Philipps Universität Marburg

Übungen zu „Parallele und Verteilte Algorithmen“, Winter 09/10

Prof. Dr. R. Loogen, Dipl.-Inform. Th. Horstmeyer · Fachbereich Mathematik und Informatik · Marburg

Nr. 10, Abgabe: Dienstag, 19. Januar 2010 vor der Vorlesung

Aufgaben

10.1 Topologische Sortierung — 5 Punkte

Gegeben sei ein gerichteter, zyklenfreier Graph G durch seine Adjazenzmatrix $A = (a_{ij})$.

Geben Sie einen PRAM-Algorithmus an, der die Knoten des Graphen topologisch sortiert.

Eine Folge von Knoten $n_1, ..., n_k$ heißt topologisch sortiert, wenn für alle Paare (n_i, n_j) mit $i < j$ gilt, dass es in G keinen Weg von n_j nach n_i gibt. D.h. ein Knoten steht vor allen anderen Knoten, die von ihm aus erreichbar sind. Eine topologische Sortierung ist nicht zwingend eindeutig.

10.2 Kürzeste Wege — 7 Punkte

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

Bitte wenden!

```
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. / 2

(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. / 5

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?