CUDA 编程校园编程竞赛指定题目

最小生成树(Minimal Spanning Tree)问题的 GPU 加速

随着《潜伏》等谍战电视剧热播，以余则成为代表的地下党特工在银幕上纵横捭阖，煞是热闹。然而，真正的余则成恐怕更多考虑的是这样一个问题：怎样在保证与上下级联系畅通的前提下，最小化他的特工组织暴露的风险。如果余则成也是一位计算机科学家的话，他会把特工组织抽象为一个网络，组织中的每一位特工都是网络中的一个节点。然后，余则成需要确定一个最佳的联络方案，这里既要保证网络中的每一对节点都能够存在通信渠道(有可能是间接的，即通过别的特工)，又要保证通信渠道的数量最小[1]。以上第一条要求保证了上级的命令能够传达到每一位情报员，而每一位情报员收集的情报也能够汇总到上级。同时，第二条要求则保证不存在冗余的联系渠道，否则每增加一条渠道就也就增加了暴露的机会。谍战片里常说的“单线联系”，实际上就反映了树(计算机科学中的树)的一个重要特性，即任意两个节点只有一条通路。

实际上，余则成还真是“摊上大事儿了”，因为他面临的是计算机科学经典问题之一，最小生成树(Minimal Spanning Tree，MST)。我们首先来定义生成树(spanning tree)。

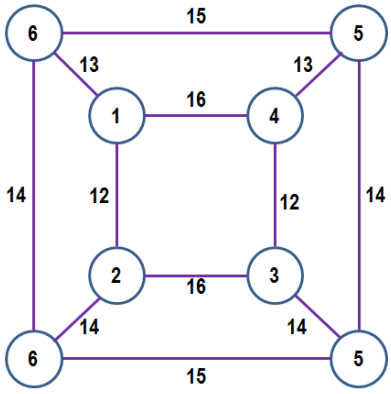

图 1. 一个图的例子

生成树 给定一个连通图(connected graph)，G=(V, E)，其中 V 为节点组合，E 为边组合(每一条边用两个节点定义)，满足以下条件的子图(sub-graph)称为一个生成树：

[1] 本文的写作借鉴了 J. Eisner, “State-of-the-Art Algorithms for Minimum Spanning Trees∗ A Tutorial Discussion”一文的类比。

1. 是 G 的连通子图；
2. 没有循环(即从 1 个节点只有 1 条路径到达任意其它节点)；
3. 满足树的条件(边的个数为节点个数减 1)
4. 包含 G 的全部节点。

图 1 是一个图的例子，其中一共有 8 个节点，分别编号为 1 到 8。该图还有 12 条边，每条边都有相应权重，标记在边的旁边。权重可以是相应节点之间的距离，或者代表其它物理含义。图 2 中用红色粗线标出的是图 1 例子的一个生成树，包括全部节点和 7 条用红色标出的边。显然，一个图可以有多个生成树，其中所有边的权重之和最小的一棵就是最小生成树。图 3 是图 1 中例子的最小生成树。最小生成树在网络(计算机、通信、道路和电力等网络)规划、集成电路设计、图像处理、计算生物学等科学和工程领域都有重要应用。

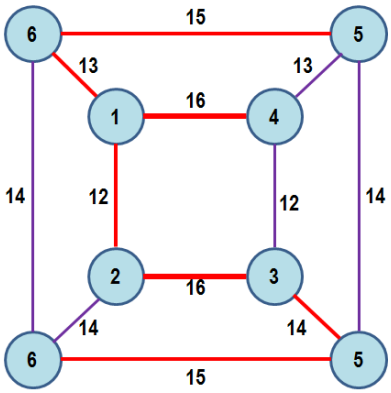

图 2. 生成树的例子

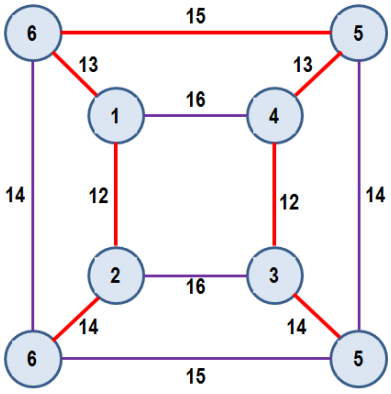

图 3. 最小生成树的例子

解决 MST 问题的算法很多，参考文献[1]给出了全面的综述，而《算法导论》[2]对该问题的一些经典方法如 Kruskal、Prim 和 Borůvka 算法有着很好的阐述。

输入：图G=(V, E)

输出：最小生成树T

1. 把所有边按权重以权重递增顺序存放到数组L中
2. 初始化T为空集，即T=Φ
3. 从L中选取权重最小的边e，如果e加入T不导致循环，则将e与T合并
4. 重复3直至T覆盖G的所有节点

图 4. Kruskal 算法伪代码

图 4 是 Kruskal 算法的伪代码。该算法思路很简单，是贪婪算法保证找到优化解的经典例子。首先对所有边按权重排序，存放在数组 L 中；然后从空集开始，每次从数组 L 中选择权重最小并且不产生循环的边加入集合，直至覆盖全部节点。Kruskal 算法的瓶颈是排序，对边数量远大于节点数量的图(稠密图)的效率较差。同时，其贪婪式优化的本质不利于并行。文献[3]提出了一种改进的方法，称为 Filter-Kruskal 算法，该算法采用递归结构，把图不断划分为更小的子图，并且过滤掉肯定不能进入最小生成树的边，在子图小到一定规模后则调用传统的 Kruskal 算法。

图 5 是 Prim 算法的伪代码。该算法从任意节点开始，每次从与现有节点集合连接的边中选择权重最小、并且不产生循环的边加入集合，直至覆盖全部节点。Prim 算法的稠密图的效果较好，但是也属于贪婪算法，需要选择满足条件、权重最小的边，因此本质上不利于并行。

输入：图G=(V, E)

输出：最小生成树T

1. 初始化T，包含任意单一节点
2. 找出与T中节点连接的、权重最小的边e，如果e加入T不导致循环，则将e与T合并
3. 重复3直至T覆盖G的所有节点

图 5. Prim 算法伪代码

Borůvka 算法在本质上更容易并行，现有基于 GPU 的 MST 构造方法(如[4]

和[5])都使用了该算法。Borůvka 算法的伪代码如图 6 所示。该算法从各个节点开始，以聚类(clustering)方式生长出最小生成树。起步时，所有节点都作为一个独立的联通子图(component)，大家分别把权重最小并且不产生循环的边合并进来。接下来，每个子图会“坍缩”为一个节点，该节点继承了其中所有原来节点与子图外部节点之间的边。Borůvka 算法在这些坍缩节点上继续进行以上操作，直至只有一个唯一的联通子图。Borůvka 算法的并行性一目了然，但是坍缩节点并在这些节点上建立新图的操作比较复杂，已经成为现有 GPU 实现(如[4]和[5])的主要瓶颈。

```
输入：图G=(V, E)
输出：最小生成树T
1. 初始化T，包含全部节点，每一节点作为一个联通子图(component)
2. While T中包含多余一个联通子图(component)
3.     For each component C of T
4.         初始化边集合S，S=Φ
5.         For each 节点 v in C
6.             找到与v连接的、权重最小的、并且不在S内的边e，加入S
7.         把S中权重最小的边加入集合T
8. T即为最小生成树
```

图 6. Borůvka 算法伪代码

除了以上三种算法外，参考文献[6-9]也是代表性的 MST 算法，但是相对复杂，欢迎大家探索。其中文献[8]使用了随机化思想，可能为并行化引入新的机会，值得深入挖掘。

本次 CUDA 竞赛希望大家针对 GPU 设计高效的 MST 算法和代码。选手可以使用第 9 届 DIMACS 算法暨编程竞赛[10]和 Graph 500[11]的图作为输入，前者是欧美国家的道路网形成的图(比较稀疏)，后者是随机图。评委会使用其中一部分对参赛作品进行评估，评分标准同时考虑算法的创新性和运行性能。

参考文献

1. J. Eisner, State-of-the-Art Algorithms for Minimum Spanning Trees - A Tutorial Discussion,” www.cs.jhu.edu/~jason/papers/eisner.mst-tutorial.pdf, 1997.

2. T. H. Cormen, C. E. Leiserson, R. L. Riverst, and C. Stein, Introduction to algorithms, 2nd Edition, MIT Press, 2001.
3. V. Osipov, P. Sanders, and J. Singler. TheFilter-Kruskal minimum spanning tree algorithm. In ALENEX, pages 52–61, 2009.
4. Harish, P., Narayanan, P. J. 2007. Accelerating large graph algorithms on the GPU using CUDA. In Proc. of High Performance Computing – HiPC. 197-208.
5. V. Vineet, P. Harish, S. Patidar, P. J. Narayanan, Fast minimum spanning tree for large graphs on the GPU. In Proceeding of ACM SIGGRAPH High Performance Graphics. 2009.
6. Michael L. Fredman and Robert E. Tarjan. 1987. Fibonacci heaps and their uses in improved network optimization algorithms. Journal of the ACM, 34(3):596–615.
7. H. N. Gabow, Z. Galil, T. Spencer, and R. E. Tarjan. 1986. Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica 6:109–122.
8. D. R. Karger, P. N. Klein, and R. E. Tarjan. 1995. A randomized linear-time algorithm for finding minimum spanning trees. Journal of the ACM, 42(2):321–328.
9. Greg Frederickson. 1985. Data structures for on-line updating of minimum spanning trees, with applications. SIAM Journal of Computing, 14(4):781–798.
10. 9th DIMACS Implementation Challenge - Shortest Paths, http://www.dis.uniroma1.it/challenge9/download.shtml.
11. Graph 500 Benchmark, http://www.graph500.org/specifications.