

Prüfung aus

Diskrete und geometrische Algorithmen (Hetzl)

Zuname: _____ Matrikelnummer: _____

Vorname: _____

Titelseite ab hier bitte freilassen!

TU Wien, 29.1.2019

Arbeitszeit: 100 Minuten

1)

2)

3)

4)

1) (7 P.) Lösen Sie die Rekursionsgleichung $a_n = 5a_{n-1} - 6a_{n-2}$ (für $n \geq 2$) mit den Anfangswerten $a_0 = 1$ und $a_1 = 5$.

$$z^2 - 5z + 6 = 0 \Leftrightarrow z = \frac{5 \pm \sqrt{25-24}}{2} \Leftrightarrow z = 3 \vee z = 2$$

$$\text{also } a_n = c_1 3^n + c_2 2^n$$

$$\left. \begin{array}{l} 1 = a_0 = c_1 + c_2 \Leftrightarrow c_1 = 1 - c_2 \\ 5 = a_1 = 3c_1 + 2c_2 \end{array} \right\} \Rightarrow 5 = 3(1 - c_2) + 2c_2 = 3 - c_2 \Leftrightarrow c_2 = -2$$

$$\Rightarrow c_1 = 1 - c_2 = 1 + 2 = 3$$

$$a_n = 3^{n+1} - 2^{n+1} = 5 \cdot 3^n - 5 \cdot 2^n - 2 \cdot 3^n + 3 \cdot 2^n = 5(3^n - 2^n) - 6(3^{n-1} - 2^{n-1}) = 5a_{n-1} - 6a_{n-2}$$

2) (7 P.) Wir definieren die Menge von Zeichen

$$A = \{a, b, c\} \cup \{ (i \mid i \in \mathbb{N} \} \cup \{)_i \mid i \in \mathbb{N} \}$$

wobei wir sowohl eine indizierte öffnende Klammer $(i$ als auch eine indizierte schließende Klammer $)_i$ als jeweils ein einzelnes Zeichen betrachten. Sei A^* die Menge aller endlich langen Zeichenketten die nur aus Zeichen aus A bestehen. Die Menge der *indiziert wohlgeklammerten Ausdrücke* ist die kleinste Menge $W \subseteq A^*$ für die gilt:

1. Die leere Zeichenkette $\varepsilon \in W$.
2. Die Zeichen $a, b, c \in W$.
3. Falls $w_1, w_2 \in W$, dann ist auch $w_1 w_2 \in W$.
4. Falls $w \in W$ und $i \in \mathbb{N}$, dann ist auch $(i w)_i \in W$.

So ist zum Beispiel $(3c)_3 ab(3ab(1c)_1)_3 \in W$ aber $bc(1b(2a)_1 cc)_2 \notin W$.

Geben Sie einen Algorithmus in Pseudocode an, der eine Zeichenkette $v \in A^*$ als Eingabe erhält und entscheidet, ob $v \in W$ ist. Die Eingabe wird in Form eines Datenfelds übergeben. Die Laufzeit des Algorithmus muss $O(|v|)$ sein wobei $|v|$ die Anzahl der Zeichen in v ist.

```

Prozedur InW(v)
  if v.länge = 0
    return 1          ▷ z ∈ W
  end if
  if v[0] ∈ {a, b, c}
    return (InW(v[1... v.länge-1]))
  else if v[0] und v[v.länge-1] sind passende Klammern
    return (InW(v[1... v.länge-2]))
  else
    return 0          ▷ v ∉ W, falsch geklammer
  end if

```

Die Laufzeit des Programms ist in $O(|v|)$, da bei jedem rekursiven Aufruf das Wort mindestens um ein Zeichen kürzer wird und somit keine weiteren Kosten anfallen.

3) (7 P.) Sei $G = (V, E)$ ein nicht-leerer endlicher ungerichteter Graph, dann definieren wir $m(G) = \max\{|V_i| \mid 1 \leq i \leq k\}$ wobei $(V_1, E_1), \dots, (V_k, E_k)$ die Zusammenhangskomponenten von G sind. Geben Sie einen Algorithmus in Pseudocode an, der einen nicht-leeren endlichen ungerichteten Graph $G = (V, E)$ als Eingabe erhält und $m(G)$ als Ausgabe liefert. Der Eingabegraph wird in Form einer Adjazenzliste übergeben wobei $V = \{1, \dots, n\}$. Die Laufzeit des Algorithmus muss $O(|V| + |E|)$ sein.

Prozedur $ZSKP(G)$

Sei z neues Datenfeld der Länge $|V|$ überall mit 0 initialisiert

Sei Q Warteschlange

$l := 0$

While $SucheNull(z) < n+1$

$i := SucheNull(z)$

$l := l+1$

$z[i] := l$

Hinzufügen(Q, i)

While $Q \neq \emptyset$

$k := Entfernen(Q)$

for alle $\{k, j\} \in E$

if $z[j] = 0$

$z[j] := z[k]$

Hinzufügen(Q, j)

End if

End for

End While

return $FINDEMAX(z, l)$

Suche Null(z)

$i := 1$

while $i < n+1 \wedge z[i] \neq 0$

$i := i+1$

end while

return i

$FINDEMAX(z, l)$

Sei M neues Datenfeld der Länge l mit 0 initialisiert

for $i = 1, \dots, z\text{-Länge}$

$M[z[i]] := M[z[i]] + 1$

end for

$m := 0$

for $j = 1, \dots, l$

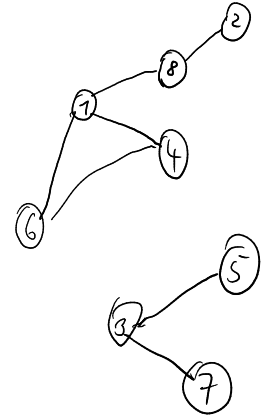
if $M[j] > m$

$m := M[j]$

end if

end for

return m

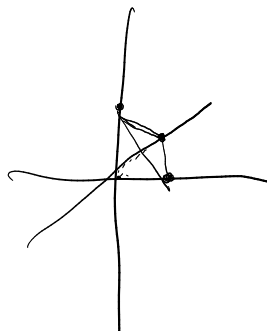


4) (3 P.) Geben Sie ein lineares Programm in Standardform an, das drei optimale Lösungen besitzt die nicht auf einer Gerade liegen.

$$A := (1, 1, 1)$$

$$c := (1, 1, 1)$$

$$b := 1$$



$$Ax \leq b$$

$$cx \text{ maximal}$$

$$\forall i \in \{1, 2, 3\}: x_i \geq 0$$

e_1, e_2, e_3 sind optimale Lsg. die nicht auf einer Geraden liegen

Prüfung aus

Diskrete und geometrische Algorithmen (Hetzl)

Zuname: _____ Matrikelnummer: _____

Vorname: _____

Titelseite ab hier bitte freilassen!

TU Wien, 1.3.2019

Arbeitszeit: 100 Minuten

1)

2)

3)

4)

1) (4 P.) Geben Sie asymptotische Lösungen der folgenden Rekursionsgleichungen an:

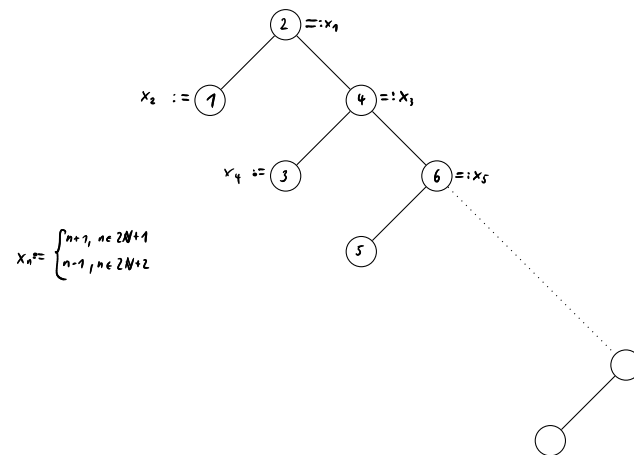
a) $T(n) = 9T(\frac{n}{3}) + n(n+1)$

b) $T(n) = 3T(\frac{n}{4}) + n \log n$

d) $n(n+1) \in \mathcal{O}(n^{\log_3(9)-\varepsilon}) = \mathcal{O}(n^{3-\varepsilon}) \quad \forall \varepsilon < 1$
 daher nach dem Mastertheorem $T(n) \in \Theta(n^{\log_3(9)})$

b) $3 \cdot \frac{n}{4} \log(\frac{n}{4}) \leq c n \log(n) \Leftrightarrow 3 \log(n) - 3 \log(4) \leq 4c \log(n) \Leftrightarrow -3 \log(4) \leq (4c-3) \log(n)$
 $\forall c > \frac{3}{4}$ ist hinr. problem $n_0 \in \mathbb{N}$ erfüllt, also $T(n) \in \Theta(n \log(n))$

2) (4 P.) Geben Sie, für jedes gerade $n \in \mathbb{N}$ paarweise unterschiedliche Schlüssel x_1, \dots, x_n an, so dass ein Suchbaum in Kammform (sh. Abbildung) entsteht wenn, beginnend mit dem leeren Suchbaum, die Schlüssel x_1, \dots, x_n in dieser Reihenfolge eingefügt werden.



3) (7 P.) Sei

$$A = (a_{i,j})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}$$

eine Matrix. Als *Untermatrix* von A bezeichnen wir eine Matrix der Form

$$(a_{i,j})_{\substack{p \leq i \leq q \\ r \leq j \leq s}}$$

wobei $p, r \geq 1$, $q \leq m$ und $s \leq n$.

Geben Sie einen Algorithmus in Pseudocode an der eine Matrix $A \in \{0, 1\}^{m \times n}$ als Eingabe erhält und der als Ausgabe das größte $k \in \mathbb{N}$ zurückliefert so dass A eine $k \times k$ Untermatrix enthält die nur aus Einsen besteht. Die Laufzeit des Algorithmus muss $O(m \cdot n)$ sein.

Hinweis: Dynamische Programmierung, betrachten Sie $m_{q,s}$, die maximale Größe einer quadratischen Einser-Matrix die an der Stelle (q, s) endet.

Prozedur *Untermatrix*

$m_{q,s} := 0$, falls $q=0$ oder $s=0$

for $i=1, \dots, m$

for $j=1, \dots, n$

if $a_{i,j} = 0$

$m_{i,j} := 0$

else if $m_{i-1,i} = m_{i,j-1}$

if $m_{i,j-1} \leq m_{i-1,i-1}$

$m_{i,j} := m_{i-1,i-1} + 1$

else

$m_{i,j} := m_{i-1,j}$

end if

else

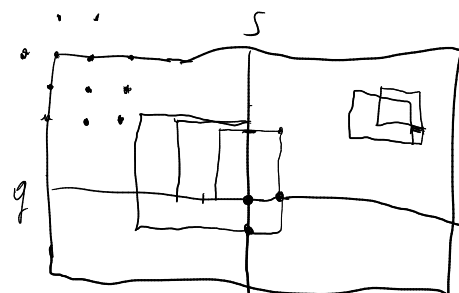
$m_{i,j} := \min\{m_{i,i-1}, m_{i-1,j}\} + 1$

end if

end for

end for

return $\max(m_{i,j})$



$$\begin{matrix} & & & 0 & 0 \\ & & & 1 & 1 \\ & & 1 & 1 & 1 \\ & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{matrix}$$

4) (9 P.) Eine *Strecke* ist ein ungeordnetes Paar $\{p, q\}$ wobei $p, q \in \mathbb{R}^2$ und $p \neq q$. Geben Sie einen Algorithmus in Pseudocode an, der als Eingabe eine endliche Menge von Strecken erhält, bei denen es sich um die Kanten eines konvexen Polygons handelt und der als Ausgabe eine im Uhrzeigersinn sortierte Liste der Knoten des Polygons zurückgibt.

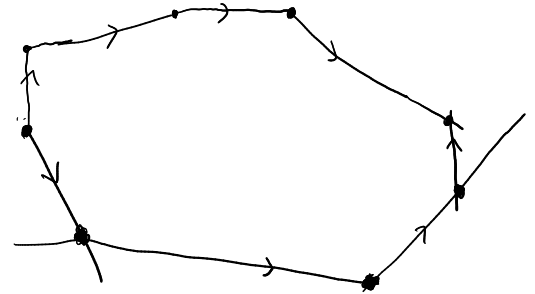
Nehmen Sie der Einfachheit halber an, dass die Eingabe keine zwei Strecken $\{p, q\}, \{q, r\}$ enthält so dass p, q und r auf einer Geraden liegen.

Die Laufzeit des Algorithmus muss bei Eingabe von n Strecken $O(n \log n)$ sein.

```

Sei  $S$  ein Array mit Strecken der Länge  $n$ 
for  $i = 1, \dots, n$ 
     $S[i] := \text{lexsort-k-y}(S[i])$   $\swarrow$  sortiert aufsteigend erst nach  $x$ , dann nach  $y$ 
end for
 $S := \text{lexsort2}(S)$   $\swarrow$  sortiert lexikographisch nach dem ersten Punkt der Strecke
Sei  $P$  neues Datenfeld der Länge  $S$  Länge
 $P[1] := S[1][2]$ 
if  $S[1][2].y < S[2][2].y$ 
     $P[2] := S[1][2]$ 
else
     $P[2] := S[2][2]$ 
end if
 $i := 2$ 
 $j := 3$ 
while  $P[i] <_{\text{lex}} S[S.\text{Länge}][1]$ 
    while  $P[i] \neq S[j][1]$ 
         $j := j + 1$ 
    end while
     $i := i + 1$ 
     $P[i] := S[j][2]$ 
end while
 $j := S.\text{Länge}$ 
while  $P[i] >_{\text{lex}} S[1][1]$ 
    while  $P[i] \neq S[j][1]$ 
         $j := j - 1$ 
    end while
     $i := i + 1$ 
     $P[i] := S[j][2]$ 
end while
return  $P$ 

```



Prüfung aus

Diskrete und geometrische Algorithmen (Hetzl)

Zuname: _____ Matrikelnummer: _____

Vorname: _____

Titelseite ab hier bitte freilassen!

TU Wien, 26.4.2019

Arbeitszeit: 100 Minuten

1)

2)

3)

4)

1) (5 P.) Geben Sie für unendlich viele $n \in \mathbb{N}$ ein Datenfeld A_n der Länge n an, auf dem Einfügesortieren Laufzeit $\Theta(n\sqrt{n})$ hat und bewiesen Sie dass dem so ist.

$$A_1 = (1)$$

$$A_2 = (2, 1)$$

$$A_3 = (3, 1, 2)$$

$$A_4 = (4, 1, 3, 2)$$

$$A_n[i] = \begin{cases} A_{n-1}[i] + 1, & \text{falls } i < n \text{ und } A_{n-1}[i] \geq n - \sqrt{n} \\ A_{n-1}[i], & \text{falls } i < n \text{ und } A_{n-1}[i] < n - \sqrt{n} \\ \lceil \frac{n - \sqrt{n}}{2} \rceil, & \text{falls } i = n \end{cases}$$

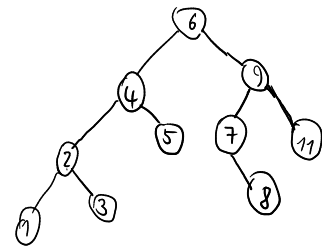
geht es für A_n eine Anzahl von $n\sqrt{n}$ Schritten
dann kommen bei A_{n-1} im letzten Schritt noch
ungefähr \sqrt{n} dazu um das letzte Element
an die richtige Stelle zu bringen

2) (7 P.)

a) Geben Sie einen Algorithmus in Pseudocode an, der als Eingabe einen Suchbaum B mit n Elementen erhält und als Ausgabe ein Datenfeld liefert, das die Elemente von B in aufsteigender Reihenfolge enthält. Die Laufzeit des Algorithmus muss $O(n)$ sein.

b) Geben Sie einen Algorithmus in Pseudocode an, der als Eingabe zwei Suchbäume B_1 und B_2 mit n_1 bzw. n_2 Elementen enthält und feststellt ob deren Schlüsselmengen disjunkt sind. Die Laufzeit des Algorithmus muss $O(n_1 + n_2)$ sein.

a) ~~Prozedur SORT(B)~~



b) ~~Prozedur DISJUNKT(B_1, B_2)~~

~~$A_1 := \text{SORT}(B_1)$~~

~~$A_2 := \text{SORT}(B_2)$~~

~~$i := 1$~~

~~$j := 1$~~

~~while $A_1[i] \neq A_2[j] \wedge j \leq A_2.\text{Länge} \wedge i \leq A_1.\text{Länge}$~~

~~if $A_1[i] < A_2[j]$~~

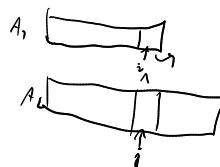
~~$i := i + 1$~~

~~else if $A_2[j] < A_1[i]$~~

~~$j := j + 1$~~

~~end if~~

~~end while~~



3) (7 P.) Ein *gerichteter Zyklus* in einem gerichteten Graphen $G = (V, E)$ ist ein Pfad der Form $v_1, v_2, \dots, v_{k-1}, v_k = v_1$ mit $k \geq 2$ so dass für alle $i, j \in \{1, \dots, k-1\}$ mit $i \neq j$ auch $v_i \neq v_j$ ist. Geben Sie einen Algorithmus in Pseudocode an, der als Eingabe einen endlichen gerichteten Graphen $G = (V, E)$ sowie ein $s \in V$ erhält und feststellt ob s in G auf einem gerichteten Zyklus liegt. Der Eingabgraph wird als Adjazenzliste übergeben. Die Laufzeit des Algorithmus muss $O(|V| + |E|)$ sein.

Prozedur ZYKLUS(V, E, s)

Sei A neues Datenfeld der Größe $|V|$ überall mit falsch initialisiert

Sei Q leere Warteschlange

HINZUFÜGEN(Q, s)

$A(s) := \text{wahr}$

while $Q \neq \emptyset$

$v := \text{ENTFERNEN}(Q)$

for alle $(v, w) \in E$

if $w = s$ und v .Vorgänger $\neq s$

return "zyklus"

end if

if $A(w) \neq \text{wahr}$

w .Vorgänger $:= v$

$A(w) := \text{wahr}$

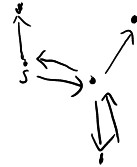
HINZUFÜGEN(Q, w)

end if

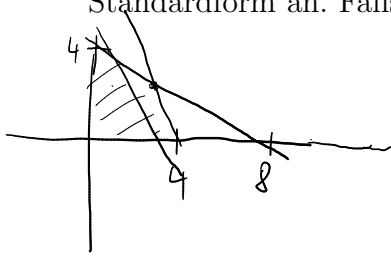
end for

end while

return "kein zyklus"



4) (5 P.) Wenden Sie den Simplex-Algorithmus auf das folgende lineare Programm in Standardform an. Falls das Programm eine optimale Lösung hat geben Sie diese an.



$$z = 2x_1 + x_2 \max!$$

$$x_1 + 2x_2 \leq 8 \leadsto 8 - x_1 - 2x_2 = x_3$$

$$\wedge x_3 \geq 0$$

$$3x_1 + x_2 \leq 12 \leadsto 12 - 3x_1 - x_2 = x_4$$

$$\wedge x_4 \geq 0$$

Basislösung: $(0, 0, 8, 12)$, also $z = 0$

aus x_3 -Gleichung $x_1 \leq 8$ und aus x_4 -Glp.: $x_1 \leq 4$

also tausche x_1 und x_4 :

$$x_1 = (12 - x_2 - x_4) \frac{1}{3} = 4 - \frac{1}{3}x_2 - \frac{1}{3}x_4$$

$$x_3 = 8 - (4 - \frac{1}{3}x_2 - \frac{1}{3}x_4) - 2x_2 = 4 - \frac{5}{3}x_2 + \frac{1}{3}x_4$$

$$z = 2(4 - \frac{1}{3}x_2 - \frac{1}{3}x_4) + x_2 = 8 + \frac{1}{3}x_2 - \frac{2}{3}x_4$$

Basislösung $(4, 0, 4, 0)$

$$z = 8$$

aus x_3 -Glp.: $x_2 \leq 12$, aus x_4 -Glp.: $x_2 \leq \frac{12}{5}$

also tausche x_2 und x_4 :

$$x_2 = \frac{3}{5}(4 + \frac{1}{3}x_4 - x_3) = \frac{12}{5} + \frac{1}{5}x_4 - \frac{3}{5}x_3$$

$$x_1 = 4 - \frac{1}{3}(\frac{12}{5} + \frac{1}{5}x_4 - \frac{3}{5}x_3) - \frac{1}{3}x_4 = \frac{16}{5} - \frac{6}{15}x_4 + \frac{1}{5}x_3$$

$$z = 8 + \frac{1}{3}(\frac{12}{5} + \frac{1}{5}x_4 - \frac{3}{5}x_3) - \frac{2}{3}x_4 = \frac{44}{5} - \frac{2}{15}x_4 - \frac{1}{5}x_3$$

Basislösung $(\frac{16}{5}, \frac{12}{5}, 0, 0)$

$z = \frac{44}{5}$: optimale Lsg.!

Prüfung aus

Diskrete und geometrische Algorithmen (Hetzl)

Zuname: _____ Matrikelnummer: _____

Vorname: _____

Titelseite ab hier bitte freilassen!

TU Wien, 2.7.2019

Arbeitszeit: 100 Minuten

- 1)
- 2)
- 3)
- 4)

1) (6 P.) Lösen Sie die Rekursionsgleichung $a_n = 4a_{n-1} - 3a_{n-2}$ (für $n \geq 2$) mit den Anfangswerten $a_0 = 1$ und $a_1 = 2$.

2) (6 P.) Geben Sie einen Algorithmus in Pseudocode an, der als Eingabe ein aufsteigend sortiertes Datenfeld A paarweise verschiedener ganzer Zahlen erhält und feststellt ob ein $i \in \{1, \dots, n\}$ existiert mit $A[i] = i$. Die Laufzeit des Algorithmus muss $O(\log n)$ sein wobei n die Anzahl der Elemente in A ist.

3) (6 P.) Seien B_1 und B_2 Suchbäume die die selbe Menge von Einträgen enthalten. Gibt es eine Folge von Links- und Rechtsrotationen mit denen B_1 in B_2 transformiert werden kann? Beweisen Sie die Korrektheit Ihrer Antwort.

4) (6 P.) Geben Sie einen Algorithmus in Pseudocode an, der als Eingabe ein Datenfeld A paarweise verschiedener Elemente sowie ein $k \in \{1, \dots, A.L\ddot{a}nge\}$ erhlt und als Ausgabe eine zufllige k -elementige Teilmenge von A als Datenfeld liefert. Die Laufzeit des Algorithmus muss $O(k)$ sein.

Sie drfen dazu die Existenz einer Prozedur $ZUFALL(i, j)$ voraussetzen, die fr $i, j \in \mathbb{N}$ mit $i \leq j$ in Zeit $O(1)$ eine im Intervall $[i, j] \subseteq \mathbb{N}$ uniform verteilte Zufallszahl liefert.

Prüfung aus

Diskrete und geometrische Algorithmen (Hetzl)

Zuname: _____ Matrikelnummer: _____

Vorname: _____

Titelseite ab hier bitte freilassen!

TU Wien, 4.10.2019

Arbeitszeit: 100 Minuten

1)

2)

3)

4)

1) (6 P.) Lösen Sie die Rekursionsgleichung $a_n = 6a_{n-1} - 5a_{n-2}$ (für $n \geq 2$) mit den Anfangswerten $a_0 = 0$ und $a_1 = 1$.

2) (6 P.) Geben Sie einen Algorithmus in Pseudocode an, der als Eingabe ein aufsteigend sortiertes Datenfeld A paarweise verschiedener ganzer Zahlen erhält und feststellt ob ein $i \in \{1, \dots, n\}$ existiert mit $A[i] = i$. Die Laufzeit des Algorithmus muss $O(\log n)$ sein wobei n die Anzahl der Elemente in A ist.

3) (7 P.) Sei

$$A = (a_{i,j})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq n}}$$

eine Matrix. Als *Untermatrix* von A bezeichnen wir eine Matrix der Form

$$(a_{i,j})_{\substack{p \leq i \leq q \\ r \leq j \leq s}}$$

wobei $p, r \geq 1$, $q \leq m$ und $s \leq n$.

Geben Sie einen Algorithmus in Pseudocode an der eine Matrix $A \in \{0, 1\}^{m \times n}$ als Eingabe erhält und der als Ausgabe das größte $k \in \mathbb{N}$ zurückliefert so dass A eine $k \times k$ Untermatrix enthält die nur aus Einsen besteht. Die Laufzeit des Algorithmus muss $O(m \cdot n)$ sein.

Hinweis: Dynamische Programmierung, betrachten Sie $m_{q,s}$, die maximale Größe einer quadratischen Einser-Matrix die an der Stelle (q, s) endet.

4) (5 P.) Wenden Sie den Simplex-Algorithmus auf das folgende lineare Programm in Standardform an. Falls das Programm eine optimale Lösung hat geben Sie diese an.

$$z = x_1 + 2x_2 \text{ max!}$$

$$2x_1 + x_2 \leq 8$$

$$x_1 + 3x_2 \leq 12$$