Prof. G. Zachmann
A. Srinivas

University of Bremen
School of Computer Science
CGVR Group
May 28, 2014

Summer Semester 2014

Assignment on Massively Parallel Algorithms - Sheet 6

Due Date 11. 06. 2014

Exercise 1 (Matrix Vector Multiplication, *5 Credits*)

In the given Framework `MatrixVectorMul` Matrix A is stored using a row major order.

Your tasks are the following:

a) Implement a Matrix Vector multiplication kernel for the above Matrix stored in row major order.
b) Implement a method to store the above Matrix in column major order and then modify the above Matrix vector multiplication kernel to handle matrix stored in column major order .
c) Compare run times between the above two implementations (**row major order vs column major order**) for different Matrix sizes and provide arguments for the differences/similarities between run times for these two implementations .

Exercise 2 (Matrix Multiplication for APSP, *5 Credits*)

Given a directed graph $G = (V, E)$ with distance function $dist : E \to \mathbf{R}, |V| = n$ where V is the vertices (or nodes) set and E edges set. The adjacency Matrix of the above given directed graph is defined as follows:

$n \times n$ matrix $A = (\delta_{ij})$ of edge distances :

$$\delta_{ij} = \begin{cases} dist(v_i, v_j), & \text{if } (v_i, v_j) \in E \\ \infty, & \text{if } (v_i, v_j) \notin E \\ 0, & \text{if } i = j \end{cases} \tag{1}$$

Given $D^1 = A$, compute a series of matrices $D^2, D^3, ..., D^{n-1}$ (generated by algorithm $EXTEND-PATH(D^{m-1}, A)$ see Algorithm 2), where $D^k = (d_{ij}^k)$ for $m = 1, 2, ..., n-1$ contains the shortest path distances between each pair of vertices v_i, v_j with at most k edges Then the final matrix D^{n-1} contains actual shortest path distances. Simple algorithm for computing D^{n-1} see Algorithm 1

The given `APSP` framework computes the matrix product of two matrices A (8×8) and B (8×8) and then resulting matrix $C = c_{ij}$ is displayed using color coding as follows.

$$color_{ij} = \begin{cases} (r = 0, g = 0, b = 0) & \text{if } c_{ij} = 0 \\ (r = 2 * c_{ij}/20, g = 0, b = 0), & \text{if } 0 < c_{ij} \leq 10 \\ (r = 1.0, g = c_{ij}/20, b = 0), & \text{if } 10 < c_{ij} \leq 20 \\ (r = 0, g = 1.0, b = 0), & \text{if } c_{ij} > 20 \end{cases} \tag{2}$$

Algorithm 1 Slow APSP Algorithm

procedure SLOW-APSP(A)
    $D^1 \leftarrow A$
    for $k \leftarrow 0$ `to` $n-1$ **do**
        $D^k \leftarrow$ `EXTEND-PATH`(D^{k-1}, A)
    end for
    `return` D^{n-1}
end procedure

Algorithm 2 Method for computing the next D matrix

procedure EXTEND-PATH(D, A)
    $D = d_{ij}$`is an` $n \times n$`matrix`
    for $i \leftarrow 1$ `to` n **do**
        for $j \leftarrow 1$ `to` n **do**
            $d_{ij} \leftarrow \infty$
            for $l \leftarrow 1$ `to` n **do**
                $d_{ij} \leftarrow$ `min`$(d_{ij}, d_{il} + \delta_{lj})$
            end for
        end for
    end for
    `return` D
end procedure

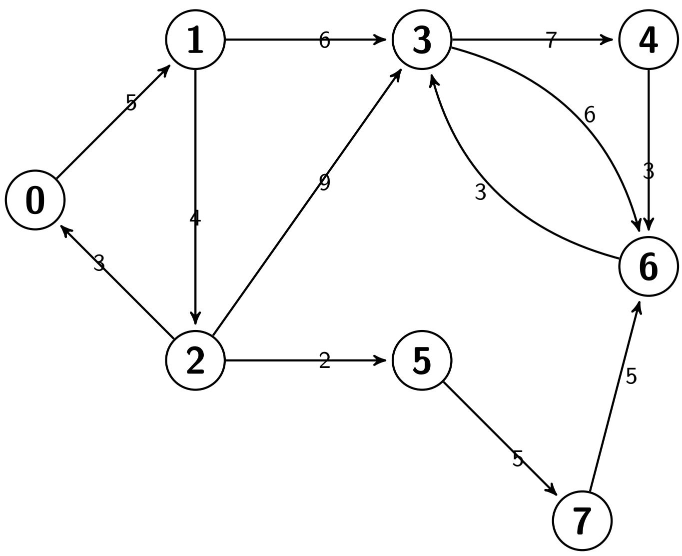

Figure 1: Graph

Your task is to modify the `APSP` framework in order to compute the shortest path matrix for the graph given above see Figure 1.You need `freeglut` for the framework. For installation please see the slides from the first tutorial.

Hint: Please note that the tiled version of Matrix Multiplication is used in the above given framework and use the similarities between algorithm EXTEND-PATH and Matrix multiplication algorithm to modify the above framework for computing All pairs shortest path as discussed in the lecture..