Project 2 Milestone 2

By,

Joshua Standiford, Chris Sidell

Pseudocode:

int parallel\_lcs(X, Y, n, m):

int L[n + 1][m + 1];

for i = 0 to (n + m – 1):

int col = max( 0, i – n)

int size = min(i, m – col, n)

parallel for j = 0 to size:

int l = min(n, i)

int r = col + j

int x = l – j -1

if X[x] = Y[r]:

if x = 0 or r = 0

L[x][r] = 1

Else

L[x][r] = L[x – 1][r – 1] + 1;

Else:

If x = 0 and r = 0

L[x][r] = 0

Else if x = 0

L[x][r] = L[x][r – 1]

Else if r = 0

L[x][r] = L[x – 1][r]

Else

L[x][r] = max(L[x - 1][r], L[x][r – 1])

Return L[n – 1][m – 1]

Work Calculations:

The span calculation is based off of

Tinf (n) = lg(n) + max-iter(k)

When analyzing the pseudo-code above, we can see that the outer most for loop will take:

Theta( n + m – 1)

For the given run-time above. Given an n and m sufficiently large, we can reduce the theta( n + m – 1) runtime to be a linear runtime of theta( k ). Where k = (n + m – 1)

Within the algorithm there is constant time work we will denote as:

Theta( 1 )

Next we have the parallel for loop which diagonally iterates through the 2D LCS matrix. This inner-loop will have the complexity:

Theta( lg(n \* m) )

Since we will have the entire N x M matrix traversed through in parallel. That will be the max run time. So we’re left with the Span relation as follows:

Tinf = lg(n) + k + lg(n \* m)

Since we any linear time will always run faster than a logarithmic runtime. It is trivial to say that the span of the parallel algorithm will be:

Span = Tinf = Theta ( k )