CS 255: Homework 5

This assignment is **optional**.
Due Friday December 8 at 11:59 pm

Instructions:

1. You may work in groups of up to 3 students.
2. Each group makes one submission on Canvas.
3. Include the name and SJSU ids of all students in the group.
4. You may discuss high level concepts with other groups, but do not copy other's work.

Problem 1

Give pseudocode for an efficient multithreaded algorithm that transposes an $n \times n$ matrix in place using divide-and-conquer to divide the matrix recursively into four $n/2 \times n/2$ submatrices. Analyze the work, span, and parallelism of your algorithm.

Solution. To avoid annoying edge cases, let's assume n is a power of 2. We want to represent the matrix, say M, as block matrix and exploit the following fact

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad \implies \quad M^T = \begin{bmatrix} A^T & C^T \\ B^T & D^T \end{bmatrix}.$$

The above fact leads to the following divide and conquer based approach to compute the transpose. Here, (i, j) is the top left corner of the matrix block and n is the dimension of the block. Calling Transpose$(M, 1, 1, n)$, transposes the entire matrix M.

Transpose(M, i, j, n)**:**
    if $n = 1$ **then**
        return
    $k \leftarrow n/2$
    spawn Transpose$(M, i, j, n/2)$
    spawn Transpose$(M, i, j + k, n/2)$
    spawn Transpose$(M, i + k, j, n/2)$
    Transpose$(M, i + k, j + k, n/2)$
    sync
    `// swap positions of blocks B and C`
    parallel for $i' \leftarrow i$ **to** $i + k - 1$ **do**
        parallel for $j' \leftarrow j + k$ **to** $j + n$ **do**
            swap $M_{i'j'}$ and $M_{j'i'}$

We have the following metrics.

- Work satisfies the recurrence: $T_1(n) = 4T_1(n/2) + \Theta(n^2)$; which has solution $T_1(n) = n^2 \log n$ by the master theorem.
- Span satisfies the recurrence: $T_\infty(n) = T_\infty(n/2) + \Theta(\log n)$; where first term accounts for the four parallel transpose calls and the second is the nested parallel for loops. The solution is $T_\infty(n) = \log^2 n$ using the result from the textbook/lecture slides.
- Parallelism is $T_1(n)/T_\infty(n) = n^2/\log n$.

Problem 2

The Floyd-Warshall algorithm computes shortest paths between all pairs of vertices in a weighted graph. Give pseudocode for an efficient multithreaded implementation of the Floyd-Warshall algorithm. (See section 25.2 of CLRS for more details on the algorithm).

Solution. Let $n = |V|$ and $w(i, j)$ be the weight of the edge between vertices i and j. We want to compute $dist[1 \dots n][1 \dots n]$, for all pairs of vertices i and j. Essentially, we just parallelize the two inner most loops of the standard single processor algorithm.

```
FW(V, E):
    n ← |V|
    parallel for i ← 1 to n do
        parallel for j ← i to n do
            if i = j then
                dist[i][j] ← 0
            else if (i, j) in E then
                dist[i][j] ← w(i, j)
            else
                dist[i][j] ← ∞
    for k ← 1 to n do
        parallel for i ← 1 to n do
            parallel for j ← 1 to n do
                if dist[i][j] > dist[i][k] + dist[k][j] then
                    dist[i][j] ← dist[i][k] + dist[k][j]
```

We have the follow metrics.

- Work is $T_1(n) = \Theta(n^3)$.
- The span of the nested parallel loops is $\Theta(\log n)$. Overall, with the outer for loop, the span is $T_\infty(n) = \Theta(n \log n)$.
- Parallelism is $T_1(n)/T_\infty(n) = \Theta(n^2/\log n)$.