Mastermath midterm examination
Parallel Algorithms

Teacher: Rob H. Bisseling, Utrecht University

October 20, 2021

Each of the three questions is worth 10 points. Total time 120 minutes. Open-book exam. Motivate you answers!

1. Let $\mathbf{x} \neq 0$ be a given vector of length n, which is distributed by the cyclic distribution over p processors, with $n \bmod p = 0$.

   (a) [5 pt] Give an efficient BSP algorithm for processor $P(s)$ (in the notation we learned) for the computation of the ratio between the 2-norm and the 1-norm of $\mathbf{x}$,

   $$\frac{||\mathbf{x}||_2}{||\mathbf{x}||_1},$$

   where the k-norm is defined by

   $$||x||_k = \left(\sum_{i=0}^{n-1} |x_i|^k \right)^{\frac{1}{k}}.$$

   On output, every processor has to know the result.

   (b) [5 pt] Analyse the BSP cost of your algorithm. Assume here that taking the absolute value or the square root of a double costs 1 flop.

2. Let $\mathbf{a}_j, 0 \leq j < m$, be a set of m vectors, each of length n. Each vector is distributed by the block distribution over p processors, where $n \bmod p = 0$. Assume $m \geq p$.

   The *Modified Gram-Schmidt (MGS)* process is an algorithm for orthogonalising a set of vectors. MGS has good numerical properties (unlike the well-known original Gram-Schmidt process, from which it derives).

Algorithm 1 Modified Gram-Schmidt (MGS) process.

Input: $\mathbf{a}_j, 0 \leq j < m$, set of m independent vectors.
Output: $\mathbf{q}_j, 0 \leq j < m$, set of m vectors, with $\mathbf{q}_k^{\mathrm{T}}\mathbf{q}_j = 0$ for $k \neq j$ and $||\mathbf{q}_k|| = 1$, that span the same subspace as the input vectors.

for $k := 0$ **to** $m-1$ **do**
  $r_{kk} = ||\mathbf{a}_k||$;
  $\mathbf{q}_k = \frac{\mathbf{a}_k}{r_{kk}}$;
  for $j := k+1$ **to** $m-1$ **do**
    $r_{kj} = \mathbf{q}_k^{\mathrm{T}}\mathbf{a}_j$;
    $\mathbf{a}_j := \mathbf{a}_j - r_{kj}\mathbf{q}_k$;

The sequential MGS algorithm is as given as Algorithm 2. Here, the norm $||x||$ is the Euclidean 2-norm.

The computation is in-place, overwriting the input vectors.

(a) [5 pt] Give an efficient BSP algorithm, in the notation we have learned for processor $P(s)$, $0 \leq s < p$, for performing the MGS process. For ease of notation, write a_{ik} for the ith component of vector $\mathbf{a}_k$, and similarly q_{ik} for the ith component of vector $\mathbf{q}_k$. Be sure to minimise the amount of communication and the number of supersteps of your algorithm!

(b) [5 pt] Analyse the BSP cost of your algorithm. In the analysis, you may assume that m mod p is a convenient number.

3. Let $\mathbf{x}$ be a vector of length n that represents a *DNA sequence*, i.e., $x_i \in$`{A,C,T,G}` for $0 \leq i < n$. We want to compare it to another DNA sequence $\mathbf{y}$, of the same or different length, and see how similar it is. A method for finding similarity is *sequence alignment*, which inserts gaps (hyphens `-`) into both sequences such that every nucleotide x_i is either matched to an identical nucleotide $y_i = x_i$, or to a gap $y_i =$`-`, and vice versa.

   The aim of sequence alignment in its simplest form is to minimise the cost, i.e., the total number of gaps for given input sequences $\mathbf{x}$ and $\mathbf{y}$. Gaps at the start or end also count towards the cost. An example alignment with cost 4 is

```
-ATCGGTAAT
AATCGG-A--
```

(a) [3 pt] Design a sequential algorithm for comparing two DNA sequences of length n based on the *dynamic programming* technique. For this purpose, define c_{ij} as the minimal cost for aligning the first i nucleotides of $\mathbf{x}$ with the first j nucleotides of $\mathbf{y}$. Trivially, $c_{00} = 0$, $c_{0j} = j$ and so on. Hint: Express c_{ij} in terms of previous costs such as $c_{i-1,j}$ and some others, and possibly also dependent on the values of x_i and/or y_j. Note that we only ask for the minimal cost and not for the corresponding solution.

(b) [4 pt] Design an efficient BSP algorithm for the computation of the minimal cost $c_{n,n}$ of the sequence alignment using p processors. You can describe the algorithms in words (instead of the full notation we have learned), but still be precise so that it is clear which processor does what and when. You are free to distribute or replicate the vectors and distribute the (computation of the) cost matrix in any way you like. Note that there is no unique good answer to this question. Be creative and draw pictures!

(c) [2 pt] Analyse the BSP cost of your algorithm.

(d) [1 pt] Can you save memory and use less than $\mathcal{O}(n^2)$ memory? How?