作品信息

作品名称	最短路径问题
待求解的问题	
基于 CUDA 的单源点最短路径并行算法。利用 GPU 的并行性，在经典的单源点方法上挖掘出并行的方法，进一步提升算法的速度。作品基于 Dijkstra、Bellman-Ford、Delta-Stepping 和 Sparse Matrix-Vector Bellman-Ford 实现了 CUDA 并行算法。	
使用的算法	
1) CUDA Dijkstra 本算法根据经典的 Dijkstra 方法改进而来。在提取最近距离点和松弛阶段分别用到了并行性，进一步提升了算法的效率。 **2) CUDA Bellman-Ford** Bellman-Ford 的最短路径算法不同于 Dijkstra，它的核心思想是基于一个队列中的节点进行松弛。而队列中的节点来自于每次松弛过后到源节点路径变小的节点。 **3) CUDA Delta-Stepping** Delta-Stepping 的基本思想是利用 Bucket 保存第 i 步的 i*delta 和(i+1)*delta 中的松弛节点，并对其松弛。 **4) CUDA Sparse Matrix-Vector Bellman-Ford** 利用稀疏矩阵运算，对松弛函数进行改进的并行 Bellman-Ford 算法。 （具体算法可参考 doc 中的论文）	
编程和优化技巧	
对于 Dijkstra 和 Delta-Stepping 各自都用到了 Reduction 的技术。Bellman-Ford 则用到了循环队列的思想。另外对于 Delta-Stepping 我进行了适用于 CUDA 的改进。Sparse Matrix-Vector Bellman-Ford 运用了稀疏矩阵的思想加速松弛操作的速度。	
与传统的 CPU 开发的程序相比达到的加速比	
本作品与 Boost 库中的串行算法 Dijkstra、Bellman-Ford，以及并行算法 Delta-Stepping、Crauser 进行比较。具有较强的竞争力。具体的速度分析请参考论文中的实验部分。	
补充信息（可以放入任何与此作品相关的说明信息）	1）由于本作品的显卡采用 Nvidia GeForce GTX 550 Ti（1G 显存），与比赛的测试显卡 Nvidia GeForce GTX 580 有一定差距。有些算法（如 CUDA Delta-Stepping）对显存有较大需求，所以对于大地图不能很好的发挥算法性能。对于比赛测试显卡，算法应该能运行的更快，且对大地图的支持更好。 2）程序假定 GTX 580 在设备 0 中，代码中对设备 0 进行了参数查询。如 GTX 580 不在设备 0 中，请做相应更改。