Assignment #4

Low-level Programming

Due date: as specified on Moodle

1 Assignment

One way of representing a graph is to use an *adjacency matrix*. E.g.;

```
int G[N][N];
```

If N is the number of vertices in g, then G is an named $v_0, v_1, v_2, \ldots, v_{N-1}$. If `G[u][v]` is 1, it means that there is an edge from u to v. If the graph is directed, it is possible that `G[u][v]` is 1 while `G[v][u]` is 0, indicating the direction of the edge is from u to v. An undirected graph would require either `G[v][u]` or `G[u][v]` to be 1 to indicate an edge between 2 vertices.

The goal of the assignment is to optimize a function that performs the function of converting a directed into an undirected graph. The basic code is shown as below:

```
void basic_col_convert(int *G, int dim)
{
    int i, j;

    for (i = 0; i < dim; ++i)
        for (j = 0; j < dim; ++j) {
            G[j * dim + i] = G[j * dim + i] || G[i * dim + j];
        }
}
```

Your job is to optimize the code for the single-threaded case via minimizing cache misses and increasing the cache hit rate. You are not allowed to change the operation of logical OR || to bit-wise OR | to speed up.

You are given two files `main.cpp` and `clock.h`. Fill in your optimized code in function `opt_col_convert_single_threaded`. Your should submit only the `colconvert.cpp`.

2 Rubrics

The assignment is scored over 100 points.

- Optimization (50 points). Awarded only if your code is correct. The reason for the optimization has to be due to optimization of cache access, such as making use of spatial or temporal locality of reference. This part is graded by the given test cases in VPL environment.

- Documentation for optimization (40 points) - If you attempted to optimize but failed to, explain in comments what are the bottlenecks of the performance and how you have tried to address it. If you succeeded, explain what you have done to make it work and why it works. The documentation is included in your code.
- Coding style and indentation (10 points).