计算机算法设计与分析大作业

1 产生随机图

对于问题(a)， 我们可以基于邻接表直接构造一个有m_0个节点的环。循环m_0次，每一次对当前结点添加相连的两个结点即可。

对于问题(b)同理，循环t次，每次连接m条边。相关的概率问题可以通过计算机自带的产生随机数的函数产生一个小于图的总度数的数， 之后与结点的度数计算比较即可选择相连的结点。

```
pseudocode:
   for 0 to m0-1:
      ni link to ni+1
      ni link to ni-1
      graph_list.push(ni)

   for m0 to m0+t:
      calculate the degree of node
      for 0 to m:
         create the random num
         if(num < total_degree[node]) //choose the node
           link node with the new node
           change the graph_list
```

2 实现

上传的文件中graph.h为几个类的声明， 相应的实现在graph.cpp中。在linux环境下使用./start.sh即可编译并运行， 运行时需要输入t的值。

graph. h中， graph类对应图， vertex类对应算法产生随机图时使用的结点， node类对应单源最短路径时使用的结点。 由于使用优先队列，

node中重载了操作符<。

在main.cpp中， graph_random_produce()函数是用来产生随机图的。graph_show()函数使用邻接链表的形式打印随机图。

graph_unweighted()函数是对于所有边权重相等的图进行单源最短路径的函数，graph_weighted()则是可以针对权重不等的图进行处理的函数。两者使用的都是dijkstra算法。

3 单源最短路径

程序输出实例如下方所示：

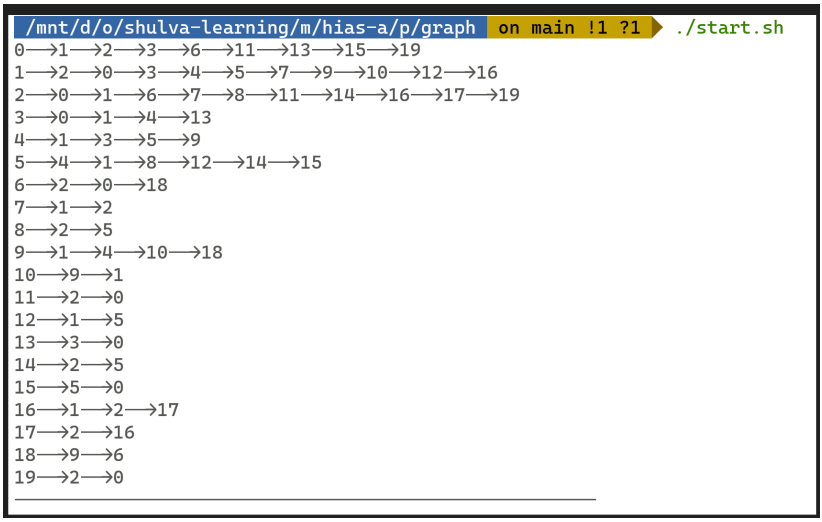

图 1. 随机图

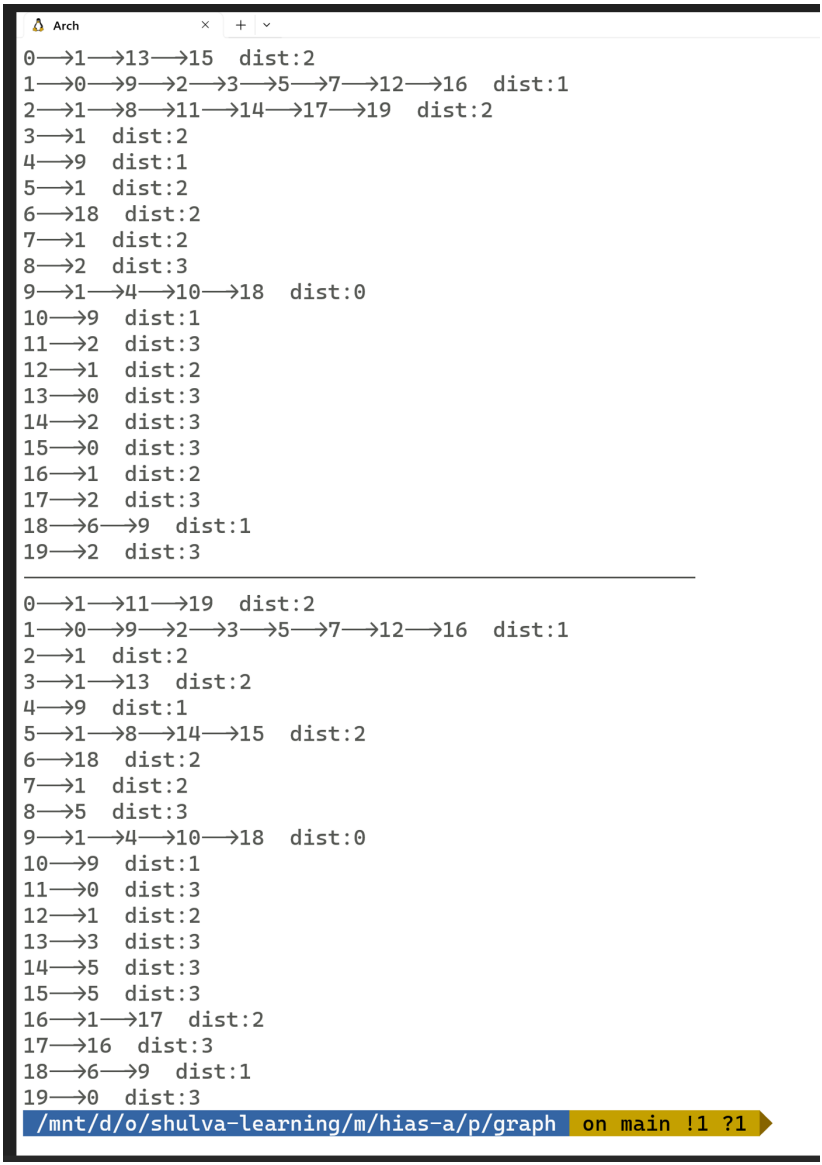

图 2. unweighted与weighted共同测试同一张图

理论上对于所有边权重相等的图，单源最短路径算法生成的树应当与

广度优先搜索生成的树一致。不过最短路径树不唯一。

如上图所示，算法使用邻接表的形式表示树。两个函数所产生的最短路径树便不唯一，但是可以看到每一个从此结点到源节点的最短路径的长度都是相等的。

4 不同t值测试效率

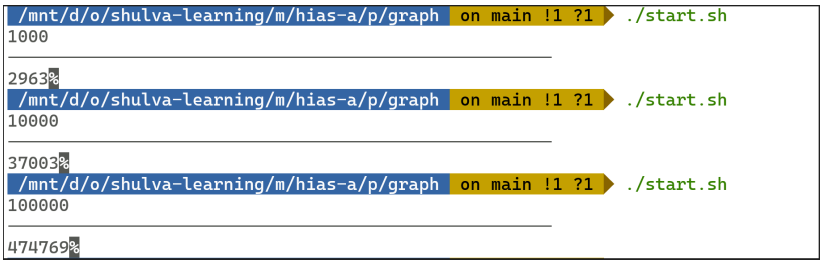

图 3. 运行时间

代码通过c++的clock()函数来计算时间。注释掉图和树的输出代码后结果如上图所示。可见其的运行时间大致符合使用优先队列优化的dijkstra算法的复杂度$O((|E|+|V|)\log|V|)$。