==x

d. **Einfügen**

Alle Elemente der Skipliste in einem Array abspeichern (entlang von L.kopf.next(0) gehen) neues Element hinzufügen und Array der Größe nach ordnen
Skipliste der Reihe nach (auf ebene next(0)) durchgehen und aktuellen key durch den aktuellen Key im Array ersetzen. So lange bis letzter Container erreicht. Jetzt fehlt nur mehr 1 Element.
Höhe des letzten Container bestimmen:
Skipliste durchgehen bis Zeiger auf höchster Ebene (next(i)) auf unendlich zeigt. Falls dies der Fall ist, i vermindern (aber nur bis Ebene 1).
Container ab diesem Element (=akt_element) zählen (=anz_el). Falls 2 hoch akt_element.höhe > anz_el +1 nach dem Container kann Element eingefügt wird (mit Höhe i) falls nicht muss Container mit Höhe l.kopf.hoehe+1 erstellt werden und Kopf und Ende auch erhöht werden bzw. die entsprechenden kopf.next(hoehe+1) auf den neuen Container gesetzt werden.

Suchen

Suchen geht für perfekte Skiplisten ident wie für randomisierte Skiplisten

4.2 Beispiel 2

a. B,C,AD,ACE,DD,DCB,DCC,DCAA,DCDE

b.

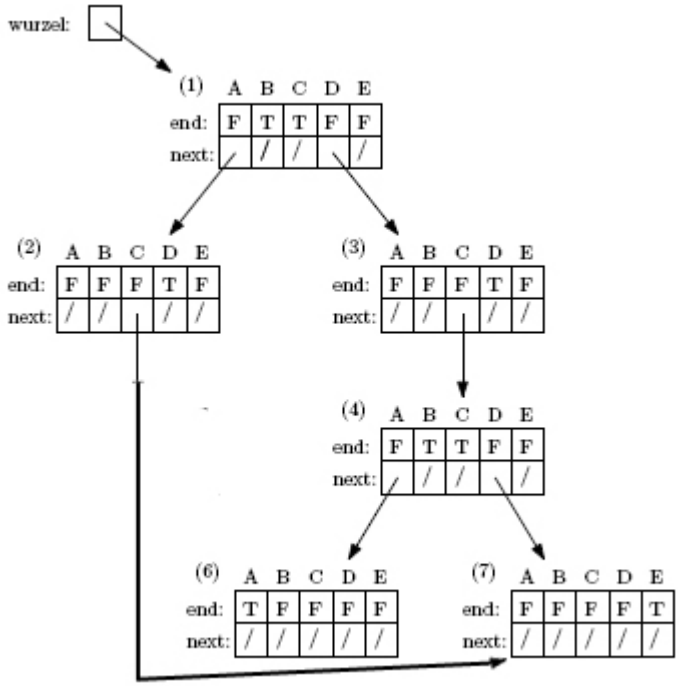


Abbildung 2: Indexed Trie mit Suffix Compression

c.

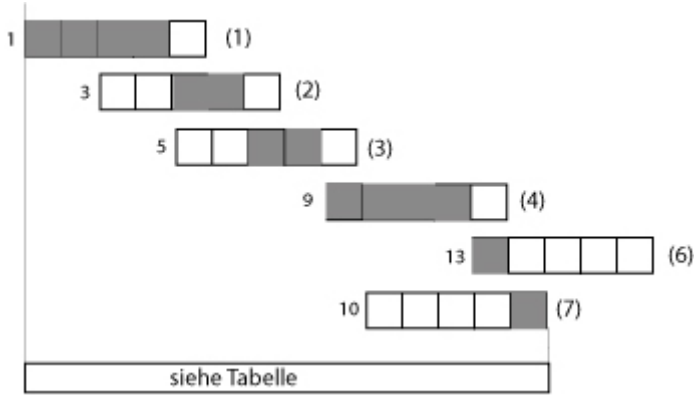


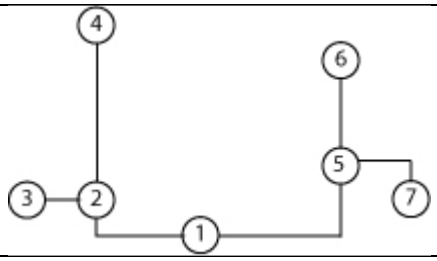
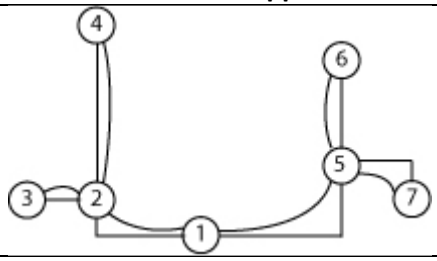
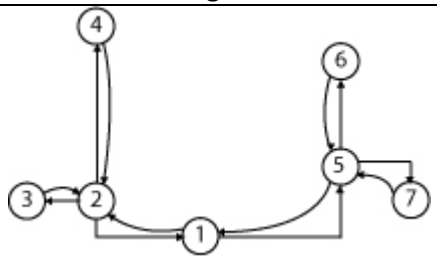
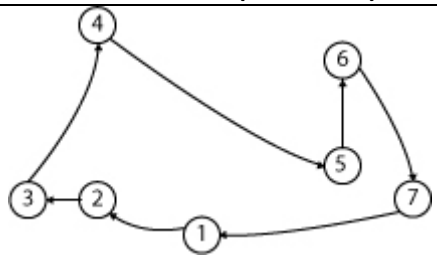
Abbildung 3: Packed Trie

1	2	3	4	5	6	7	8	9	10	11	12	13	14
A	B	C	D	C	D	C	D	A	B	C	D	A	E
F	T	T	F	F	T	F	T	F	T	T	F	T	T
3	N	N	5	10	N	9	N	13	N	N	10	N	N

Index der Wurzel: 1

4.3 Beispiel 3

a.

MST	Kanten verdoppeln
	
Orientierung einzeichnen	**Tour erstellen (start bei 1)**
	

b. Approximationsgüte ST-Heuristik für das euklidische TSP:

$$\frac{c_{ST}(P)}{c_{opt}(P)} \leq 2$$

c. Polynomial Time Approximation Scheme (PTAS)
Das euklidische TSP kann beliebig nahe approximiert warden. Gütegarantie $\varepsilon > 1$kann mit Laufzeit $O\left(N^{\frac{1}{\varepsilon-1}}\right)$ approximiert werden. Eine solche Familie von Approximationsalgorithmen wird als PTAS bezeichnet.

4.4 Beispiel 4

FEHLT

4.5 Beispiel 5

a. Wenn die Hashwerte übereinstimmen besteht die Möglichkeit eines Matches. Die Textstellen müssen jedoch noch genauer überprüft werden.

Wenn die Hashwerte nicht übereinstimmen, kann man sicher sagen, dass es sich um KEINEN Match handelt.

b. siehe 8.5a

c. n ist prim heißt, das es keinen Zeugen gibt, der beweist, dass n nicht prim ist, es gibt jedoch zahlen die durch den Miller-Rabin Test durchkommen, obwohl sie keine Primzahl sind. Diese nennen sich Carchmichael Zahlen.
n ist nicht prim bedeutet, dass n sicher keine Primzahl sein kann.

d. Eine Tabu Liste speichert bereits gefundene Lösungen, um so ein „Gedächtnis“ zu erstellen. Dieses hilft dabei, dass bereits erforschte Lösungen nicht noch einmal gelöst werden müssen.
Vorteile lange Tabu Liste: sämtliche Ergebnisse werden in der Tabu-Liste gespeichert und es muss so mit hoher Wahrscheinlichkeit bereits generierte Lösung nicht erneut berechnet werden.
Nachteil: hoher Speicherplatzbedarf; bei großer Tabu Liste wird der Zeitaufwand, um zu finden ob Lösung in Tabu-Liste vorhanden ist, groß.

5 Prüfung 10.10.2008

5.1 Beispiel 1

next(P = AAABAABA)

Q	AAABAABA	Next	Shift
1	A#	0	1
2	AA#	1	1
3	AAA#	2	1
4	AAAB#	0	4
5	AAABA#	1	4
6	AAABAA#	2	4
7	AAABAAB#	0	7
8	AAABAABA#	1	7

next(P^{-1}= ABAABAAA)

Q	ABAABAAA	Next	Shift
1	A#	0	1
2	AB#	0	2
3	ABA#	1	2
4	ABAA#	1	3
5	ABAAB#	2	3
6	ABAABA#	3	3
7	ABAABAA#	4	3
8	ABAABAAA#	1	7

Suffix

Q	S	Erklärung
1	7	
2	7	
3	7	
4	7 → 3	Q=7: 7-next^-1(7) = 3 && 7>3 --> S=3
5	7 → 3	Q=6: 6-next^-1(6) = 3 && 7>3 --> S=3
6	7 →3	Q=5: 5-next^-1(5) = 3 && 7>3 --> S=3
7	7 →2	Q=3: 3-next^-1(3) = 2 && 7>2 --> S=2 Q=4: 4-next^-1(4) = 3 && 2>3 NO Q=8: 8-next^-1(8) = 7 && 2>7 NO
8	7 →1	Q=1: 1-next^-1(1) = 1 && 7>1 --> S = 1 Q=2: 2-next^-1(2) = 2 && 1>2 NO

a) **suffix:** 7,7,73,3,3,2,1
b) **last:** A=8, B = 7, C=0, D=0…

c)

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
T	A	A	A	B	A	C	B	A	A	A	A	B	A	A	B	A	
last	A	A	A	B	A	**A**	B	A									I=0;J=6; suff(6)=3; 6-last((T6)=**6**
suff							A	A	A	B	A	A	**B**	A			I=6; J=7; suff(7)=**2**; 7-last(T(13))=-1
match									A	A	A	B	A	A	B	A	I=8

5.2 Beispiel 2

a.

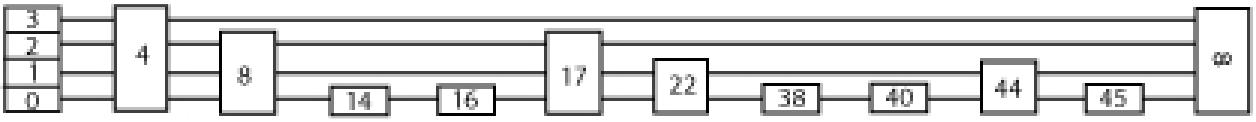


Abbildung 4: randomisierte Skipliste

b.

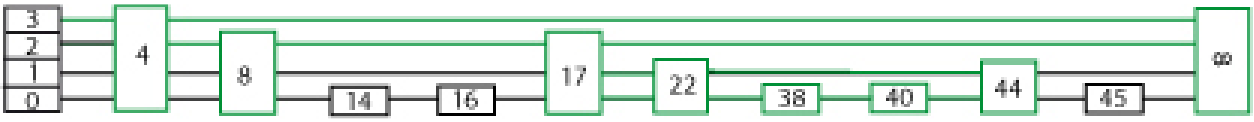


Abbildung 5: randomisierte Skipliste - Suche nach 44

c.

Anzahl	Vergleichswert	Anmerkung
1	4	4<44 p = p.next(3)
2	Inf	inf<44 i dekrementieren
3	8	8<44 p = p.next(2)
4	17	17<44 p=p.next(2)
5	Inf	Inf<44 i dekrementieren
6	22	22<44 p=p.next(1)
7	44	44<44 i dekrementieren
8	38	38<44 p=p.next(0)
9	40	40<44 p=p.next(0)
10	44	44<44 i-- Schleife aus
11	44	Vergleich p == x

d. Im Gegensatz zur perfekten Skipliste kann das einfügen eines neuen Elements nicht zur Reorganisation der gesamten Skipliste führen.
Nachteile
Die Laufzeit ist variabel. Es könnte theoretisch auch jedes nur ein Element mit der Höhe 0 erzeugt werden. Sprich: Die einzelne Laufzeit kann sehr schlecht sein, aber wenn man es öfters ausführt ist sie dann im Mittel sehr gut (also log(n))
Und die Analyse sind komplex (sind halt die algemeinen Probleme von randomisierten Algorithmus). Mehr wüsste ich jetzt auch nicht. (Danke an michaelllll)

5.3 Beispiel 3

a. GLAUBE FAST DAS ICH B WEGLASSEN HÄTTE KÖNNEN → WIRD NIE BENÖTIGT

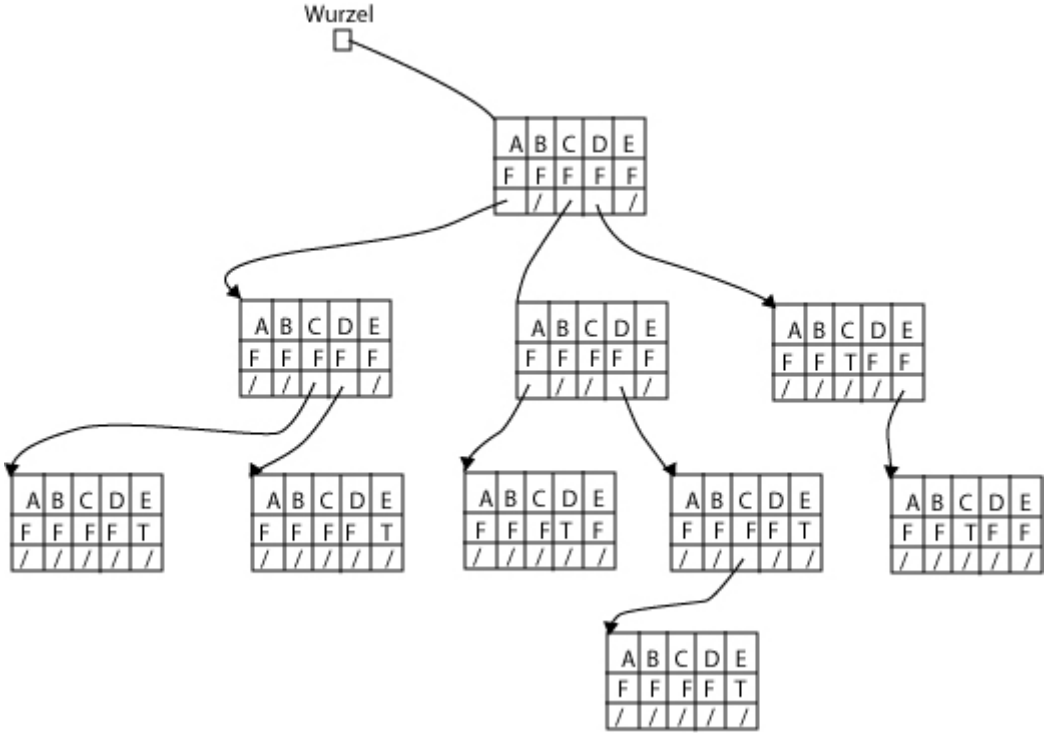


Abbildung 6: Indexed Trie

b.

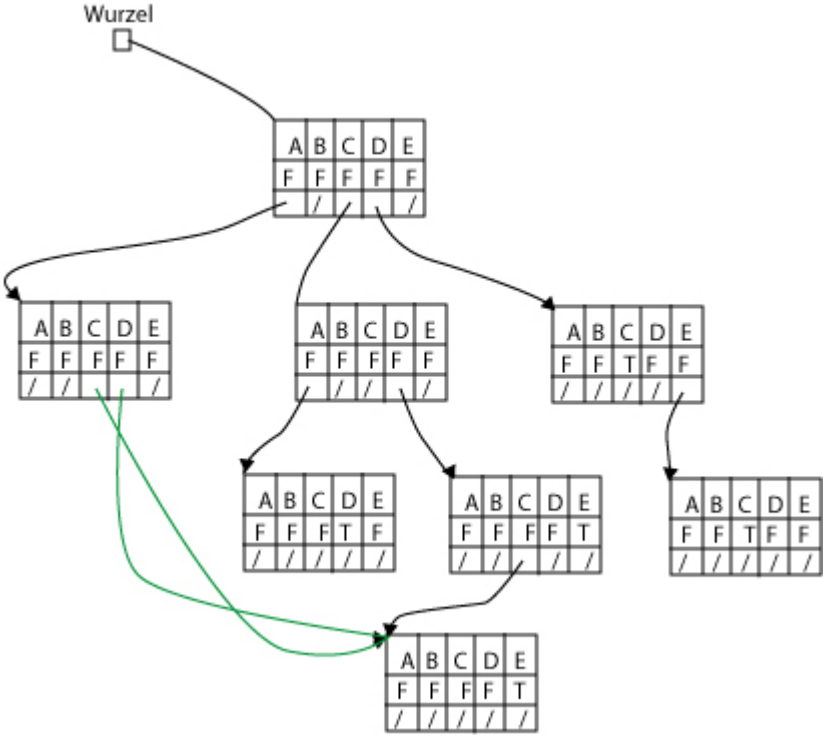


Abbildung 7: Indexed Trie mit Suffix Compression

c. Während in einem Index Trie sehr viele Einträge mit Next = NULL und end == false existieren, werden diese bei einem Linked Trie entfernt. Dazu werden die Felder in eine lineare Liste abgelegt. Zusätzlich muss das jeweilige c(Buchstabe/Wert) gespeichert werden.

5.4 Beispiel 4

FEHLT

5.5 Beispiel 5

a. siehe 8.5a
b. siehe 8.5b
c. In der Scan Line Struktur werden alle im aktuellen Schritt vorhandenen Linien gespeichert. Die Scan Line Status Struktur speichert die Segmente geordnet nach der y Koordinate.
d. Bei zeitlicher Lokalität geht es darum, das gleiche Speicherbereiche mit hoher Wahrscheinlichkeit in einer kurzen Zeitspanne öfters angesprochen werden. Daten werden hierzu länger im Cache behalten, um so eventuell zusätzliche Cache Hits durch zeitliche Lokalität zu erreichen.

6 Prüfung 27.06.2008

6.1 Beispiel 1

next(P = AAACBAA)

Q	AAACBAA	Next	Shift
1	A#	0	
2	AA#	1	
3	AAA#	2	
4	AAAC#	0	
5	AAACB#	0	
6	AAACBA#	1	
7	AAACBAA#	2	

next(P^{-1}= AABCAAA)

Q	AABCAAA	Next	Shift
1	A#	0	
2	AA#	1	
3	AAB#	0	
4	AABC#	0	
5	AABCA#	1	
6	AABCAA#	2	
7	AABCAAA#	2	

suffix

Q	S	Erklärung
1	5	
2	5	
3	5	
4	5	
5	5→4	Q=6: 6-next^{-1}(6)=4 && 5>4 → S=4 Q=7: 7-next^{-1}(7)=5 && 4>5
6	5 → 1	Q=2: 2-next^{-1}(2)=1 && 5>1 → S=1 Q=5: 5-next^{-1}(5)=4 && 1>4
7	5 → 1	Q=1: 1-next^{-1}(1)=1 && 5>1 → S=1 Q=3: 3-next^{-1}(3)=3 && 1>3 Q=4: 4-next^{-1}(4)=4 && 1>4

a. last: A=7; B = 5, C=4
b. suffix: 5,5,5,5,4,1,1
c.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	
T	A	B	A	B	C	C	A	A	A	C	B	A	A	A	C	B	
last	A	A	A	C	B	**A**	A										i=0;J=6;suff(6) = 1; 6-last(T(6)) = 2
suff			A	A	A	C	**B**	A	A								i=2;j=5;suff(5) = 4; 5-last(T(7)) = -2
match							A	A	A	C	B	A	A				i=6; i+suff(1) = 11
												A	A	A	C	B	ENDE erreicht

6.2 Beispiel 2

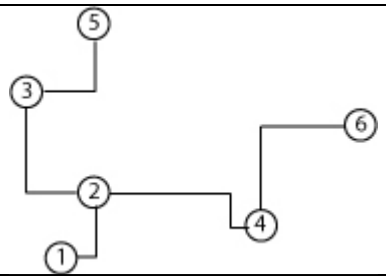
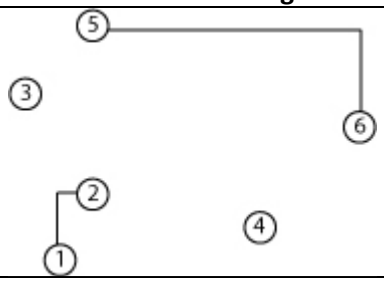
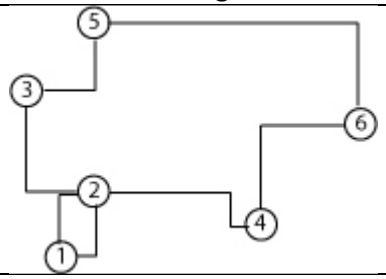
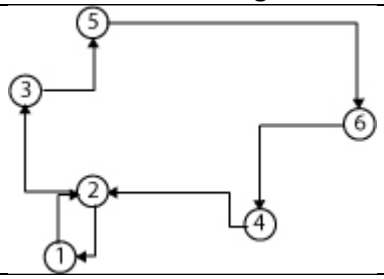
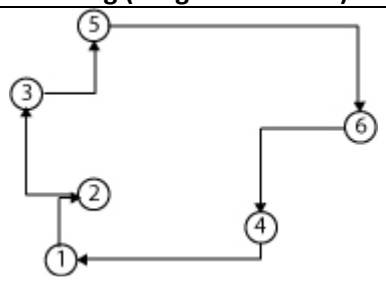
a. Schnitt von Liniensegmenten: O((n+k)logn); benötigter Speicherplatz: $O(n^2)$
Sortieren der 2n Endpunkte nach x-Koordinate O(nlogn)
Zugriff auf ES $O(\log n^2) = O(\log n)$; Zugriff auf SSS: O(logn)
Gesamtlaufzeit: O((n + k) log n) (Danke an michaelllll)

b.

Wann wird der Schnittpunkt AnB (5) in die Ereignisstruktur eingefügt?	Beim Einfügen von A (3)
Wann wird der Schnittpunkt AnD (9) in die Ereignisstruktur eingefügt?	Beim Entfernen von E (8)
Wann wird der Schnittpunkt AnC (10) in die Ereignisstruktur eingefügt?	Beim Schnittpunkt AnD(9)
Wann wird der Endpunkt von C (12) in die Ereignisstruktur eingefügt?	Bei der Initialisierung (0)

Änderung bei Antwort 2,3 – Danke an Sharkal

6.3 Beispiel 3

MST	**Perfect Matching**
	
Hinzufügen	**Orientierung**
	
Lösung (ausgehend von 1)	
	

b) Nein die Lösung ist nicht immer eindeutig. Es kommt darauf an, welcher Startpunkt gewählt wird. Wenn in unserem Beispiel z.B.: 5 als Startpunkt gewählt worden wäre, würde folgende Lösung herauskommen: 5-6-4-2-1-3-5

6.4 Beispiel 4

FEHLT

6.5 Beispiel 5

a. siehe 4.1 d
b. Beim Linked Trie werden nicht benötigte Einträge (next == NULL && end == false) aus der Datenstruktur entfernt um Sie kompakter zu machen. Dabei werden die Felder jedes Knotens durch eine Linked List ersetzt.
   Vorteile: bei Problemen, die nur sehr wenige Zeichen verwenden kann enorm viel Platz gespart werden, weil nicht verwendete Zeichen nicht gespeichert werden.
   Nachteile: Es muss zusätzlich das aktuelle Zeichen gespeichert werden. Aufwand kann durch das benötigte Iterieren bei größerem A und dichten Tries ein höherer Aufwand als bei Indexed Tries sein. O(m*|A|)
c. Für Minimierungsprobleme: falls $\frac{c_A(P)}{c_{opt}(P)} \leq \varepsilon$ für alle Probleminstanzen P und $\varepsilon > 0$ dann ist A ein ε-approximativer Algorithmus.

   Gütegarantie Christophides Heuristik:

$$\frac{c_{CH}(P)}{c_{opt}(P)} \leq \frac{3}{2} \; f\text{ür } alle\ P\ \in \Pi$$

7 Prüfung 25.04.2008

7.1 Beispiel 1

a. ne

Q	BABBABAB	Next	Shift
1	B#	0	1
2	BA#	0	2
3	BAB#	1	2
4	BABB#	1	3
5	BABBA#	2	3
6	BABBAB#	3	3
7	BABBABA#	2	5
8	BABBABAB#	3	5

b.

	1	2	3	4	5	6	7	8	9	10	11	12						
	B	A	B	A	B	B	A	B	B	A	B	A	B	B	A	B	A	B
	B	A	B	**B**	A	B	A	B										
			B	A	B	B	A	B	**A**	B								
Match						B	A	B	B	A	B	A	B					
Match											B	A	B	B	A	B	A	B

c. Wenn die Hashwerte übereinstimmen besteht die Möglichkeit eines Matches. Die Textstellen müssen jedoch noch genauer überprüft werden.
Wenn die Hashwerte nicht übereinstimmen, kann man sicher sagen, dass es sich um KEINEN Match handelt.

7.2 Beispiel 2

a. Randomisierte Skipliste

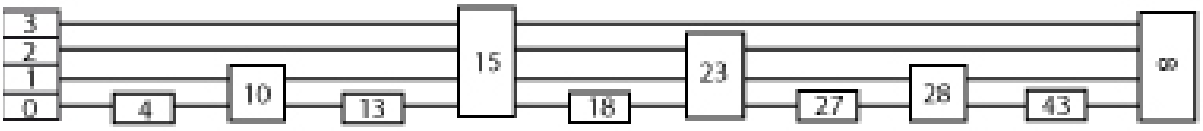


Abbildung 8: Randomisierte Skipliste

b. Suche nach 28

Anzahl	Vergleichswert	Anmerkung
1	15	15<28: p=p.next(3)
2	Inf	Inf<28: i--
3	23	23<28: p=p.next(2)
4	Inf	Inf<28: i--
5	28	28<28: i--
6	27	27<28: p=p.next(0)
7	28	28<28: Ende der Schleife
8	28	p == x

c. Höhe 3 Wahrscheinlichkeit?

$$\frac{1}{2^{i+1}} = \frac{1}{2^4} = \frac{1}{16}$$

7.3 Beispiel 3

a. b,c,ad,aee,cd,ceb,cec,ceaa,ceab, cece

b.

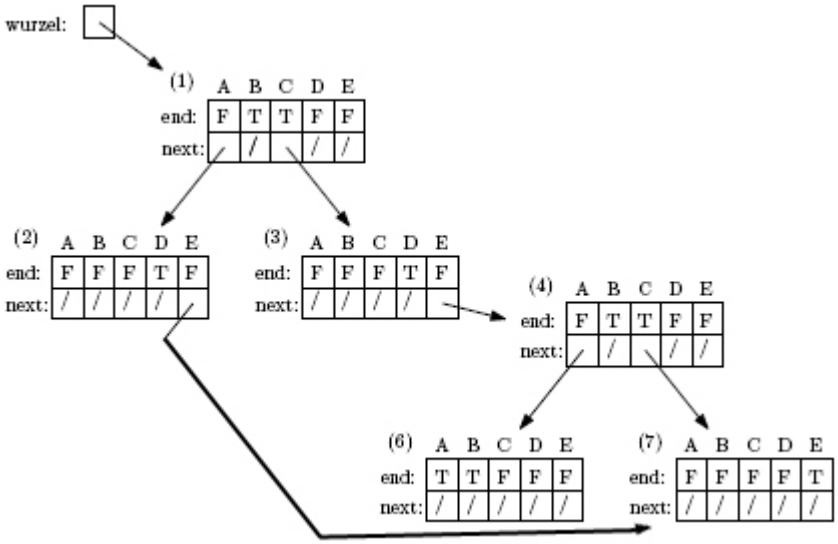


Abbildung 9: Indexed Trie - Suffix Compression

c.

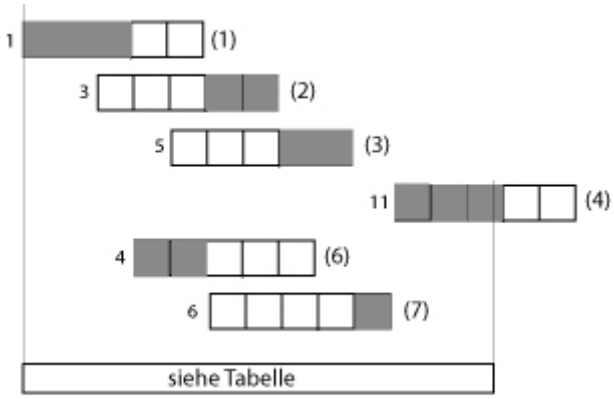


Abbildung 10: Packed Trie

1	2	3	4	5	6	7	8	9	10	11	12	13
A	**B**	**C**	**A**	**B**	**D**	**E**	**D**	**E**	**E**	**A**	**B**	**C**
F	T	T	T	T	T	F	T	F	T	F	T	T
3	N	5	N	N	N	6	N	11	N	4	N	6

Index der Wurzel: 1

7.4 Beispiel 4

FEHLT

7.5 Beispiel 5

a. n ist prim heißt, das es keinen Zeugen gibt, der beweist, dass n nicht prim ist, es gibt jedoch zahlen die durch den Miller-Rabin Test durchkommen, obwohl sie keine Primzahl sind. Diese nennen sich Carchmichael Zahlen.
   n ist nicht prim bedeutet, dass n sicher keine Primzahl sein kann.
b. örtliche Lokalität existiert, wenn Speicherbereiche die öfter angesprochen werden nahe beieinander liegen. Bei Miss werden ganze Speicherblöcke aus dem Cache geholt.
c. Für Maximierungsproblem: falls $\frac{c_A(P)}{c_{opt}(P)} \geq \varepsilon$
   für alle Probleminstanzen P und $\varepsilon > 0$ dann ist A ein ε-approximativer Algorithmus.
d. Nein, die Lösung hängt immer vom Startknoten ab. Da dadurch Knoten in einer anderen Reihenfolge besucht werden und es so zu unterschiedlichen Ergebnissen kommen kann.
   Approximationsgüte ST-Heuristik für das euklidische TSP:

$$\frac{c_{ST}(P)}{c_{opt}(P)} \leq 2$$

8 Prüfung 14.03.2008

8.1 Beispiel 1

a. Randomisierte Skipliste

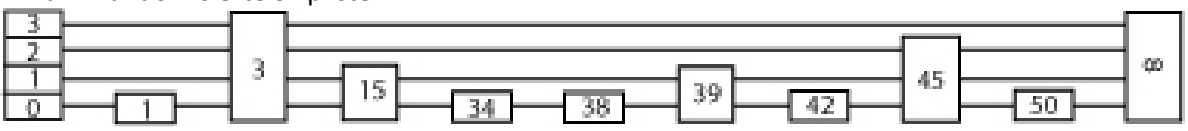


Abbildung 11: Randomisierte Skipliste

b. Suche nach 39

Anzahl	Vergleichswert	Anmerkung
1	3	3>39 p=p.next(3)
2	Inf	Inf>39 i--
3	45	45<39 i--
4	15	15<39 p=p.next(1)
5	39	39<39 i--
6	34	34<39 p=p.next(0)
7	38	38<39 p=p.next(0)
8	39	39<39 i—ENDE
9	39	p ==x

c. siehe 5.2 a

8.2 Beispiel 2

a. Median: $a_m = \left\lceil \frac{l+r}{2} \right\rceil$

Step 1: (nach x sortieren) am = $a_m = \left\lceil \frac{1+10}{2} \right\rceil = 6$

V	L	E	A	O	I	Y	D	R	M
0	1	2	3	4	5	6	7	8	9

Step 2: Teiläste (V,L,E,A,O & Y,D,R,M) nach y sortieren

V	O	A	E	L	I	R	D	M	Y
1	2	4	7	8		0	3	5	9

Step 3: Teiläste (V,O & E,L & R,D & Y) nach z sortieren

O	V	A	L	E	I	D	R	M	Y
4	5		6	7		8	9		9

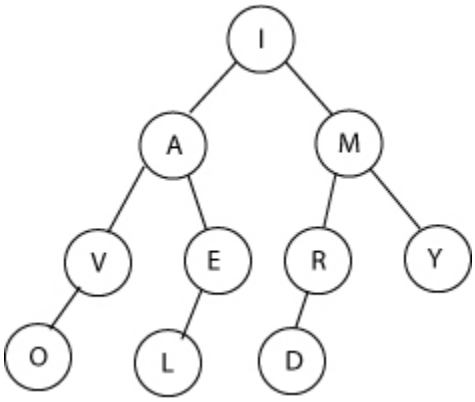


Abbildung 12: 3-dimensionale Bereichssuche

b.

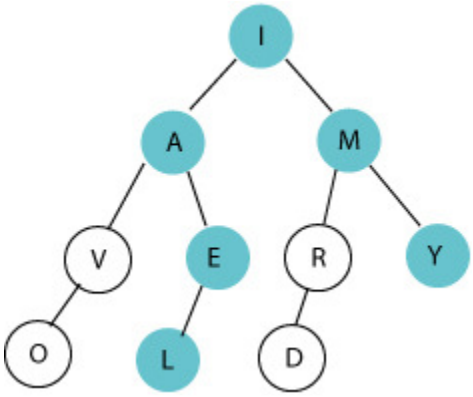
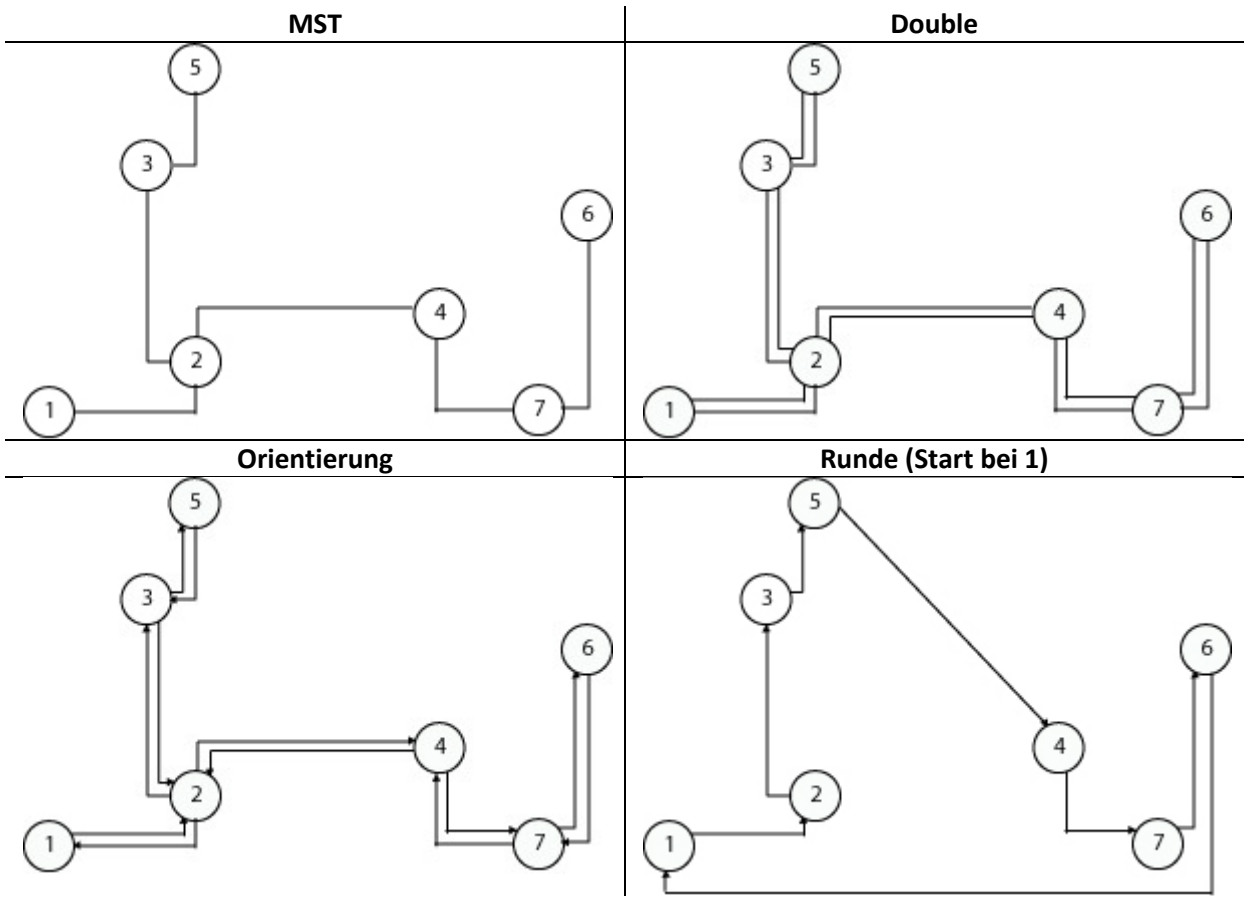


Abbildung 13: Suche nach Bereich

gefundene Punkte im Bereich (3,6,0) – (6,9,5):
I(5,6,2) im Bereich, A(3,4,3) nicht im Bereich; E(2,7,7) nicht im Bereich, L(1,8,6) nicht im Bereich, M(9,5,1) nicht im Bereich, Y(6,9,0) im Bereich (Danke an Sharkal)

8.3 Beispiel 3

a.



b. First Fit Heuristik Bin Packing:
Gegenstände werden in beliebiger Reihenfolge behandelt. Jeder Gegenstand wird in die erstmögliche Kiste gelegt in die er passt.
Approximationsgüte: $\frac{c_{FF}(P)}{c_{opt}(P)} < \frac{17}{10} + \frac{2}{c_{opt}(P)} \quad \forall P \in \Pi$

8.4 Beispiel 4

FEHLT

8.5 Beispiel 5

a. **Approximative Zeichenkettensuche:**
Im Gegensatz zur exakten Zeichenkettensuche wird bei der approximativen Zeichenkettensuche nicht auf genaue Übereinstimmung sondern lediglich um ungefähre.
Naives Verfahren modifizieren:
Überprüfung für gesamtes Wort durchführen und Anzahl der Fehler speichern. Wenn Anzahl Fehler unter einem festgelegten Schwellwert sind, wurde ein approximativer Treffer gefunden.

b. **Lösung des Problems**
Um die Zeuge Funktion effizient zu lösen wurde die Vertauschbarkeit von Multiplikation und Modulo Rechnung angewendet.

$$a^{2c} mod n = (a^c mod n)^2 mod n$$
$$a^{c+1} mod n = (a^c mod n) * a mod n$$

Bitkomplexität
Multiplikation zweier k-bit Zahlen hat Bitkomplexität $O(k^2)$ woraus folgt, dass der s-malige Aufruf der Zeuge() Funktion $O(k^3)$ hat.

c. siehe 5.5 d)

d. siehe 4.5 d)

9 Prüfung 30.01.2008

9.1 Beispiel 1

next(P = ABAAABA)

Q	ABAAABA	Next	Shift
1	A#	0	1
2	AB#	0	2
3	ABA#	1	2
4	ABAA#	1	3
5	ABAAA#	1	4
6	ABAAAB#	2	4
7	ABAAABA#	3	4

next(P^{-1}= ABAAABA)… siehe oben – trotz Invertierung gleiche Zeichenkette

suffix

Q	S	Erklärung
1	4	
2	4	
3	4	
4	4	Q=7: 7-next^{-1}(7)=4 && 4>4
5	4	Q=6: 6-next^{-1}(6)= 4 && 4>4
6	4→2	Q=3: 3-next^{-1}(3)=2 && 4 > 2 → S=2 Q=4: 4-next^{-1}(4)=3 && 2 > 3 Q=5: 5-next^{-1}(5)=4 && 2 > 4
7	4 → 1	Q=1: 1-next^{-1}(1)=1 && 4 >1 → S=1 Q=2: 2-next^{-1}(2)=2 && 1 >2

a. last: A=7; B =6, C=0…
b. suffix: 4,4,4,4,4,2,1
c.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	
T	A	B	A	A	X	B	A	A	A	A	B	A	A	A	B	A	A	B	A	
last	A	B	A	A	**A**	B	A													i=0;j=5; suff(5) = 4; 5-lst(T(5) = **5**
suff						A	**B**	A	A	A	B	A								i=5;j=2; suff(2) = **4**; 2-lst(7)=-5
match										A	B	A	A	A	B	A				i=9, suff(1) = 4
														A	B	A	A	A	B	i=13; END

9.2 Beispiel 2

a.

Wann wird der Schnittpunkt AnD in die Ereignisstruktur eingefügt?	Beim Einfügen von D
Wann wird der Schnittpunkt BnD in die Ereignisstruktur eingefügt?	Beim Schnittpunkt AnD
Wann wird der Schnittpunkt CnE in die Ereignisstruktur eingefügt?	Beim Entfernen von A
Wann wird der Schnittpunkt DnE in die Ereignisstruktur eingefügt?	Beim Einfügen von E

Änderung: Danke an michaelllll & sharkal

b. Der Schnitt zweier Segmente ist stets leer oder genau ein Punkt.
Es schneiden sich nie mehr als zwei Segmente in einem Punkt.
Alle Endpunkte und Schnittpunkte haben paarweise verschiedene x-Koordinaten.

c. Eine Ereignisstruktur repräsentiert den aktuellen Status der Scanline. Es werden Anfangs, End und Schnittpunkte gespeichert. Initialisiert wird die Ereignisstruktur mit sämtlichen Start und Endpunkten die aufsteigend nach ihrem Vorkommen sortiert werden.

9.3 Beispiel 3

a.

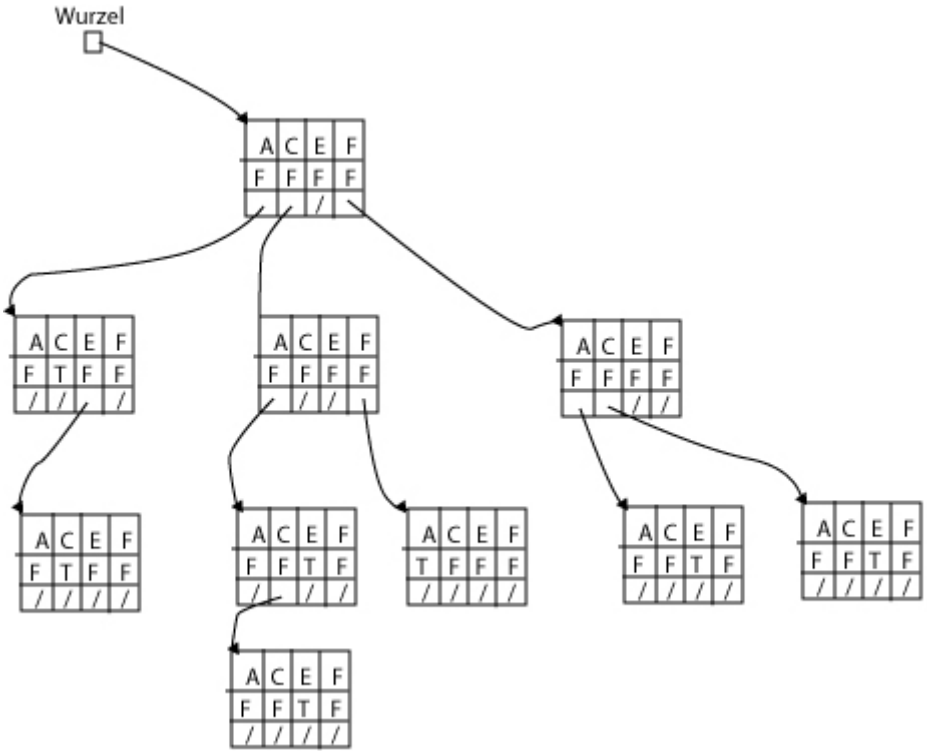


Abbildung 14: Indexed Trie

b.

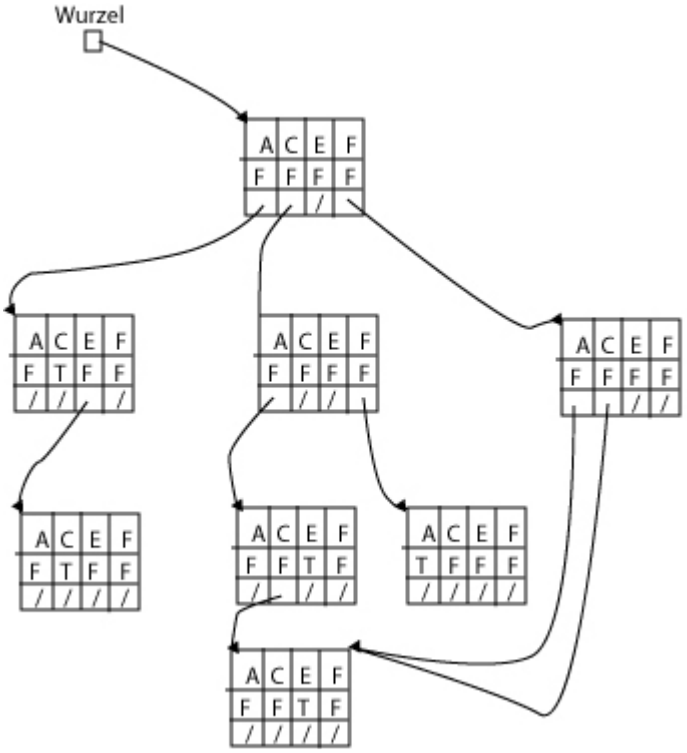


Abbildung 15: Indexed Trie mit Suffix Compression

c. Nein, keine zwei Knoten dürfen den selben Anfangsindex haben. Dies liegt daran, dass sie sich sonst überlappen würden und die Unterscheidbarkeit der beiden Knoten nicht möglich wäre. Es könnte passieren, dass bei zwei Knoten mit gleichen Anfangsindex das Alphabet fortlaufend wäre und dadurch keine Zuordnung zu den einzelnen Knoten möglich wäre.

9.4 Beispiel 4

FEHLT

9.5 Beispiel 5

a. siehe 4.1d

b. Für Minimierungsprobleme: falls $\frac{c_A(P)}{c_{opt}(P)} \leq \varepsilon$ für alle Probleminstanzen P und $\varepsilon > 0$ dann ist A ein ε-approximativer Algorithmus.
Gütegarantie Spanning Tree Heuristik Euklidisches TSP
Approximationsgüte ST-Heuristik für das euklidische TSP:
$$\frac{c_{ST}(P)}{c_{opt}(P)} \leq 2$$

c.

D=

Inf	2	6	8
1	Inf	3	2
4	2	Inf	2
7	7	1	Inf

Zeilenreduktion: Zeile minus Minimum der Zeile

D=

Inf	0	4	6
0	Inf	2	1
2	0	Inf	0
6	6	0	Inf

L = 2+1+2+1 = 6
Spaltenreduktion ist hier sinnlos, da in jeder Spalte bereits eine 0 steht.
Untere Schranke wäre 6! Stimmt das?