PDC 2024 Spring Homework 2 report

寿晨宸 2100012945

Dijkstra

1. 重写了 Dijkstra 算法。因为 priority_queue 很难并行化，所以我最后使用了一个数组用于记录已访问的节点，虽然牺牲了一些复杂度，但更容易并行。
2. 初始化 visited 和 sol->distances 数组时并行。
3. 在寻找距离源节点最近的未访问节点 u 时，我将 n 个节点分块处理（分块数 BLOCK_NUM 可以调参，经测试 256 较佳），用 `#pragma omp parallel for` 并行地找出每个块内距离源节点最近的未访问节点，并使用数组记录信息。因为各个块之间没有任何依赖，也没有对变量的竞争，所以可以不使用 critical 而充分并行。完成分块处理之后，再处理之前存在数组里的结果，这部分的开销是小常数的，虽然是串行，但可以容忍。
4. 更新与节点 u 相连的节点的距离时，因为循环各部分间无依赖，所以可以用 `#pragma omp parallel for` 充分并行。

Bellman-Ford

1. 并行版基于串行版改动而来。
2. 初始化 sol->distances 数组时并行。
3. 虽然循环体内部是有依赖的，但只要在第一层循环和第二层循环外各添加 `#pragma omp parallel for`，就能做到保持正确。（经检验确实是正确的）

Performance

grid10x10

```
------------------------------------------------------------
Your Code: Timing Summary
Threads  Par-Dijsktra           Par-Bellman-Ford
   1:    0.000 (1.00x)          0.000 (1.00x)
   2:    0.001 (0.40x)          0.000 (1.39x)
   4:    0.001 (0.33x)          0.000 (1.43x)
   8:    0.002 (0.26x)          0.000 (1.62x)
  16:    0.003 (0.14x)          0.001 (0.26x)
  32:    0.005 (0.08x)          0.002 (0.16x)
  64:    0.009 (0.05x)          0.002 (0.17x)
 128:    0.014 (0.03x)          0.002 (0.16x)
------------------------------------------------------------
```

grid100x100

```
------------------------------------------------------------
Your Code: Timing Summary
```

```
Threads  Par-Dijsktra           Par-Bellman-Ford
   1:    0.161 (1.00x)          0.779 (1.00x)
   2:    0.092 (1.74x)          0.276 (2.82x)
   4:    0.078 (2.06x)          0.214 (3.64x)
   8:    0.074 (2.16x)          0.111 (7.05x)
  16:    0.103 (1.56x)          0.058 (13.36x)
  32:    0.170 (0.94x)          0.031 (25.38x)
  64:    0.234 (0.69x)          0.036 (21.44x)
 128:    1.607 (0.10x)          0.024 (32.43x)
-----------------------------------------------------------
```

grid1000x1000

```
-----------------------------------------------------------
Your Code: Timing Summary
Threads  Par-Dijsktra           Par-Bellman-Ford
   1:    1014.022 (1.00x)          5249.658 (1.00x)
   2:    520.457 (1.95x)          2627.619 (2.00x)
   4:    270.765 (3.75x)          1320.375 (3.98x)
   8:    145.722 (6.96x)          661.286 (7.94x)
  16:    73.190 (13.85x)          336.807 (15.59x)
  32:    55.008 (18.43x)          180.810 (29.03x)
  64:    55.420 (18.30x)          116.701 (44.98x)
 128:    66.102 (15.34x)          129.517 (40.53x)
-----------------------------------------------------------
```

结论

根据实验结果，可以得出，在较大的图上，Dijkstra 和 Bellman-Ford 算法都实现了较为可观的加速比，其中因为 Bellman-Ford 的并行化更彻底而 Dijkstra 中仍含一部分串行部分，所以前者的效果更好。

同时，因为并行化版本的复杂度具有较大的常数项（openmp 和线程切换带来的额外开销），所以在较小的图，如 grid10x10,tiny,grid4x4 乃至 gird100x100 上，并行化反倒会导致性能的下降。

总体而言，实验效果符合预期。