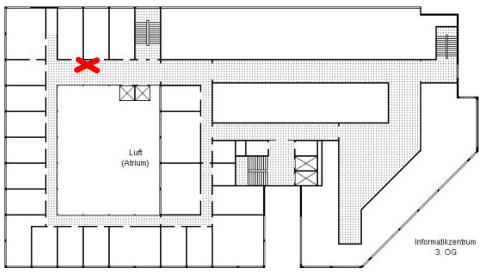
Abteilung Algorithmik **Sommer '12**
Institut für Betriebssysteme und Rechnerverbund
TU Braunschweig

Alexander Kröller
Henning Hasemann
Stephan Friedrichs

Verteilte Algorithmen
Übung 1 vom 24. 4. 2012

Abgabe bis Dienstag, den 8. 5. 2012, entweder

- vor der Übung im IZ358, oder
- bis 9:40 im Hausaufgabenrückgabeschrank.



Bitte die Blätter vorne deutlich mit eigenem Namen versehen!

P-Aufgaben sind als Quelltext per Mail an `hasemann@ibr.cs.tu-bs.de` abzugeben.

Zu den drei Tracks: Entscheidet euch, ob ihr lieber die T- oder die P-Aufgaben bearbeiten wollt. Ihr könnt euch auf späteren Blättern wieder umentscheiden.
Also: entweder A+T oder A+P abgeben.

A — Allgemeiner Teil

Diese Aufgaben können von jedem bearbeitet werden, egal ob sich ansonsten T oder P entschieden wird.

Aufgabe A1: Algorithmus 1.18 zur 3-Färbung eines Baumes wurde in der VL für SYNC formuliert. Gib analog dazu einen korrekten Algorithmus für ASYNC an, der einen Baum mit ausgezeichneter Wurzel in Zeit $\mathcal{O}(\log^* n)$ 3-färbt.
Unterroutinen (etwa eine asynchrone Variante von Algorithmus 1.15) müssen ebenfalls angegeben werden! Die Eigenschaften (Korrektheit und Laufzeit) müssen begründet werden. **(3 P.)**

Aufgabe A2: Nehmen wir an, alle Knoten hätten eine ID, können jedoch nur die Operationen $=$ und $\neq$ darauf anwenden. Ist Leader Election in diesem Modell lösbar?
(Hinweis: Begründe Deine Antwort! Ein einfaches „ja“ oder „nein“ reicht nicht aus.) **(1 P.)**

T — Theoretischer Track

Aufgabe T1: Zeige: Für jedes $n \in \mathbb{N}$ gibt es (jeweils) Netze mit mindestens n Knoten, so dass

a) Der LCR-Algorithmus tatsächlich $\Omega(n^2)$ Nachrichten verschickt.

b) Der LCR-Algorithmus nur $O(n)$ Nachrichten verschickt.

(2+2 P.)

Aufgabe T2: Gegeben sei ein Ring von n asynchronen Knoten mit IDs, die nicht fortlaufend sind. Gib einen Algorithmus an, der in Zeit $\mathcal{O}(n)$ jedem Prozessor eine neue ID aufsteigend von 1 bis n zuweist. **(2 P.)**

P — Praktischer Track

Diese Aufgabe schliesst sich mit T aus:

Aufgabe P1: Implementiere Algorithmus 1.18 in der Wiselib. Dafür ist natürlich eine asynchron verwendbare Version nötig — vgl. Aufgabe A1.
Die Wiselib ist auf `www.Wiselib.org` herunterzuladen und dokumentiert. Es empfiehlt sich **sehr**, nicht die Software selber zu installieren. Stattdessen sollte die virtuelle Maschine, in der sämtliche benötigte Software fertig konfiguriert vorliegt, von der Webseite gezogen und verwendet werden.
Ein vorbereitetes Framework für den Algorithmus, zusammen mit einer passenden Konfigurationsdatei für den Simulator Shawn, findet ihr unter
`http://www.ibr.cs.tu-bs.de/courses/ss12/va/ue/cole_vishkin.zip`.
Die Einbindung des Frameworks in die VM funktioniert folgendermassen:

- Ein Terminal öffnen.
- Das wiselib-repository aktualisieren:

  ```
  cd ~/wiselib
  git pull
  ```

- Das bereitgestellte Archiv herunterladen und unter `/wiselib/apps/generic_apps` entpacken (so dass das Verzeichnis `/wiselib/apps/generic_apps/cole_vishkin` entsteht):

  ```
  cd ~/wiselib/apps/generic\_apps
  wget http://www.ibr.cs.tu-bs.de/courses/ss12/va/ue/cole_vishkin.zip
  unzip cole_vishkin.zip
  ```

- Testen, ob der Code kompiliert:

  ```
  make
  ```

- Das so erstellte Kompilat kann man durch Aufruf von

  ```
  ./cole_vishkin < tree.conf
  ```

  testen. Die Anwendung erzeugt dabei 10 pdf-Dateien (`01.pdf` bis `10.pdf`), welche die aktuelle Farben der Knoten zu verschiedenen Zeitpunkten anzeigen (in der Vorlage ist das immer die Knotenadresse).

(6 P.)