CpSc 448B **Homework 4** Due: Dec. 4, 2011

See homework 1 for more instructions on handing in homework.

1. **Dynamic Programming (80 points)**
   An implementation of the dynamic programming algorithm that from the Nov. 3 lecture is available at `http://www.ugrad.cs.ubc.ca/~cs448b/2011-1/hw/4/edist`.

   For simplicity, the algorithm in the lecture partitions the dynamic programming tableau into a $P \times P$ array of blocks. Each of the P processors works on one column of blocks. For each block, the processor first receives a cost vector from its "left" neighbour; it then computes the costs for tableau elements in the block; and then it sends a cost vector to its "right" neigbour. However, there is no reason that that the number of rows of blocks in this partitioning needs to be equal to the number of processors. This problem explores the trade-offs in changing the partitioning of the tableau.

   Let the $N \times N$ entries of the dynamic programming tableau be partitioned into $M \times P$ blocks; each block will have N/M rows and N/P columns. As before, each processor will work on a column of blocks. To process a block, the processor will receive a cost vector for N/M rows from its left neighbour, process the block, and then send the cost vector for these N/M rows to its right neighbour.

   (a) (**5 points**) What is the dominant overhead if M is small?

   (b) (**5 points**) What is the dominant overhead if M is large?

   (c) (**10 points**) The slides presented an analysis of the runtime of the simple algorithm. Revise this analysis to allow the tableau to be partitioned as described above. In other words, M should be parameter of your analysis.

   (d) (**10 points**) Derive a function or write code to compute the optimal choice of M given N, P, the model from the the Nov. 3 slides.
   Note: I know that slide 19 includes a constant term that is not included in the formula from slide 18 – the presence (or not) of the constant term doesn't matter for this problem.

   (e) (**50 points**) Modify the `mpi_edist` function in `mpi_edist.c` to use your optimal value of M. Validate your implementation by trying it for a few values of N and P. You should run the code with the optimal value of M and a few to either side to show that your result is optimal.

   - To make the problem feasible to grade, please put a comment of the form
     ```
     /*% Your Name ...
     /
     ```
     in front of each change that you make.
   - To further make grading feasible, I will post some particular choices for N and P that you should include in your tests.
   - If your measured times disagree with your prediction of optimality, point this out and give some possible explanations.

2. **The Zero-One Principle (25 points)**

   (a) (**10 points**) Write a function in Erlang or Java that correctly sorts all lists or arrays (for Java, you can choose lists or arrays) consisting only of ones and zeros but does not correctly sort all inputs.

   (b) (**5 points**) Provide an example input that your program does not sort correctly – shorter is better!

   (c) (**5 points**) Give a brief explanation of how your code works, why it sorts all inputs of zeros-and-ones correctly, and why it does not correctly sort arbitrary inputs.

   (d) (**5 points**) Explain why your program is not a counter-example to the zero-one principle (if it is a *valid* counter-example, you'll get **lots** of extra-credit!).

3. **Mesh sorting (50 points)**
Consider the following algorithm for sorting a $N \times N$ array of values – for simplicity, assume N is a power of 2:

```
for i = 1..log₂(N) {
   forall j in 0..(N-1) {
      if j is even
         sort row j in ascending order;
      else
         sort row j in descending order;
   }
   forall k in 0..(N-1) {
      sort column k in ascending order;
   }
}
forall j in 0..(N-1) {
   sort row j in ascending order;
}
```

Use the zero-one principle to show that at the end of this procedure, the array is sorted in lexical order: for all i_1, i_2, j_1 and j_2 in 1..N

- If $i_1 < i_2$, then $A(i_1, j_1) \leq A(i_2, j_2)$.
- If $i_1 = i_2$ and $j_1 < j_2$, then $A(i_1, j_1) \leq A(i_2, j_2)$.

4. **Extra credit**: I'll post two programming problems by the end of the weekend. They will be due on Dec. 13.