Assignment 3

路橙 Lu Cheng 2015010137

Problem 1:

```
-----------------------------------------
SCORES :
-----------------------------------------
grid1000x1000.graph         |   1.00000 / 1 |
-----------------------------------------
soc-livejournal1_68m.graph  |   1.00000 / 1 |
-----------------------------------------
com-orkut_117m.graph        |   6.00000 / 6 |
-----------------------------------------
random_500m.graph           |   6.00000 / 6 |
-----------------------------------------
rmat_200m.graph             |   6.00000 / 6 |
-----------------------------------------
TOTAL                       | 20.00000 / 20 |
-----------------------------------------
```

Problem 2:

```
-------------------------------------------------------------------------
SCORES :                    |   Top-Down    |   Bott-Up     |    Hybrid     |
-------------------------------------------------------------------------
grid1000x1000.graph         |   3.00000 / 3 |  3.00000 / 3  |  3.00000 / 3  |
-------------------------------------------------------------------------
soc-livejournal1_68m.graph  |   3.00000 / 3 |  2.09279 / 3  |  3.00000 / 3  |
-------------------------------------------------------------------------
com-orkut_117m.graph        |   0.88358 / 3 |  3.00000 / 3  |  3.00000 / 3  |
-------------------------------------------------------------------------
random_500m.graph           |   6.00000 / 6 |  4.44705 / 6  |  6.00000 / 6  |
-------------------------------------------------------------------------
rmat_200m.graph             |   6.00000 / 6 |  4.15887 / 6  |  6.00000 / 6  |
-------------------------------------------------------------------------
TOTAL                                                       | 56.58230 / 63 |
-------------------------------------------------------------------------
```

1. (1)Top_down:

   Synchronization:

```
#pragma omp critical
{
    memcpy(new_frontier->vertices + new_frontier->count, local_frontier, local_count * sizeof(int));
    new_frontier->count += local_count;
}
```

   Ensure the local_frontier to be able to be added in new_frontier.

Limit overhead:

```
#pragma omp parallel private(local_count)
    {
        local_count = 0;
        Vertex* local_frontier = new Vertex [g->num_nodes];
#pragma omp for
```

Let the thread compute a number of vertexes, rather than compute one vertex, because adding vertex into new_frontier is a kind of overhead.

(2)bottom_up:

Synchronization:

No synchronization because neither of 2 threads write into the same memory.

Limit overhead:

Didn’t do that.

2. Yes.

The sparse graph uses top_down(|V| / |E| < 30), the opposite uses bottom_up.

Besides, if the graph is too sparse that |V| / |E| < 5, I simply use top_down that computes every node in frontier in parallel rather than compute a gourp of nodes. It works very well in grid1000x1000.graph.

3. I think it is beacause of the memory bandwith.