PHILIPPS-UNIVERSITÄT MARBURG
Fachbereich Mathematik und Informatik
Prof. Dr. R. Loogen, J. Berthold

D-35032 Marburg
Hans Meerwein Straße
Lahnberge

25. Juni 2002

Übungen zu „Parallele und verteilte Algorithmen“, Sommer 2002

Nr. 10 (letztes Blatt), Abgabe: 2. Juli in der Vorlesung

Die Abgabe ist in Gruppen bis zu zwei Personen erlaubt. Programme sind schriftlich (etwa als Ausdruck) **und** per email an `pinf3@mathematik.uni-marburg.de` abzugeben.

---

23. Zykeltest — 10 P.

Gegeben sei ein gerichteter Graph $G = (V, E)$ durch seine Adjazenzmatrix $A = (a_{ij})$. Es sei $a_{ij} = 1$ genau dann, wenn es eine gerichtete Kante vom Knoten i zum Knoten j gibt. Ansonsten sei $a_{ij} = 0$.

Geben Sie einen Algorithmus für den Prozessortorus mit $n^2 = |V|^2$ Prozessoren an, der in $\mathcal{O}(n \log n)$ Schritten entscheidet, ob G einen gerichteten Kantenzyklus enthält. Dabei darf jeder Prozessor nur eine konstante, d.h. insbesondere von n unabhängige Anzahl von Speicherplätzen benutzen.

Hinweis: Beweisen und verwenden Sie:

$$\sum_{i=0}^{2^k-1} A^i = \prod_{i=0}^{k-1} (1 + A^{2^i})$$

24. Kürzeste Wege — 10 P.

Beim „Einzelne-Quelle kürzeste-Wege“-Problem muss der kürzeste Weg von einem festgelegten Quellknoten s zu allen anderen Knoten eines gewichteten, gerichteten Graphen bestimmt werden.

Der folgende sequentielle Algorithmus wurde hierzu 1959 von E. F. Moore entwickelt:

Parameter:	n	Anzahl der Graphknoten
Globale Variablen:	s	Quellknoten
	distance	Feld mit Abständen von s zu allen anderen Knoten
	weight	Kostenmatrix

```
begin
   for i := 1 to n do
       INITIALISE(i)
   od
   insert s into the queue
   while the queue is not empty do
       SEARCH()
   od
end
```

`INITIALISE` initialisiert das Feld *distance* für den Quellknoten mit 0 und für jeden anderen Knoten mit ∞.

`SEARCH()`:

Local:	u	untersuchter Knoten
	v	von u über einzelne Kante erreichbarer Knoten
	new.distance	Abstand zu v bei Weg über u

```
begin
   dequeue vertex u
   for every edge (u,v) in the graph do
       new.distance := distance(u) + weight(u,v)
       if new.distance < distance(v) then
           distance(v) := new.distance
           if v is not in the queue then
               enqueue v
           fi
       fi
   od
end
```

(a) Diskutieren Sie die Parallelisierbarkeit dieses Algorithmus.

(b) Beschreiben Sie (in Pseudocode) einen parallelen Algorithmus, der mit einer Reihe von asynchronen Prozessen arbeitet, die die Prozedur `SEARCH` parallel ausführen. Den Prozessen stehe ein gemeinsamer Speicher zur Verfügung.

Beachten Sie, dass die Prozesse erst terminieren, wenn keine Arbeit mehr zu leisten ist, und dass beim Schreiben in den gemeinsamen Speicher, falls erforderlich, eine Synchronisation erfolgt.

Zur Synchronisation können Sie Prozeduren `lock()` und `unlock()` auf geeigneten Synchronisationsvariablen verwenden. Welcher Speicherbereich muss wann geschützt werden?