

AlgoDat 2 - Prüfungsausarbeitung

20041004.....	2
Aufgabe 2.A: Scanline	2
Aufgabe 4.A: Bereichssuche	4
20060126.....	5
Aufgabe 1.A: Preflow-Push Algorithmus	5
Aufgabe 4.A: Geometrische Algorithmen	6
Aufgabe 5.A: Algorithmen für große Datenmengen	7
20070125.....	8
Aufgabe 1.A: Textsuche.....	8
Aufgabe 2.A: Schnitt von allgemeinen Liniensegmenten.....	8
Aufgabe 3.A: Simplex-Verfahren	10
Aufgabe 4.A: Preflow-Push Algorithmus	11
Aufgabe 5.A: Verschiedenes.....	13
Aufgabe 1.B: Simplex-Verfahren	14
Aufgabe 2.B: Preflow-Push Algorithmus	15
Aufgabe 3.B: Verschiedenes	17
Aufgabe 4.B: Textsuche	18
Aufgabe 5.B: Schnitt von allgemeinen Liniensegmenten	19
20070309.....	20
Aufgabe 1.A: Textsuche.....	20
Aufgabe 2.A: Skiplisten	21
Aufgabe 3.A: Simplex-Verfahren	21
Aufgabe 4.A: Fluss-Algorithmen.....	23
Aufgabe 5.A: Verschiedenes.....	25

Hinweis

Für Richtigkeit und Vollständigkeit der Lösungen wird keine Haftung übernommen!
Einige Beispiele aus älteren Prüfungsangaben wurden absichtlich ausgelassen, da diese im neuen Modus eher nicht kommen (z.B. Pseudo-Code Beispiele)

Falls Fehler oder Ergänzungen gefunden werden, bitte schreibt sie mir per Mail oder PM

Viel Erfolg - hoffe das hilft euch,

stk

(Informatik-Forum Name)

Hilfreiche Threads:

Zur Ausarbeitung

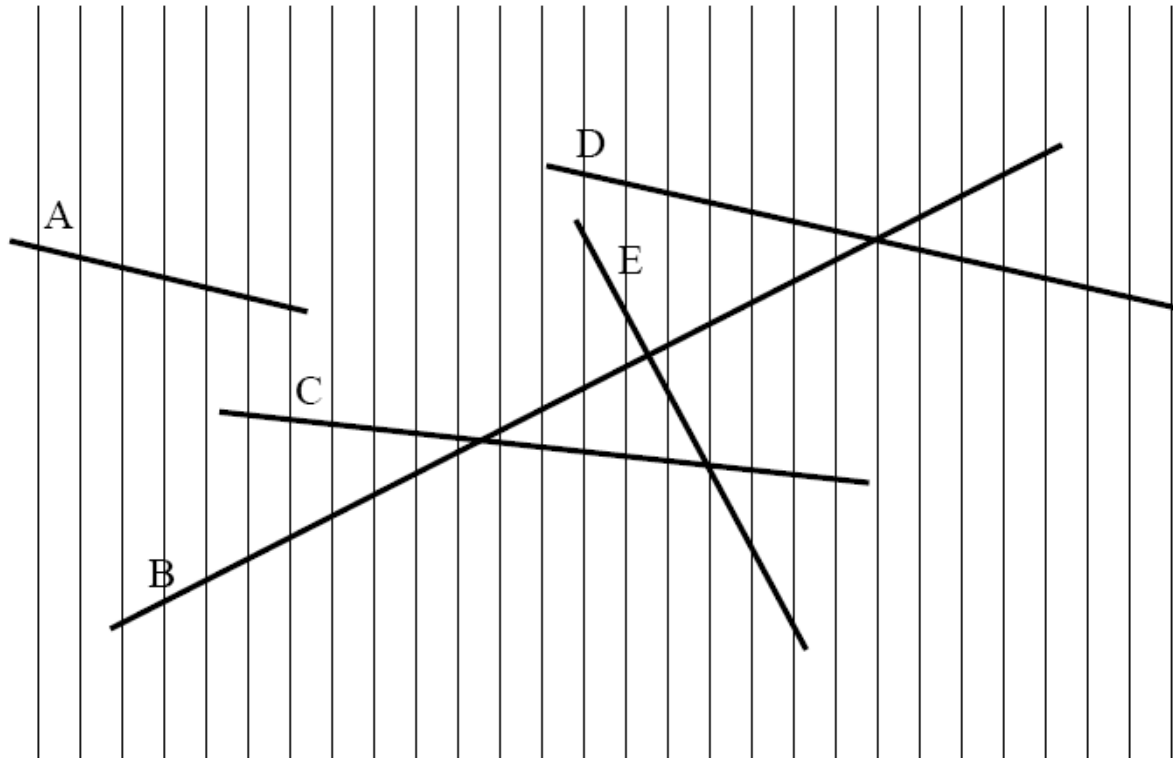
<http://www.informatik-forum.at/showthread.php?p=442148>

Blends Regel:

<http://www.informatik-forum.at/showthread.php?t=52499>

20041004

Aufgabe 2.A: Scanline



a) 6 Punkte

Gegeben sind die Geradensegmente *A*, *B*, *C*, *D* und *E* in der obigen Zeichnung (die vertikalen Segmente sollen Ihnen nur zur Orientierung dienen). Führen Sie den Algorithmus zum Schnitt von allgemeinen Liniensegmenten auf diesen Segmenten durch. Geben Sie bei jedem Haltepunkt an, was passiert. Geben Sie für jeden Haltepunkt den Inhalt sowohl der Ereignis-Datenstruktur als auch der Scanline-Datenstruktur an.

Initialisiere ES mit den Anfangs u. Endpunkten der Segmente nach X-Koordinate sortiert. (A...Anfangspunkt von A, A'...Endpunkt von A)

SSS ist nach der Y-Koordinate mit dem Schnittpunkt der Scanline sortiert

SSS	ES	Nächster Schritt
leer	A,B,C,A',D,E,E',C',B',D'	A in SSS, entfernen aus ES
A	B,C,A',D,E,E',C',B',D'	B in SSS, entfernen aus ES (in SSS B vor A weil y kleiner!), prüfe Schnitt->0
B,A	C,A',D,E,E',C',B',D'	C in SSS, entfernen aus ES, prüfe im vorgänger(B) und nachfolger(A)-> schnitt BC in ES
B,C,A	A',BC, D,E,E',C',B',D'	A' entfernen aus ES, prüfe schnitt vorg(A) nachf(A)-> 0

B,C	BC, D,E,E',C',B',D'	BC entfernen aus ES, in SSS vertauschen, auf Schnitt prüfen->0
C,B	D,E,E',C',B',D'	D in SSS, entfernen aus ES, vorg u. nachf auf schnitt prüfen-> BD in ES
C,B,D	E,E',C',BD,B',D'	E in SSS, entfernen aus ES, vorg u. nachf auf schnitt prüfen-> EB in ES
C,B,E,D	EB,E',C',BD,B',D'	EB entfernen aus ES, in SSS vertauschen, E mit vorg u. B mit nachf auf Schnitt prüfen->CE in ES BD Schon vorhanden! Nicht noch einmal einfügen! Ergänzung im Skript folgt
C,E,B,D	CE,E',C',BD,B',D'	CE entfernen aus ES, in SSS vertauschen, Schnitt prüfen-> BC liegt hinter scanline, nicht einfügen
E,C,B,D	E',C',BD,B',D'	E' entfernen aus ES, E aus SSS entfernen, vorg u. nachf von E auf schnitt testen->0
C,B,D	C',BD,B',D'	C' entfernen aus ES, C aus SSS entfernen, vorg u. nachf von C auf schnitt testen ->0
B,D	BD,B',D'	BD entfernen aus ES, in SSS vertauschen, schnitt prüfen -> 0
D,B	B',D'	B' entfernen aus ES, aus SSS entfernen, vorg u. nachf von B auf schnitt prüfen->0
D	D'	D' entfernen aus ES, aus SSS entfernen, vorg u. nachf von D auf schnitt prüfen->0

b) 7 Punkte

Was versteht man unter "allgemeiner Lage"? Was genau könnte im schlimmsten Fall bei dem Pseudocode aus der Vorlesung passieren, wenn die Geradensegmente nicht allgemeine Lage haben? Beschreiben Sie die Lage der Segmente in diesen Fällen und wie der Algorithmus darauf reagiert.

Hinweis: der folgende Teil wurde von einem Dokument aus dem Forum entnommen und soll zum besseren Verständnis der Bedingungen helfen.

Autor: Martin Pickelbauer

Ich hoffe ich verletze keine Copyrights...falls doch bitte melden ☺

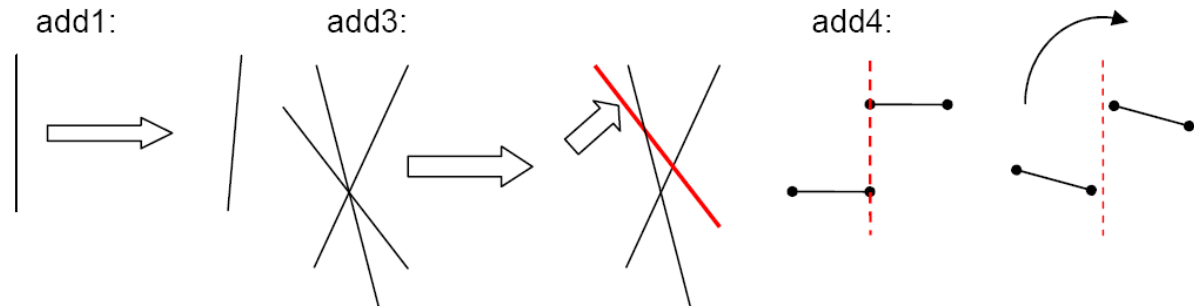
1. Kein Segment ist senkrecht
2. Der Schnitt zweier Segmente ist stets leer oder genau ein Punkt
3. Es schneiden sich nie mehr als zwei Segmente in einem Punkt
4. Alle Endpunkte und Schnittpunkte haben paarweise verschiedene x-Koordinaten.

Für unsere Fragestellung müssen wir für den dritten Punkt der Vereinfachungen eine Lösung finden.

add 1: die Ebene leicht drehen

add 3: ein Segment leicht verschieben

add 4: die Ebene wieder leicht drehen



Add 2: würde ich so interpretieren: 2 segmente dürfen nicht (teilweise) übereinander liegen

Bsp: _____

Verhalten des Algorithmus:

Aufgabe 4.A: Bereichssuche

Gegeben ist die folgende Punktmenge in 3D:

$P_0 = (40, 23, 40)$, $P_1 = (32, 41, 41)$, $P_2 = (102, 40, 11)$,
 $P_3 = (11, 11, 49)$, $P_4 = (27, 76, 23)$, $P_5 = (76, 32, 68)$, $P_6 = (49, 49, 27)$,
 $P_7 = (23, 68, 32)$, $P_8 = (41, 27, 102)$, $P_9 = (68, 102, 76)$

a) 6 Punkte

Zeichnen Sie einen balancierten Baum für 3-dimensionale Bereichssuche dessen Knoten den angegebenen

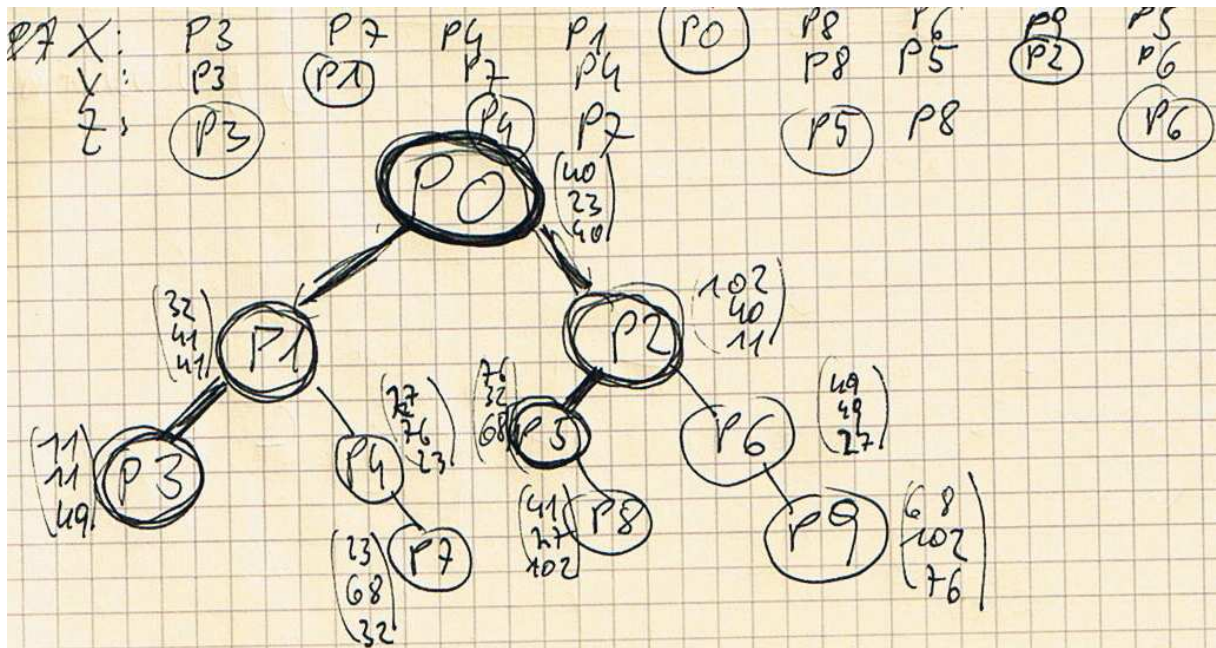
Punkten entsprechen. Achten Sie dabei auf folgende Punkte:

- Fangen Sie mit der x-Koordinate an, dann y und schließlich z.
- Beschriften Sie jeden Knoten mit der Nummer des entsprechenden Punktes aus P .
- Geben Sie für jeden Knoten des Baums die entsprechenden x-, y, und z-Folgen an.
- Der Median einer Folge mit den indizes a_1, \dots, a_k ist a_m mit

$$m = \lfloor \frac{k+1}{2} \rfloor$$

b) 6 Punkte

Markieren Sie alle Knoten in Ihrem Baum, die bei der Bereichssuche nach dem Bereich D vom Bereichssuche-Algorithmus besucht werden. Der Bereich D ist ein 3-dimensionaler Quader gegeben durch die beiden Eckpunkte $d_1 = (30, 20, 45)$ und $d_2 = (50, 39, 55)$. Welche Punkte liegen in dem Bereich?



Besuchte Knoten: P0, P1, P2, P3, P5

Knoten im Bereich: keiner

Warum:

Ich beginne bei P0, vergleiche die X-Koordinate mit den angegebenen Punkten: $30 < 40(p_0) < 50 \Rightarrow$ bedeutet ich muss beide kinder untersuchen
 P1 vergleich mit y: $20 < 39 < 41(p_1) \Rightarrow$ gehe nach links zu p3, weil rechts davon nur größere y werte sind und dieser schon drüber liegt
 P3 hat keine kinder mehr...
 P2 vergleich mit y: $20 < 39 < 40(p_2) \Rightarrow$ wieder links
 P5 vergleich mit z: $45 < 55 < 68(p_5) \Rightarrow$ links keine kinder, daher ende

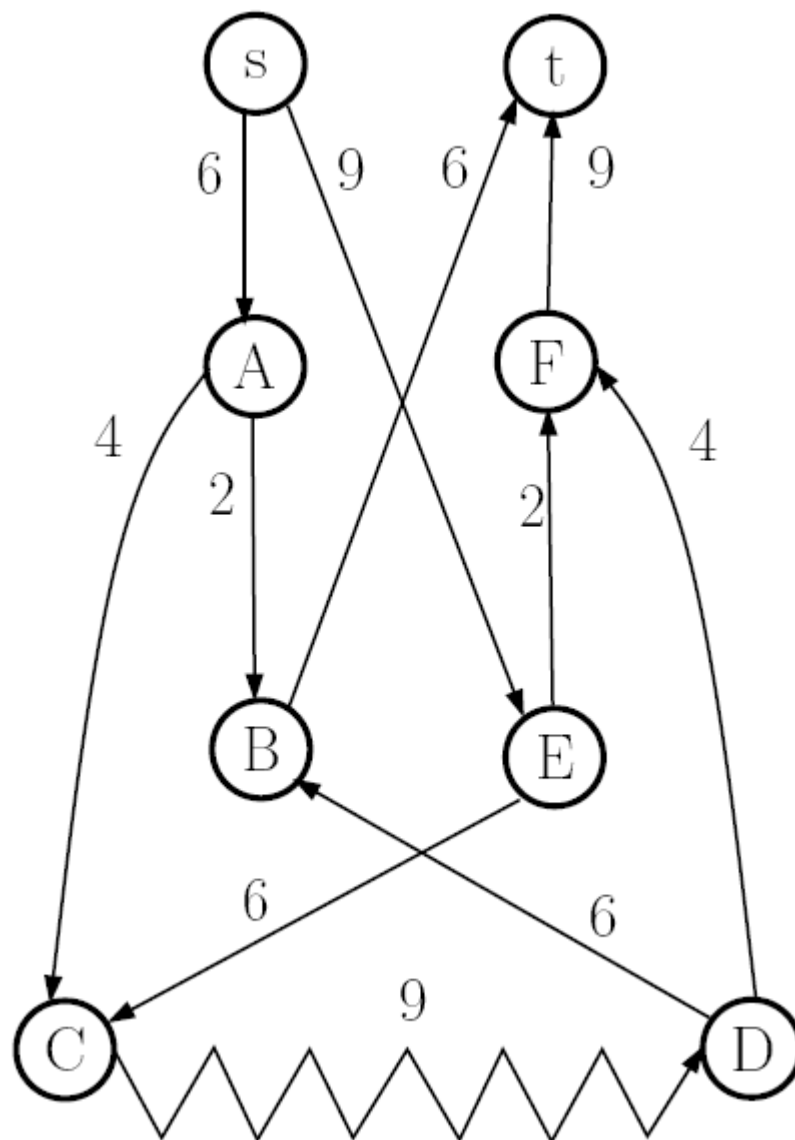
Bei jedem besuchten knoten überprüfe ich, ob er im bereich liegt -> keiner liegt im bereich.

20060126

Aufgabe 1.A: Preflow-Push Algorithmus

a) (5 Punkte)

Finden Sie zu dem nachfolgenden Flussnetzwerk N den maximalen Fluss f_{\max} von der Quelle s zur Senke t , indem Sie den Preflow-Push Algorithmus verwenden.



- a) Geben Sie für jeden Schritt des Algorithmus den "Überschuss und die Höhe (Beschriftung) d der sich ändernden Knoten an.
 b) Geben Sie anschließend einen Schnitt in N an, der von f^* saturiert wird.

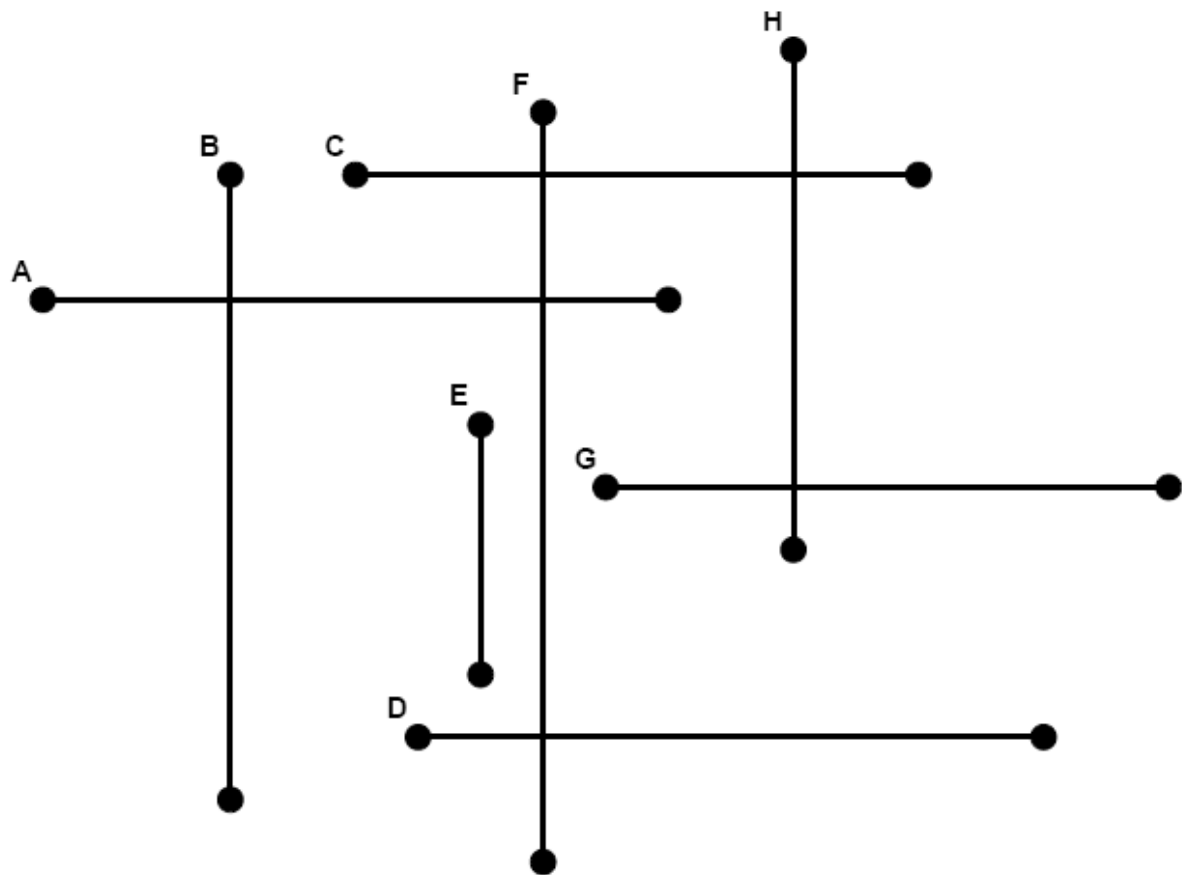
b) (1 Punkt)

Im Preflow-Push Algorithmus gibt es *lift*-Operationen und einen Preflow f .
 Beweisen

oder widerlegen Sie: Nach einer *lift*-Operation ist die Höhe (Beschriftung) d legal und f bleibt ein Preflow.

Aufgabe 4.A: Geometrische Algorithmen

Führen Sie den aus der Vorlesung bekannten Algorithmus zum Schnitt von isoorientierten Liniensegmenten für die folgende Menge von Linien aus. Geben Sie dabei für jeden relevanten Punkt im Verlauf der Bewegung der Scan-Linie den Zustand der Ereignis-Datenstruktur sowie die eventuell gefundenen Schnittpunkte an.



Aufgabe 5.A: Algorithmen für große Datenmengen

a) (2 Punkte)

Skizzieren Sie das hierarchische Speichermodell typischer moderner Computer. Beschriften und beschreiben Sie es in kurzen Worten.

CPU – Cache – RAM – Disk(secondary memory)

Ram Zugriff: um Faktor 100 langsamer

Disk Zugriff: um Faktor 100 000 langsamer

b) (2 Punkte)

Die Funktionsfähigkeit eines Cache-Speichers hängt wesentlich von der Lokalität der eingesetzten Programme ab. Welche beiden Formen der Lokalität gibt es, und wie können diese jeweils vom Cache ausgenutzt werden?

-) **Zeitliche Lokalität:** gleiche Speicherbereiche werden innerhalb einer kurzen Zeitspanne angesprochen. Caches nutzen dies aus indem sie versuchen, die angesprochenen Speicherbereiche möglichst lange im Cache zu lassen.

-) **Örtliche Lokalität:** es werden öfter Speicherbereiche angesprochen, die nahe beieinander liegen. Caches nutzen dies aus, indem sie bei jedem Miss jeweils ganze Speicherblöcke in den Cache holen.

c) (2 Punkte)

Erläutern Sie in kurzen Worten den Unterschied zwischen Algorithmen, die sich *cache-aware* bzw. *cache-oblivious* verhalten. Welches Verhalten ist im allgemeinen Fall vorzuziehen, und warum?

Cache-aware: der Algorithmus enthält parameter mit denen die Cache-Komplexität für die gegebene Cachegröße Z und Zeilenlänge L optimiert werden kann. Sonst heißt er **cache-oblivious**.
Im Allgemeinen Fall ist cache-oblivious vorzuziehen, da dieser Algorithmus für unbekanntes Z und L optimiert wurde und dadurch automatisch Zugriffe in jeder Ebene optimiert.

20070125

Aufgabe 1.A: Textsuche

Betrachten Sie den Algorithmus von *Boyer-Moore* (BM), den Text $T = \text{BAAANABANANA}$ und das Muster $P = \text{BANANA}$.

a) (2 Punkte)

Geben Sie das *last* [] Array an.

b) (4 Punkte)

Geben Sie das *suffix*[] Array für P an.

c) (4 Punkte)

Suchen Sie mittels BM in T nach P . Visualisieren Sie in jedem Schritt, wo das

Muster angelegt wird. Geben Sie an

- welche Zeichen gematcht werden,
- welches Zeichen gegebenenfalls den Mismatch verursacht, und
- mit welcher Regel (*last* oder *suffix*) verschoben wird.

(Wenn Sie die Suche nur mittels *last* [] Array oder nur mittels *suffix*[] Array durchführen, wird ein Punkt abgezogen.)

a)

last[] = B=1 A=6 N=5

b)

next[] = 0 0 0 0 0 0

$P^{-1} = \text{ANANAB}$, $\text{next}^{-1}[] = 0 0 1 2 3 0$

suffix[] = M-next[M] = 6 6 6 6 6 6

$q \dots 1-M$

$j = M - \text{next}^{-1}[q] = 6 6 5 4 3 6$

$h = q - \text{next}^{-1}[q] = 1 2 2 2 2 6$

***suffix*[] = 6 6 2 2 2 1**

c)

BA**A**ANABANANA

BANANA => $j = 6 - 3$ (gematcht von rechts) = 3; $3 - \text{last}[a] = -3$; *suffix*[3]=2, das -größere von beiden wird genommen, also *suffix*-verschiebung um 2...

BANANA => $j = 5$; $5 - \text{last}[b] = 4$; *suffix*[5]=2 => *last*-verschiebung um 4

BANANA => Match, fertig

Aufgabe 2.A: Schnitt von allgemeinen Liniensegmenten

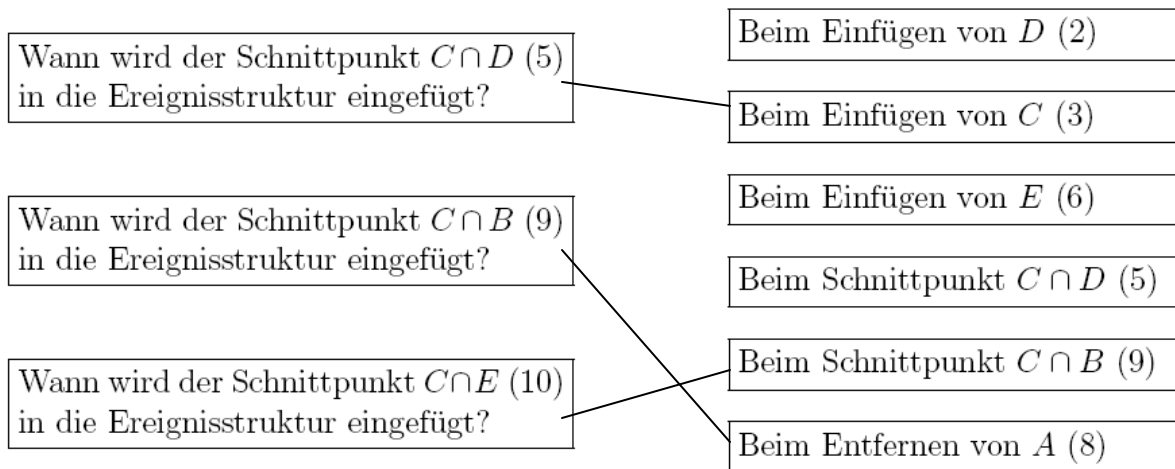
a) (7 Punkte)

Führen Sie den aus der Vorlesung bekannten Algorithmus zum Schnitt von allgemeinen Liniensegmenten für die folgende Menge von Linien aus. Geben Sie dabei für jeden Zeitpunkt (0 . . . 13) im Verlauf der Bewegung der Scan-Line den Zustand r Scan-Line-Status Struktur an.

(Die vertikalen Linien dienen nur zur Orientierung)

b) (3 Punkte)

Verbinden Sie die folgenden Fragen mit den richtigen Antworten



Aufgabe 3.A: Simplex-Verfahren

Gegeben ist das folgende lineare Programm (LP):

max $x_1 + 3x_2$

s.t. $-x_1 + 2x_2 \leq 6$

$x_1 + x_2 \leq 8$

$x_1, x_2 \geq 0$

a) (5 Punkte)

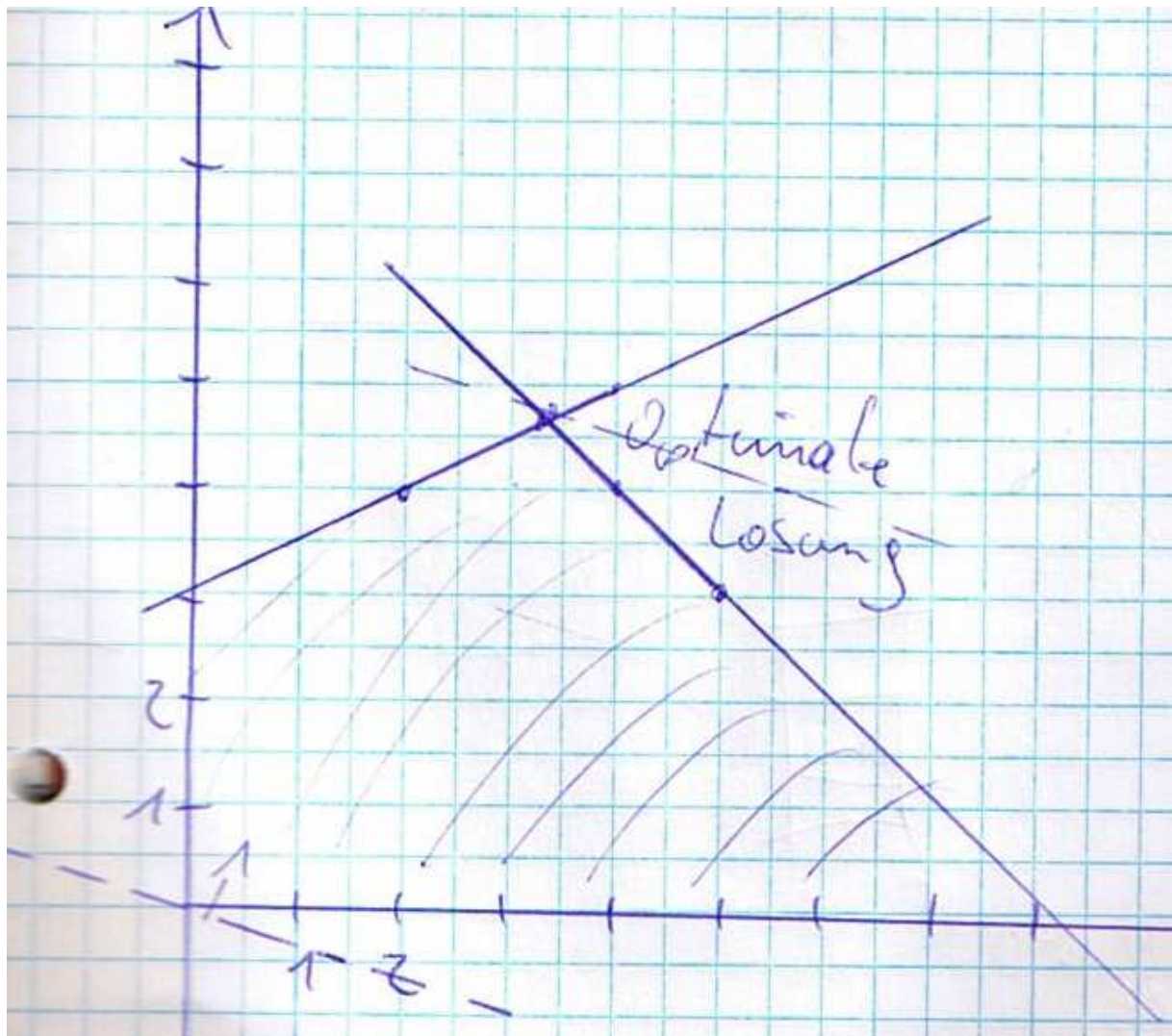
Lösen Sie das LP grafisch:

- Zeichnen Sie den zulässigen Lösungsbereich ein.
- Markieren Sie die optimale Lösung.

b) (5 Punkte)

Schreiben Sie das LP als Simplextableau auf und führen Sie die erste Simplex-Iteration aus. (D.h., das LP braucht nicht vollständig gelöst zu werden.)

a)



b)

Simplextableau

$$Z = x_1 + 3x_2$$

$$x_3 = 6 + x_1 - 2x_2$$

$$x_4 = 8 - x_1 - x_2$$

1. iteration, x_1, x_4

$$Z = 2x_2 - x_4 + 8$$

$$x_1 = -x_2 - x_4 + 8$$

$$x_3 = -3x_2 - x_4 + 14$$

x_2 einschränkungen:

bei $x_1 = 8$

Bei $x_3 = 14/3 \Rightarrow$ größere Einschränkung!

2. iteration, x_2, x_3

$$Z = -2/3x_3 - 5/3x_4 + 52/3$$

$$x_1 = x_3/3 - 2/3x_4 + 10/3$$

$$x_2 = -x_3/3 - x_4/3 + 14/3$$

Lösung $(10/3, 14/3)$ Wert $52/3$

Aufgabe 4.A: Preflow-Push Algorithmus

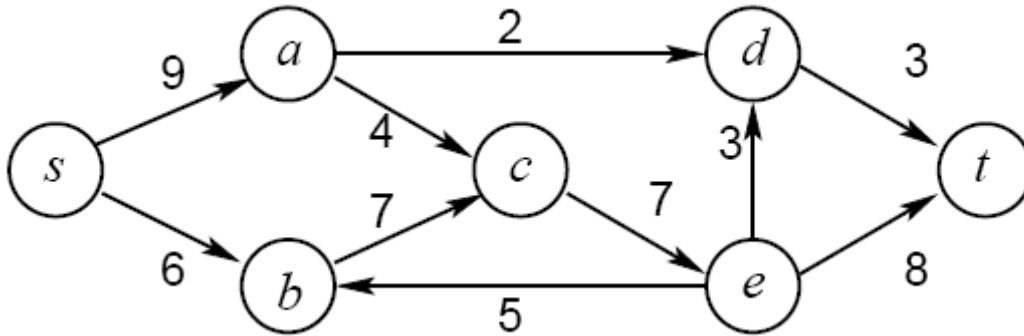
a) (3 Punkte)

Welche drei Eigenschaften muss eine Funktion $f : V \times V \rightarrow \mathbb{R}$ erfüllen, sodass Sie einen gültigen Preflow in einem Netzwerk $N = (V, E, c, s, t)$ mit Knotenmenge V darstellt?

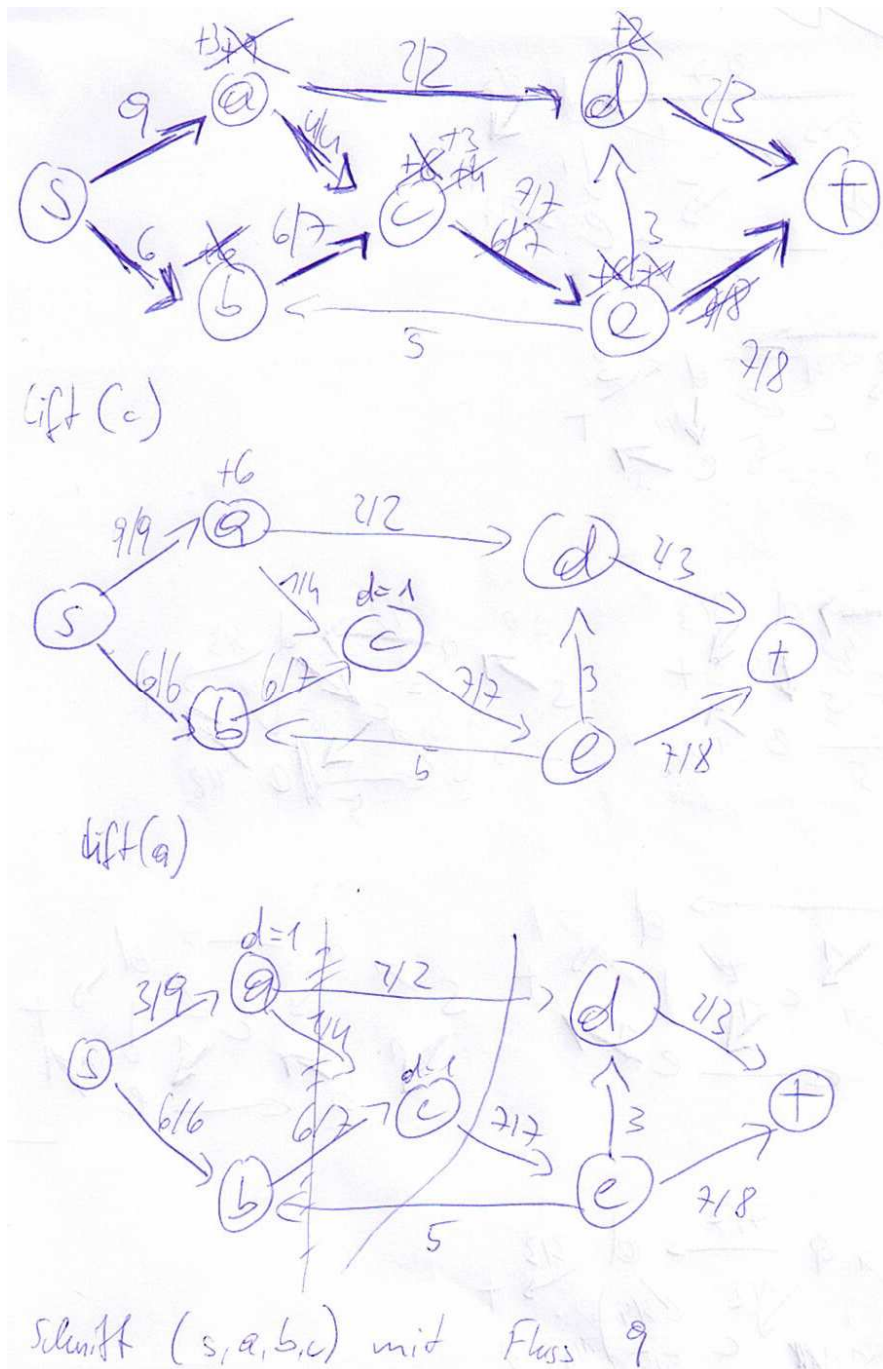
- 1) $f(u,v) = -f(v,u)$ Schiefsymmetrie
- 2) $f(u,v) \leq c(u,v)$ Kapazitätsbeschränkung
- 3) $ef(v) = \sum (f(u,v)) \geq 0$, $u \in V$, $v \in V \setminus \{s\}$ Überschussbedingung

b) (7 Punkte)

Finden Sie zu dem nachfolgenden Flussnetzwerk N den maximalen Fluss f^* von der Quelle s zur Senke t , indem Sie den Preflow-Push Algorithmus anwenden. Benutzen Sie hierfür die Templates auf der nächsten Seite.



- a) Geben Sie für jeden *lift*-Schritt des Algorithmus den "Überschuss und die Höhe (Beschriftung) d der sich ändernden Knoten an.
- b) Falls am Ende ein "Überschuss in den inneren Knoten übrig bleiben sollte, beschreiben Sie in wenigen Worten, wie der Algorithmus darauf reagiert. (D.h., Sie müssen den "Überschussabbau nicht schrittweise durchrechnen.)
- c) Geben Sie abschließend einen Schnitt in N an, der von f^* saturiert wird.



Aufgabe 5.A: Verschiedenes

a) (3 Punkte)

Beschreiben Sie in einfachen Worten, was das Pricing-Problem in einem Spaltengenerierungsverfahren ist.

b) (3 Punkte)

Beschreiben Sie in einfachen Worten, was Branch-and-Cut von einfachem LPbasierten Branch-and-Bound unterscheidet.

Dieses verfahren verbindet das branch-and-bound verfahren mit dem schnittebenenverfahren.

beim branch-and-cut verfahren wird das schnittebenenverfahren zum lösen der teilprobleme verwendet. Sollte dies keine lösung bringen, wird weiter aufgeteilt(branching)

c) (2 Punkte)

Was ist der Unterschied zwischen einer *perfekten Skipliste* und einer *randomisierten Skipliste*? Welche Vorteile besitzt die bessere der beiden Datenstrukturen?

Bei einer perfekten skipliste ist die höhe der container fix.

Bei der randomisierten werden die höhen durch zufall ermittelt.

Der vorteil der randomisierten skipliste ist, dass beim einfügen und löschen nie reorganisiert werden muss sondern nur zeiger verbogen werden müssen(weil es keine vorgabe für die höhe der container an bestimmten plätzen gibt)

d) (2 Punkte)

Randomisierter Primzahltest von Miller-Rabin:

- Welche Aussage hat das Ergebnis, wenn a ein Zeuge für die Nicht-Primheit von n ist?

N ist sicher keine Primzahl

- Welche Aussage hat das Ergebnis, wenn a kein Zeuge für die Nicht-Primheit von n ist?

N könnte eine Primzahl sein

Aufgabe 1.B: Simplex-Verfahren

Gegeben ist das folgende lineare Programm (LP):

max $-x_1 + 2x_2$

s.t $x_1 + 2x_2 \leq 7$

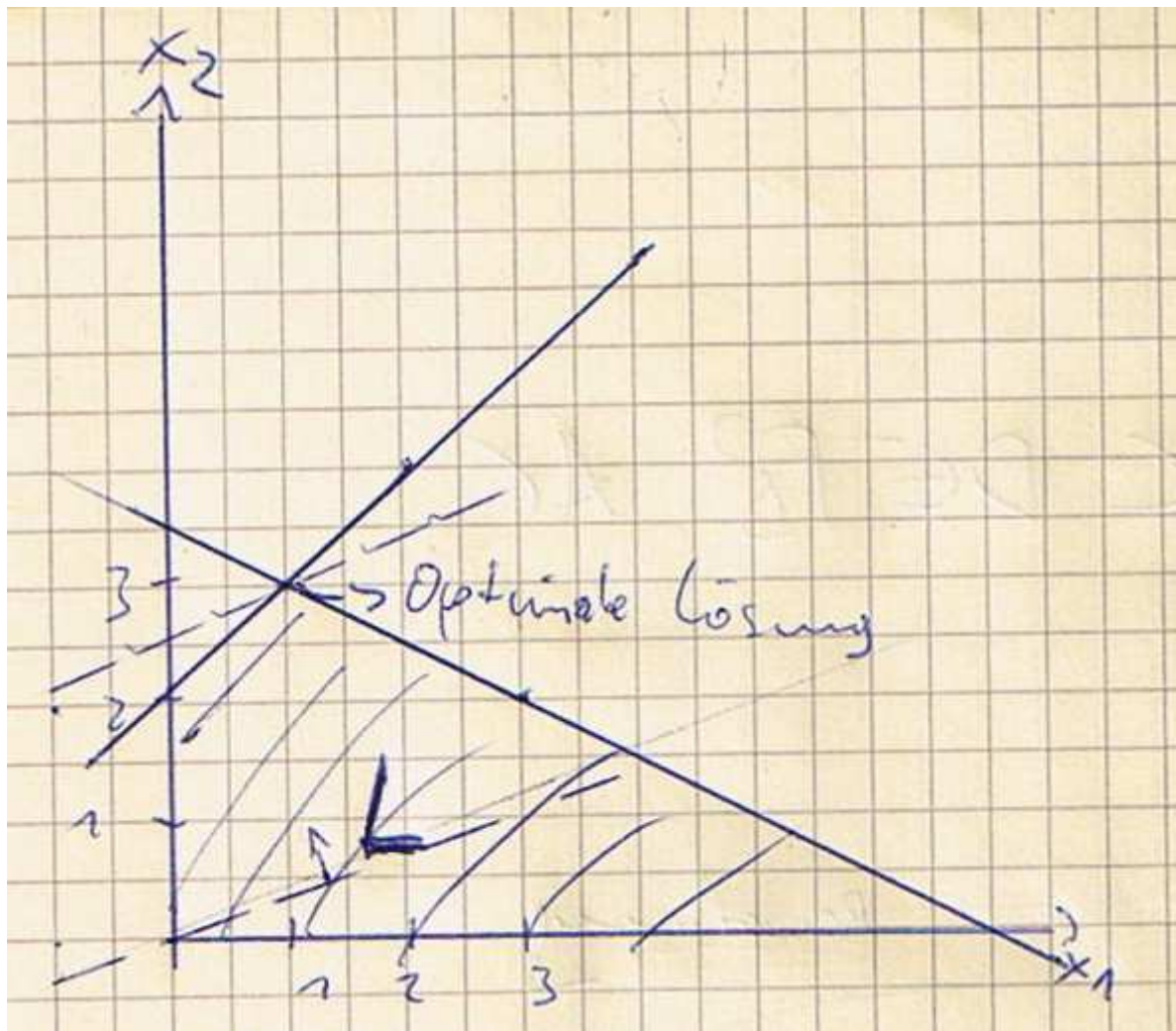
$-x_1 + x_2 \leq 2$

$x_1, x_2 \geq 0$

a) (5 Punkte)

Lösen Sie das LP grafisch:

- Zeichnen Sie den zulässigen Lösungsbereich ein.
- Markieren Sie die optimale Lösung.



b) (5 Punkte)

Schreiben Sie das LP als Simplextableau auf und führen Sie die *erste* Simplex-Iteration aus. (D.h., das LP braucht nicht vollständig gelöst zu werden.)

Simplextableau:

$$Z = -x_1 + 2x_2$$

$$x_3 = -x_1 - 2x_2 + 7$$

$$x_4 = x_1 - x_2 + 2$$

Einschränkung für x_2

$$x_3 = 3.5$$

$x_4 = 2 \Rightarrow$ größere Einschränkung

1. Iteration: x_2 kommt in die basis, x_4 verlässt diese

$$Z = x_1 - 2x_4 + 4$$

$$x_3 = -3x_1 + 2x_4 + 3$$

$$x_2 = x_1 - x_4 + 2$$

2. x_1, x_3

$$Z = -x_3/3 - 4/3x_4 + 5$$

$$x_1 = 2/3x_4 + 1 - x_3/3$$

$$x_2 = -x_4/3 + 3 - x_3/3$$

Lösung(1,3), wert 5

Aufgabe 2.B: Preflow-Push Algorithmus

a) (3 Punkte)

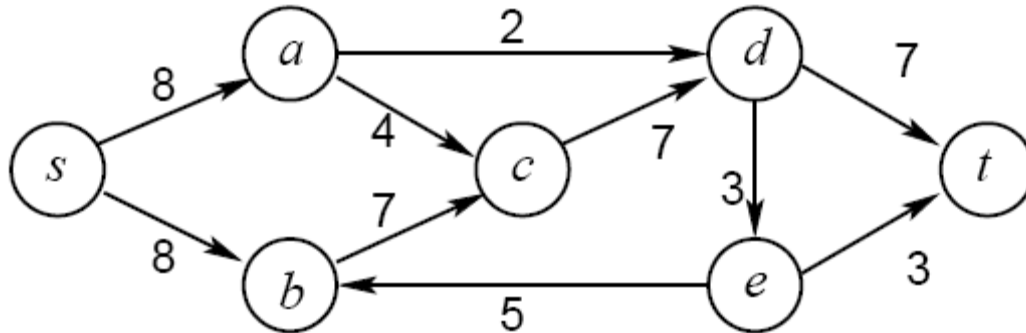
Welche drei Eigenschaften muss eine Funktion $f : V \times V \rightarrow \mathbb{R}$ erfüllen, sodass Sie einen gültigen Fluss in einem Netzwerk $N = (V, E, c, s, t)$ mit Knotenmenge V darstellt?

- 1) $f(u,v) = -f(v,u), \quad u,v \in V$
- 2) $f(u,v) \leq c(u,v), \quad u,v \in V$
- 3) $\sum_{v \in V} f(u,v) = 0, \quad u \in V \setminus \{s,t\}$

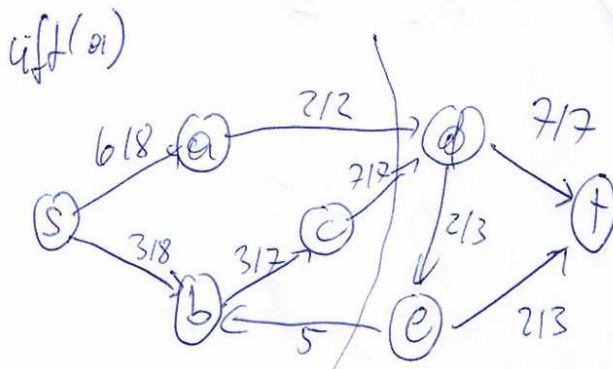
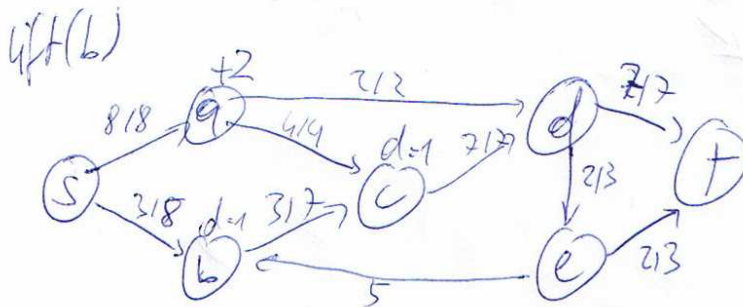
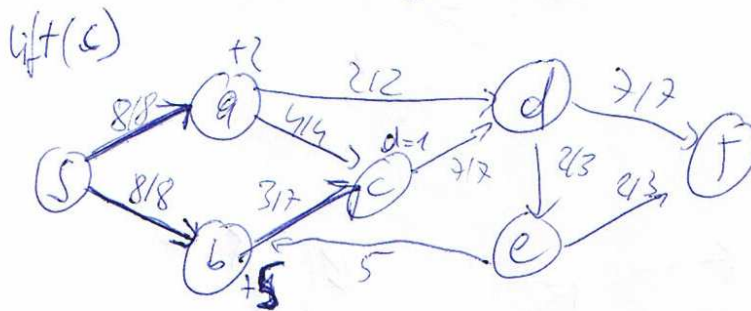
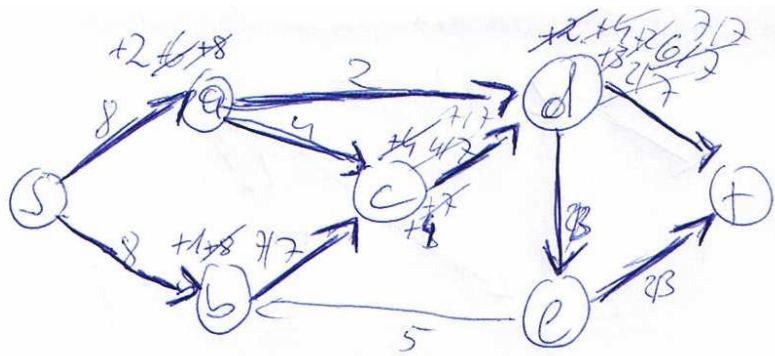
Schiefssymmetrie
Kapazitätsbeschränkung
Flusserhaltung

b) (7 Punkte)

Finden Sie zu dem nachfolgenden Flussnetzwerk N den maximalen Fluss f^* von der Quelle s zur Senke t , indem Sie den Preflow-Push Algorithmus anwenden. Benutzen Sie hierfür die Templates auf der nächsten Seite.



- a) Geben Sie für jeden *lift*-Schritt des Algorithmus den "Überschuss und die Höhe (Beschriftung) d der sich ändernden Knoten an.
- b) Falls am Ende ein "Überschuss in den inneren Knoten übrig bleiben sollte, beschreiben Sie in wenigen Worten, wie der Algorithmus darauf reagiert. (D.h., Sie müssen den "Überschussabbau nicht schrittweise durchrechnen.)
- c) Geben Sie anschließend einen Schnitt in N an, der von f^* saturiert wird.



b) Die Knoten werden so lange geliftet bis sie höher als die Quelle sind und der Überschuss an diese zurückfließt.

c) Schnitt(s,a,b,c) mit Fluss 9

Aufgabe 3.B: Verschiedenes

a) (3 Punkte)

Beschreiben Sie in einfachen Worten, was das Separationsproblem in einem Schnittebenenverfahren ist.

Finden einer weiteren Restriktion die für alle Punkte im Polytop gültig ist, jedoch für den gegebenen Punkt x_* verletzt ist.

b) (3 Punkte)

Beschreiben Sie in einfachen Worten, was *Branch-and-Price* von einfachem LPbasierten *Branch-and-Bound* unterscheidet.

Beim *Branch-and-Price* werden die Teilprobleme mittels Spaltengenerierung gelöst. Sollte dies keine Lösung bringen, wird das Problem in weitere Teilprobleme aufgespalten (branching)

c) (2 Punkte)

Was ist der Unterschied zwischen den *Las-Vegas*-Verfahren und den *Monte-Carlo*-Verfahren?

Las-Vegas-Verfahren liefern stets korrekte Ergebnisse während *Monte-Carlo*-Verfahren eine Lösung liefern, die mit einer gewissen Fehlerwahrscheinlichkeit behaftet ist.

d) (2 Punkte)

Was ist der Unterschied zwischen einer *perfekten Skipliste* und einer *randomisierten Skipliste*? Welche Vorteile besitzt die bessere der beiden Datenstrukturen?

Bei einer perfekten skipliste ist die höhe der container fix.
Bei der randomisierten werden die höhen durch zufall ermittelt.
Der vorteil der randomisierten skipliste ist, dass beim einfügen und löschen nie reorganisiert werden muss sondern nur zeiger verbogen werden müssen (weil es keine vorgabe für die höhe der container an bestimmten plätzen gibt)

Aufgabe 4.B: Textsuche

Betrachten Sie den Algorithmus von *Boyer-Moore* (BM), den Text $T = \text{ASSSOSASOSOS}$ und das Muster $P = \text{ASOSOS}$.

a) (2 Punkte)

Geben Sie das *last* [] Array an.

b) (4 Punkte)

Geben Sie das *suffix*[] Array für P an.

c) (4 Punkte)

Suchen Sie mittels BM in T nach P . Visualisieren Sie in jedem Schritt, wo das Muster angelegt wird. Geben Sie an

- welche Zeichen gematcht werden,
- welches Zeichen gegebenenfalls den Mismatch verursacht, und
- mit welcher Regel (*last* oder *suffix*) verschoben wird.

(Wenn Sie die Suche nur mittels *last* [] Array oder nur mittels *suffix*[] Array durchführen, wird ein Punkt abgezogen.)

a)

$\text{last}[] = \text{A}=1 \text{ S}=6 \text{ O}=5$

b)

$\text{next}[] = 0 \ 0 \ 0 \ 0 \ 0 \ 0$

$\text{SOSOSA next}^{-1}[] = 0 \ 0 \ 1 \ 2 \ 3 \ 0$

$\text{Suffix}[] = \text{M-Next}[\text{M}] = 6 \ 6 \ 6 \ 6 \ 6 \ 6$

$\text{J}=\text{M-next}^{-1}[\text{q}] = 6 \ 6 \ 5 \ 4 \ 3 \ 6$

Wenn ($\text{suffix}[\text{q}] > \text{q-next}^{-1}[\text{q}]$) dann $\text{suffix}[\text{j}] = \text{q-next}^{-1}[\text{q}]$

$\text{Suffix}[] = 6 \ 6 \ 2 \ 2 \ 2 \ 1$

ASSSOSASOSOS

ASOSOS => $\text{j}=6-3=3$ $\text{suffix}[3]=2$ $\text{j-last}[\text{s}] = 3-6=-3$; suffix-verschiebung

ASOSOS => $\text{j}=6$; $\text{suffix}[6] = 1$ $6-\text{last}[\text{o}] = 1$, verschiebung um 1

ASOSOS => $\text{j}=3$; $\text{suffix}[3]=2$ $3-\text{last}[\text{a}] = 2$, verschiebung um 2

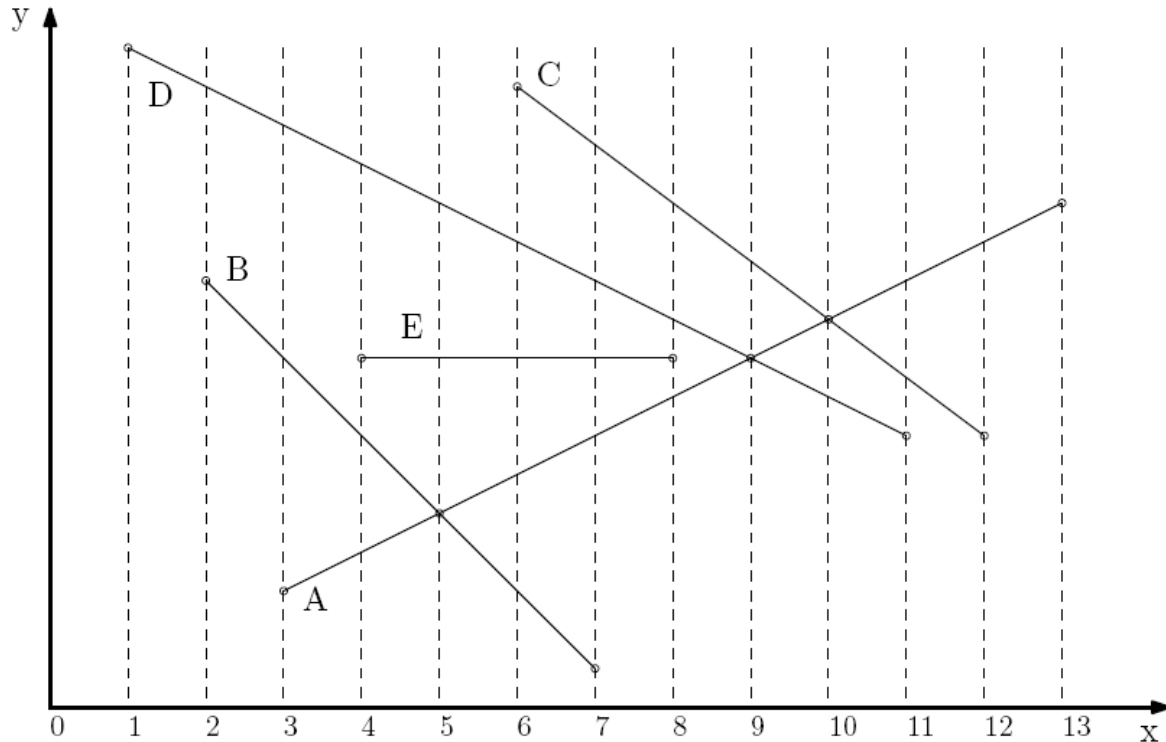
ASOSOS => match!

Aufgabe 5.B: Schnitt von allgemeinen Liniensegmenten

a) (7 Punkte)

Führen Sie den aus der Vorlesung bekannten Algorithmus zum Schnitt von allgemeinen Liniensegmenten für die folgende Menge von Linien aus. Geben Sie dabei für jeden Zeitpunkt (0 . . . 13) im Verlauf der Bewegung der Scan-Line den Zustand der Scan-Line-Status Struktur an.

(Die vertikalen Linien dienen nur zur Orientierung)

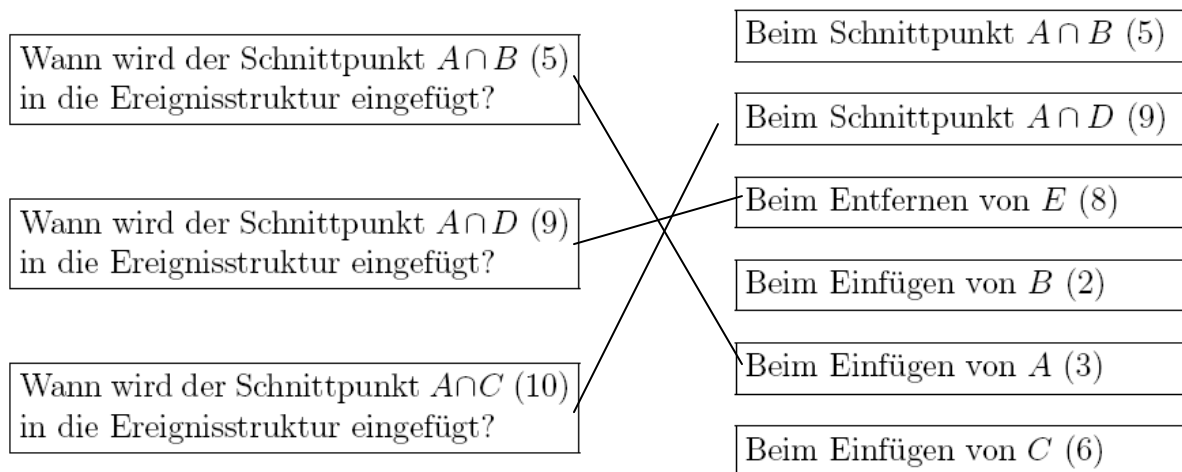


SSS	ES	Nächster Schritt
leer	D,B,A,E,C,B',E',D',C'A'	D in SSS, aus ES, Schnitt mit vorg u. nachf prüfen->0
D	B,A,E,C,B',E',D',C'A'	B in SSS, aus ES, Schnitt mit vorg u. nachf prüfen->0
B,D	A,E,C,B',E',D',C'A'	A in SSS, aus ES, Schnitt mit vorg u. nachf prüfen->AB in ES
A,B,D	E,AB,C,B',E',D',C'A'	E in SSS, aus ES, Schnitt mit vorg u. nachf prüfen->0
A,B,E,D	AB,C,B',E',D',C'A'	AB in SSS tauschen, Schnitt prüfen B mit vorg, A mit nachf->0
B,A,E,D	C,B',E',D',C'A'	C in SSS, aus ES, Schnitt mit vorg u. nachf prüfen->0
B,A,E,D,C	B',E',D',C'A'	B' aus ES, B aus SSS, vorg, nachf von B auf Schnitt prüfen->0
A,E,D,C	E',D',C'A'	E' aus ES, E aus SSS, vorg u. nachf von E auf Schnitt prüfen->AD in ES
A,D,C	AD,D',C',A'	AD in SSS tauschen,

		Schnitt prüfen D vorg, A nachf->AC in ES
D,A,C	AC,D',C',A'	AC in SSS tauschen, Schnitt prüfen, C vorg, A nachf -> 0
D,C,A	D',C',A'	D' aus ES, D aus SSS, vorg u. nachf von D auf Schnitt prüfen->0
C,A	C',A'	C' aus ES, C aus SSS, vorg u. nachf von C auf Schnitt prüfen->0
A	A'	A' aus ES, A aus SSS, vorg u. nachf von A auf Schnitt prüfen->0

b) (3 Punkte)

Verbinden Sie die folgenden Fragen mit den richtigen Antworten



20070309

Aufgabe 1.A: Textsuche

Betrachten Sie den Algorithmus von *Knuth-Morris-Pratt* (KMP), den Text $T = \text{MEERMARMORMARMARA}$ und das Muster $P = \text{MARMARA}$.

Betrachten Sie den Algorithmus von *Knuth-Morris-Pratt* (KMP), den Text $T = \text{MEERMARMORMARMARA}$ und das Muster $P = \text{MARMARA}$.

a) (2 Punkte)

Geben Sie für das Muster P das `next[]` Array an.

b) (4 Punkte)

Suchen Sie mittels KMP im Text T nach dem Muster P . Visualisieren Sie jeden Schritt und markieren Sie dabei deutlich die Mismatches.

c) (4 Punkte)

Erklären Sie in wenigen aber möglichst genauen Sätzen das Signaturverfahren von

Karp und Rabin. Geben Sie die Laufzeit dieses Algorithmus in \mathcal{f} -Notation an (der

Text besteht aus N Zeichen und das Muster aus M Zeichen).

a)

```
next[] = 0 0 0 1 2 3 0
```

b)

```
MEERMARMORMARMARA
```

```
MARMARA -> verschieben um 1
```

```
  MARMARA -> verschieben um 1
```

```
    MARMARA -> verschieben um 1
```

```
      MARMARA -> verschieben um 1
```

```
        MARMARA -> verschieben um 3(2 im next array=>3 verschieben)
```

```
          MARMARA -> verschieben um 1
```

```
            MARMARA -> verschieben um 1
```

```
              MARMARA -> verschieben um 1
```

```
                MARMARA -> Match
```

c)

Es wird eine inkrementelle Hashfunktion zum Abbilden der Werte von strings verwendet und damit der Vergleich durchgeführt. Werden 2 gleiche Hashwerte erzeugt werden sicherheitshalber die Strings verglichen.

Laufzeit: $O(m+n)$

Aufgabe 2.A: Skiplisten

a) (6 Punkte)

Fügen Sie in eine anfangs leere Skipliste S die folgenden Elemente in der vorgegebenen

Reihenfolge ein:

13, 44, 10, 3, 11, 5, 50, 43, 20

Verwenden Sie hierbei die folgenden, zufällig gewählten Höhen:

0, 1, 3, 0, 0, 1, 0, 0, 2

Zeichnen Sie die resultierende Skipliste.

b) (2 Punkte)

Suchen Sie nun nach der Zahl 45. Markieren Sie alle Container, deren Schlüssel bei

der Suche verglichen wurden. (Vergleiche mit 1 zählen mit)

c) (2 Punkte) Wieviele Schlüsselvergleiche waren hierfür notwendig?

(Vergleiche mit 1 zählen mit)

a)

```
3----->| |-----> 3
2----->| |-----> 2
1----->| |-----> 1
0-> 3 -> |5| -> |10| -> 11 -> 13 -> |20| -> 43 -> |44| -> 50 -> 0
```

b)

```
3-----1----->| |-----2-----> 3
2----->| |-----3-----> | |-----4-----> 2
1----->| |-----> | |-----5-----> | |-----6-----> 1
0-> 3 -> |5| -> |10| -> 11 -> 13 -> |20| -> 43 -> |44| -7-> 50 -> 0
```

c)

8 vergleiche – ich zähle die pfade + 1 für den letzten vergleich mit 50

Aufgabe 3.A: Simplex-Verfahren

Gegeben ist das folgende lineare Programm (LP):

max $2x_1 + 3x_2$

s.t $-x_1 + x_2 \leq 4$

$-x_1 + 2x_2 \leq 10$

$2x_1 + x_2 \leq 20$

$x_1, x_2 \geq 0$

a) (5 Punkte)

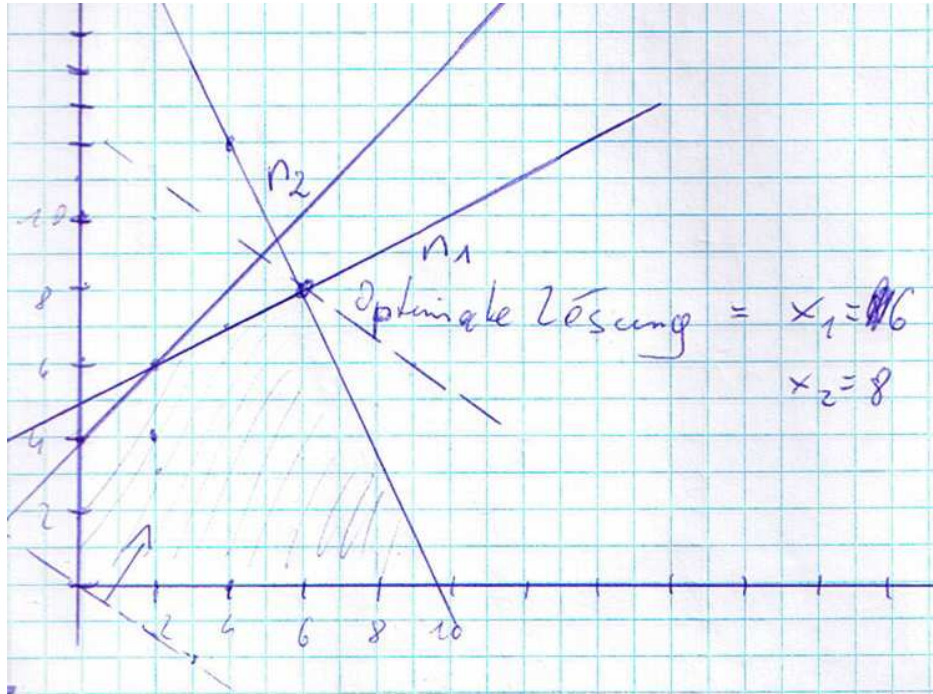
Lösen Sie das LP grafisch:

- Zeichnen Sie den zulässigen Lösungsbereich ein.
- Markieren Sie die optimale Lösung.

b) (5 Punkte)

Schreiben Sie das LP als Simplextableau auf und führen Sie die *erste* Simplex- Iteration aus. (D.h., das LP braucht nicht vollständig gelöst zu werden.)

A)



b)

tableau

$$z = 2x_1 + 3x_2$$

$$x_3 = x_1 - x_2 + 4$$

$$x_4 = x_1 + 2x_2 + 10$$

$$x_5 = -2x_1 - x_2 + 20$$

Einschränkung aus x_5 : $x_1 \leq 20/2 \dots$ also 10

x_3 & x_4 keine Einschränkungen, da bei beiden '+ x_1 ' steht.

1. iteration: x_1 kommt in die basis, x_5 verlässt sie

$$z = 20 + 2x_2 - x_5$$

$$x_1 = 10 - 1/2 x_2 - 1/2 x_5$$

$$x_3 = 14 - 3/2 x_2 - 1/2 x_5$$

$$x_4 = 20 - 5/2 x_2 - 1/2 x_5$$

=> Lösung $x_1=10$, $x_2=0$, $x_3=14$, $x_4=20$, $x_5=0$

Einschränkung aus x_1 : $x_2 \leq 20$

Einschränkung aus x_3 : $x_2 \leq 28/3 \dots 9,33$

Einschränkung aus x_4 : $x_2 \leq 8 \dots$ größtmöglicher Wert -> höchste Einschränkung!

x_4 umformen:

$$x_2 = 8 - 2/5 x_4 - 1/5 x_5$$

Einsetzen 2. Iteration:

$$\max z = 36 - 4/5 x_4 - 7/5 x_5$$

$$x_1 = 6 + 1/5 x_4 - 4/10 x_5$$

$$x_2 = 8 - 2/5 x_4 - 1/5 x_5$$

$$x_3 = 2 + \frac{6}{10} x_4 - \frac{2}{10} x_5$$

$$\Rightarrow \text{Lösung } x_1=6, x_2=8, x_3=2, x_4=0, x_5=0$$

Aufgabe 4.A: Fluss-Algorithmen

a) (3 Punkte)

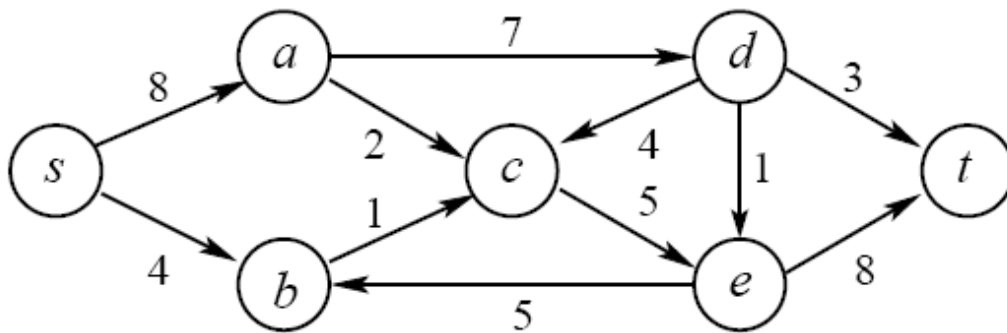
Schreiben Sie das Max-Flow-Min-Cut-Theorem auf.

- 1) Es gibt einen Schnitt S in N der von f saturiert wird.
- 2) f ist max. Fluss N
- 3) Es existiert kein augmentierender Pfad im Restgraph G_f .

b) (4 Punkte)

Finden Sie zu dem nachfolgenden Flussnetzwerk N den maximalen Fluss f^* von der Quelle s zur Senke t , indem Sie den Ford-Fulkerson Algorithmus anwenden.

Benutzen Sie hierfür die Templates auf der nächsten Seite. (Anm.: Sie müssen nicht alle Templates verwenden.)



a) Geben Sie für jeden Schritt des Algorithmus den augmentierenden Pfad an.

Aufgabe 5.A: Verschiedenes

a) (2 Punkte)

Scan-Line Algorithmus:

Was ist eine *Ereignisdatenstruktur* bzw. was wird darin gespeichert? Womit wird sie initialisiert?

Es werden alle Ereignisse (Startpunkt, Endpunkt, Schnittpunkt) aufsteigend nach deren X-Koordinate gespeichert.

Initialisiert wird sie mit den Start- und Endpunkten, die Schnittpunkte werden während der Laufzeit hinzugefügt.

b) (2 Punkte)

Cache optimale Algorithmen:

Beschreiben Sie in einfachen Worten, was unter dem Begriff "*örtliche Lokalität*" verstanden wird. Wie kann diese vom Cache ausgenutzt werden?

-) **Örtliche Lokalität:** es werden öfter Speicherbereiche angesprochen, die nahe beieinander liegen. Caches nutzen dies aus, indem sie bei jedem Miss jeweils ganze Speicherblöcke in den Cache holen.

c) (3 Punkte)

Beschreiben Sie in einfachen Worten, was das *Pricing-Problem* in einem *Spaltengenerierungsverfahren* ist.

Lösen des Subproblems das jene Variable mit maximalen (minimalen) reduzierenden Kosten findet.

d) (3 Punkte)

Beschreiben Sie in einfachen Worten, was *Branch-and-Cut* von einfachem LPbasierten Branch-and-Bound unterscheidet.

Dieses verfahren verbindet das branch-and-bound verfahren mit dem schnittebenenverfahren.

beim branch-and-cut verfahren wird das schnittebenenverfahren zum lösen der teilprobleme verwendet. Sollte dies keine lösung bringen, wird weiter aufgeteilt (branching)