Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Institut für Theoretische Informatik
Peter Widmayer
Holger Flier

Prüfung
Datenstrukturen und Algorithmen
D-INFK

11. August 2011

Name, Vorname: _____

Stud.-Nummer: _____

Ich bestätige mit meiner Unterschrift, dass ich diese Prüfung unter regulären Bedingungen ablegen konnte und dass ich die untenstehenden Hinweise gelesen und verstanden habe.

Unterschrift: _____

Hinweise:

- Ausser einem Wörterbuch dürfen Sie keine Hilfsmittel verwenden.
- Bitte schreiben Sie Ihre Studierenden-Nummer auf **jedes** Blatt.
- Melden Sie sich bitte **sofort**, wenn Sie sich während der Prüfung in irgendeiner Weise bei der Arbeit gestört fühlen.
- Bitte verwenden Sie für jede Aufgabe ein neues Blatt. Pro Aufgabe kann nur eine Lösung angegeben werden. Ungültige Lösungsversuche müssen klar durchgestrichen werden.
- Bitte schreiben Sie **lesbar** mit blauer oder schwarzer Tinte. Wir werden nur bewerten, was wir lesen können.
- Die Prüfung dauert 120 Minuten. **Keine Angst!** Wir rechnen nicht damit, dass irgendjemand alles löst! Sie brauchen bei weitem nicht alle Punkte, um die Bestnote zu erreichen.

Viel Erfolg!

Stud.-Nummer: _____

Aufgabe	1	2	3	4	5	**Σ**
Mögl. Punkte	12	7	9	10	9	47
$\sum$ Punkte						

Aufgabe 1:

Hinweise:

1. In dieser Aufgabe sollen Sie **nur die Ergebnisse** angeben. Diese können Sie direkt bei den Aufgaben notieren.
2. Sofern Sie die Notationen, Algorithmen und Datenstrukturen aus der Vorlesung "Datenstrukturen & Algorithmen" verwenden, sind Erklärungen oder Begründungen nicht notwendig. Falls Sie jedoch andere Methoden benutzen, müssen Sie diese **kurz** soweit erklären, dass Ihre Ergebnisse verständlich und nachvollziehbar sind.
3. Als Ordnung verwenden wir für Buchstaben die alphabetische Reihenfolge, für Zahlen die aufsteigende Anordnung gemäss ihrer Grösse.

1 P a) Das Verfahren von Karatsuba/Ofman zur Multiplikation ganzer Zahlen berechnet das Produkt zweier Zahlen rekursiv anhand einer Formel, die bis auf Addition und Multiplikation mit der Basis (hier: 10) drei Produkte enthält. Geben Sie zwei Zahlen x und y an, für welche diese drei Produkte $(74 \cdot 93)$, $(51 \cdot 80)$ und $(74 \pm 80) \cdot (51 \pm 93)$ sind.

$x =$ _____, $y =$ _____

1 P b) Geben Sie eine Folge von 5 Zahlen an, bei denen Bubblesort zum Sortieren genau 10 Vertauschungen von Schlüsseln durchführt.

Folge: _____

1 P c) Das untenstehende Array enthält die Elemente eines Min-Heaps in der üblichen Form gespeichert. Wie sieht das Array aus, nachdem das Minimum entfernt wurde und die Heap-Bedingung wieder hergestellt wurde?

25	60	32	61	62	52	57	80	86
1	2	3	4	5	6	7	8	9

1 P d) Nehmen Sie an, die Schlüssel $7, 40, 13, 3, 24, 17, 1, 10, 14, 31, 15, 4, 23, 28, 11$ sollen in dieser Reihenfolge mittels Cuckoo-Hashing in Hashtabellen t_1 und t_2 eingefügt werden. Die Hashfunktion für Tabelle t_1 lautet $h_1(k) = k \mod 7$, die für Tabelle t_2 lautet $h_2(k) = 3k \mod 7$. Nehmen Sie ferner an, dass t_1 und t_2 jeweils die Grösse 7 haben. Wie lautet der erste Schlüssel in der oben genannten Folge, der nicht mehr eingefügt werden kann, ohne die Tabellen zu vergrössern (also eine `rehash` Operation auszuführen)?

Schlüssel: _____

1 P e) Zeichnen Sie den binären Suchbaum für die Schlüssel $1, 2, 3, 4, 5, 6, 7, 8$, dessen Preorder-Traversierung mit der Folge $5, 3, 1, 2$ beginnt und dessen Postorder-Traversierung mit der Folge $7, 8, 6, 5$ endet.

1 P f) Geben Sie eine Folge von höchstens 5 Zugriffen auf die Liste $A \to B \to C \to D \to E$ an, welche unter Benutzung der Move-To-Front-Regel genau 17 Vergleichsoperationen benötigt.

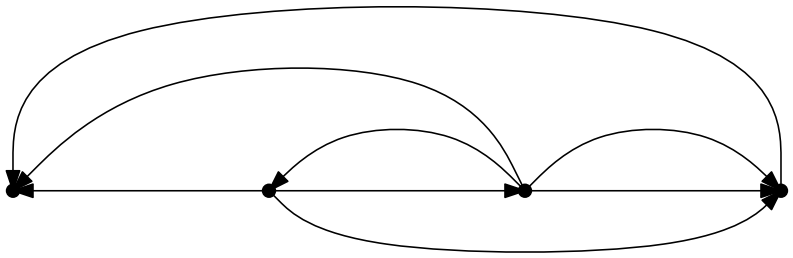
Folge von Zugriffen: _____

1 P g) Gegeben sind acht Schlüssel mit der relativen Anzahl der Zugriffe. Erstellen Sie mit Hilfe des Huffman-Algorithmus einen Codierungsbaum (Trie).

Schlüssel	a	b	c	d	e	f	g	h
Anzahl Zugriffe	41	33	70	39	23	25	78	98

1 P

h) Markieren Sie in folgendem Graphen $G = (V, E)$ eine kleinstmögliche Menge S an Kanten, so dass der Graph $G' := (V, E \setminus S)$ eine topologische Sortierung hat :

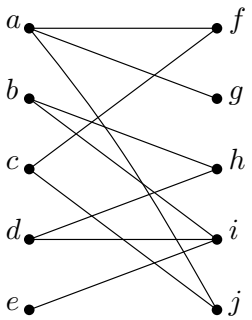


1 P

i) Geben Sie ein Beispiel für einen gerichteten Graphen $G = (V, E)$ an, der höchstens 6 Kanten hat und für den der Zunehmende-Wege-Algorithmus (Ford-Fulkerson) bei unglücklicher Wahl von zunehmenden Wegen bis zu 10000 Flusserhöhungen vornimmt, um einen maximalen Fluss zu berechnen. Geben Sie für jede Kante die Kapazität an. Kennzeichnen Sie auch den Startknoten s und Zielknoten t.

1 P

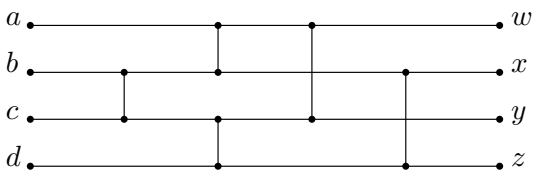
j) Geben Sie eine Teilmenge der Knoten des folgenden bipartiten Graphen an, die mit dem Satz von Hall beweist, dass der Graph kein perfektes Matching hat.



1 P k) Vervollständigen Sie die lückenhaften Zeilen des folgenden Pseudocodes, so dass der resultierende Algorithmus eine 2-Approximation für Minimum-Vertex-Cover liefert.

Input: Graph $G = (V, E)$
$S = \emptyset$;
$U = E$;
while $U \neq \emptyset$ **do**
  wähle eine Kante $e = (u, v) \in U$;
  $S = S \cup$ _____ ;
  $U = U \setminus$ _____ ;
end
return S;

1 P l) Folgendes Netzwerk aus Sortierbausteinen ist kein Sortiernetzwerk. Geben Sie Werte für die Inputs a, b, c, d des Netzwerks an, welche durch das Netzwerk nicht korrekt sortiert werden. (Ein korrekt sortierter Output hat dabei die Form $w \leq x \leq y \leq z$.)



Aufgabe 2:

1 P a) Geben Sie für die untenstehenden Funktionen eine **Reihenfolge** an, so dass folgendes gilt: Wenn Funktion f links von Funktion g steht, so gilt $f \in O(g)$.

Beispiel: Die drei Funktionen n^3, n^7, n^9 sind bereits in der entsprechenden Reihenfolge, da $n^3 \in O(n^7)$ und $n^7 \in O(n^9)$ gilt.

- $\sqrt{2^n}$
- $3^{\frac{n}{2}}$
- $\frac{2^n}{n!}$
- $n \log(n)$
- $n^{\log n}$
- $n^{\frac{3}{2}}$

3 P b) Gegeben ist die folgende Rekursionsgleichung:

$$T(n) := \begin{cases} 2 + 2T(\frac{n}{4}) & n > 1 \\ 1 & n = 1 \end{cases}$$

Geben Sie eine geschlossene (d.h. nicht-rekursive) und möglichst einfache Formel für $T(n)$ an und beweisen Sie diese mit vollständiger Induktion.

Hinweise:
(1) Sie können annehmen, dass n eine Potenz von 4 ist.
(2) Für $q \neq 1$ gilt: $\sum_{i=0}^{k} q^i = \frac{q^{k+1}-1}{q-1}$.

1 P

c) Geben Sie die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ für folgenden Algorithmus in Theta-Notation an (Sie müssen Ihre Antwort nicht begründen):

```
from j := 1 until j > n loop
  from k := 1 until k > n loop
    k := k + j
  end
  j := j + 1
end
```

1 P

d) Geben Sie die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ für folgenden Algorithmus in Theta-Notation an (Sie müssen Ihre Antwort nicht begründen):

```
from j := 1 until j > n*n loop
  from k := 2 until k > n*n*n loop
    k := k * 2
  end
  j := j + 1
end
```

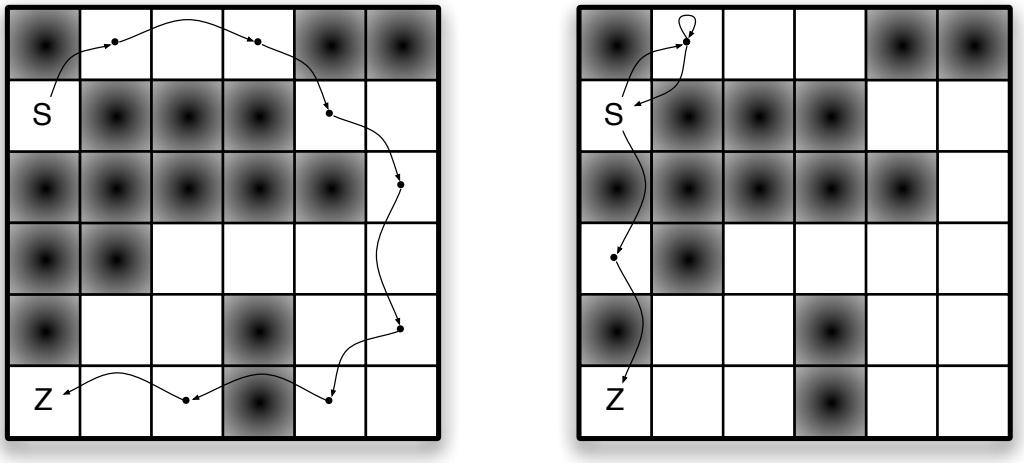
1 P

e) Geben Sie die asymptotische Laufzeit in Abhängigkeit von $n \in \mathbb{N}$ für folgenden Algorithmus in Theta-Notation an (Sie müssen Ihre Antwort nicht begründen):

```
from h := 1 until h > n loop
  from j := 1 until j*j > n*n loop
    j := j + 1
  end
  from k := 2 until k > n loop
    k := k * k
  end
  h := h + 10
end
```

Aufgabe 3:

In dieser Aufgabe geht es darum, einen kürzesten Weg für ein Pogo-Stick-Rennen in einem Parcours zu berechnen. Ein Parcours besteht aus einem Gitter von a mal b Feldern, wobei jedes Feld entweder *frei* oder durch ein Hindernis *belegt* ist. Mit einem Pogo-Stick kann man zwischen freien Feldern hin und her springen. Belegte Felder dürfen dagegen nie besucht werden. Wir bezeichnen mit *Sprungweite* (i, j), wobei $i, j \in \{-3, \ldots, 3\}$, ein Sprung von i Feldern in x- und j Feldern in y-Richtung. Somit sind 49 Sprungweiten von $(-3, -3)$ bis $(3, 3)$ möglich. Allerdings dürfen sich zwei aufeinander folgende Sprünge in der Sprungweite um nicht mehr als 1 pro Richtung unterscheiden. Hatte der letzte Sprung beispielsweise eine Sprungweite von (-2,3) (d.h. 2 nach links, 3 nach oben), so kann der nächste Sprung ausschliesslich mit einer der folgenden Sprungweiten ausgeführt werden: (-1,2), (-1,3), (-2,2), (-2,3), (-3,2) oder (-3,3).
Gesucht ist nun ein kürzester Weg, gemessen in der Anzahl an Sprüngen, von einem gegebenen Startfeld s zu einem gegebenen Zielfeld z, wobei s und z frei sind. Ein Sprung muss immer innerhalb des Parcours erfolgen, wobei belegte Felder übersprungen werden dürfen. Wir nehmen an, dass die Sprungweite beim Start von Feld s ursprünglich (0,0) ist und bei Ankunft am Ziel z beliebig ist. In den folgenden Beispielen besteht der kürzeste Weg aus 8 bzw. 5 Sprüngen:



2 P a) Modellieren Sie den Zustandsraum als gerichteten Graph $G = (V, E)$. Beschreiben Sie, welche Knoten und Kanten der Graph hat. Geben Sie dabei möglichst genau an, welche Knoten von einem gegebenen Knoten $v \in V$ aus erreichbar sind.

4 P b) Entwerfen Sie einen möglichst effizienten Algorithmus für das oben beschriebene Problem. Beschreiben Sie den Algorithmus kurz in Worten. Geben Sie Ihren Algorithmus in Pseudocode an. Der Algorithmus soll die Länge eines kürzesten Weges ausgeben, oder eine entsprechende Meldung, falls kein Weg von s nach z existiert.

1 P c) Geben Sie die Laufzeit Ihres Algorithmus in Abhängigkeit von a und b an.

1 P d) Erläutern Sie, wie Ihr Algorithmus ergänzt werden kann, um den berechneten kürzesten Weg auszugeben, d.h. als Folge von Feldern von s nach z. Geben Sie nur die zu b) nötigen Ergänzungen in Pseudocode an.

1 P e) Hat G eine topologische Sortierung? Begründen Sie Ihre Antwort kurz.

Aufgabe 4:
Gegeben sei eine Menge $S = \{p_1, \ldots, p_n\}$ von Punkten in der Ebene. Gesucht ist ein *dichtestes* Paar von Punkten, d.h., zwei Punkte, welche die geringste Euklidische Distanz unter allen Paaren von Punkten aus S haben. Formal suchen wir also ein Paar $a, b \in S$, $a \neq b$, so dass $\text{dist}(a, b) = \min_{1 \leq i < j \leq n} \text{dist}(p_i, p_j)$ ist; dabei ist für zwei Punkte $p_i = (x_i, y_i)$ und $p_j = (x_j, y_j)$ die Euklidische Distanz definiert als $\text{dist}(p_i, p_j) := \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$
In dieser Aufgabe sollen Sie einen möglichst effizienten Sweepline-Algorithmus entwerfen, der ein dichtestes Paar von Punkten einer gegebenen Menge S berechnet.

1 P a) Geben Sie die Haltepunkte der Sweepline an sowie die Reihenfolge, in der die Sweepline auf die Haltepunkte trifft.

1 P b) Geben Sie die Invariante der Sweepline an, also welche Informationen in der mit der Sweepline assoziierten Datenstruktur D gespeichert werden.

1 P c) Welche Datenstruktur bietet sich für D an, damit Ihr Algorithmus möglichst effizient implementiert werden kann?

4 P d) Beschreiben Sie Ihren Algorithmus in Pseudocode.

3 P e) Geben Sie die Laufzeit Ihres Algorithmus an. Leiten Sie dazu eine konstante obere Schranke für die Anzahl der nötigen Distanzberechnungen her, welche an einem Haltepunkt der Sweepline ausgeführt werden. Geben Sie ferner die Laufzeit der im Verlauf Ihres Algorithmus ausgeführten Operationen auf der Datenstruktur D an und begründen Sie diese kurz.

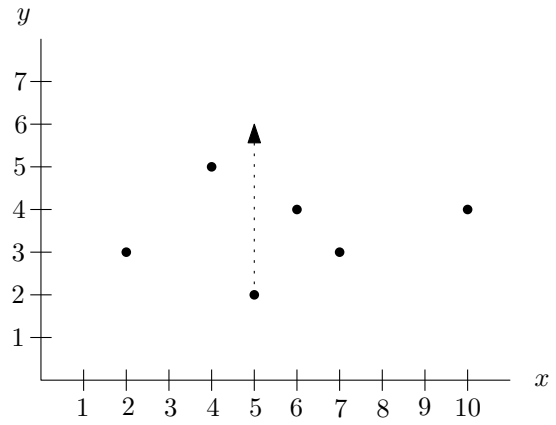
Hinweis: Sie dürfen annehmen, dass keine zwei Punkte aus S dieselbe x- oder y-Koordinate haben.

Aufgabe 5:

In dieser Aufgabe soll eine Datenstruktur für das folgende Problem entworfen werden: Gegeben sind n Punkte $p_1, p_2, \ldots, p_n$ mit ganzzahligen Koordinaten $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. Die x_i sind alle verschieden und unveränderlich. Die Punkte können jedoch auf der y-Achse verschoben werden, d.h. die Werte y_i können durch Operationen verändert werden.
Die zu entwerfende Datenstruktur soll die folgenden Operationen möglichst effizient unterstützen:

- `RangeMin(`$x_{\min}$`, `$x_{\max}$`)`:
  Liefert die kleinste y-Koordinate eines Punktes im Intervall $[x_{\min}, x_{\max}]$ zurück.
- `Update(`x_i`, `y_{new}`)`:
  Setzt die y-Koordinate von Punkt (x_i, y_i) auf den Wert y_{new}.

Im untenstehenden Beispiel würde also `RangeMin(4, 7)` zuerst den Wert 2 liefern. Wenn danach jedoch die Operation `Update(5, 6)` ausgeführt wird, liefert `RangeMin(4, 7)` den Wert 3.



3 P a) Entwerfen und beschreiben Sie eine Datenstruktur für das obige Problem. Geben Sie eine nützliche Invariante an, welche für die in Ihrer Datenstruktur gespeicherten Werte gelten soll (unabhängig von der Anzahl ausgeführter `Update`-Operationen).

3 P b) Beschreiben Sie, wie die in a) entworfene Datenstruktur benutzt werden kann, um die Operation `RangeMin(`$x_{\min}$`, `$x_{\max}$`)` effizient zu implementieren. Geben Sie zudem die Laufzeit dieser Operation im schlechtesten Fall an.

3 P c) Beschreiben Sie, wie die Datenstruktur aktualisiert wird, wenn die Operation `Update(`x_i`, `y_{new}`)` ausgeführt wird.