实验设计

想法：

1. 首先我们对图的分类可以根据节点数的大小进行分类。
2. 然后我们考虑在N个节点的图中，我们考虑边的数量，这里我们比较容易想到两种特殊的图：
3. N个节点的最小连通情况 – 树
4. N个节点之间相互都连有边 – 完全图

他们分别代表了稀疏图和稠密图的极端情况。

那么我们想到可以通过加边的形式来从树变成完全图，这里我们根据prim和Kruskal算法集中式和分布式地构建其最小生成树的方式，想到了两种方法：

1. 平均的在图上加
2. 集中的在图上加

这样我们就可以生成两种稠密类型的图，看在这些图上两种算法可以带来怎样的效果。

1. 同时我们还观察到排序算法中，对边权的值可能也会有差别，因此我们将边权设置成了三种类型来进行实验：1. 等权 2. 排列权值 3. 随机权值

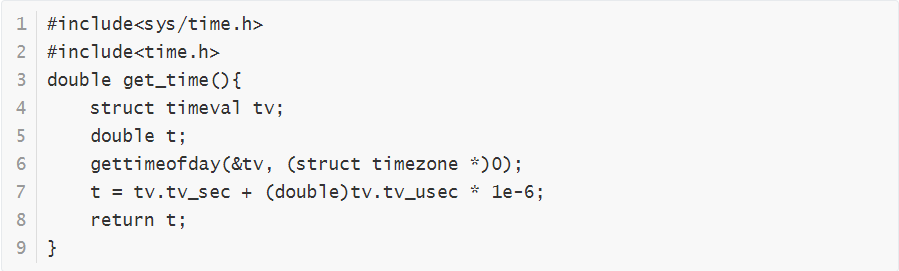
操作：

环境：ubuntu 8核

Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-142-generic x86\_64) 48核

在我们的实验中，我们设计了生成边权的程序，生成图的程序，运行实验的脚本，以及实验用的Kruskal和Prim程序。

我们的测速使用的是<sys/time.h>中的gettimeofday()函数，可以精确到微秒(1e-6 s)，

我们没有计算输入输出的耗时，只对核心代码部分进行测速。每个图让程序执行10次，将运行时长取平均值。

我们创建图的步骤分为两步，第一步是生成边权，第二步是创建图

1. 生成边权

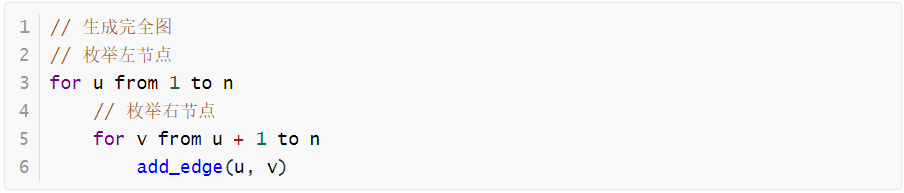
小边权中等权和随机直接输出，排列用数组生成然后random\_shaffle

1. 创建图

分为两种：

1. 一类是只探究树和完全图的极端情况，通常是因为N太大，不适合进行分步加边，在这种情况下我们使用的是vector进行存储，只有vector才能支持N=20000，M=199990000的存储。

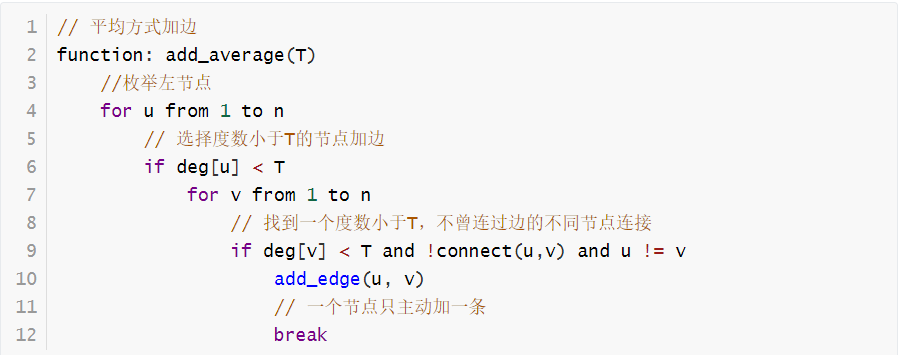
这种情况下只需要读入边权，然后构建一棵树输出，完全图则只需要两个for循环判断两个节点不相同就相连就可以



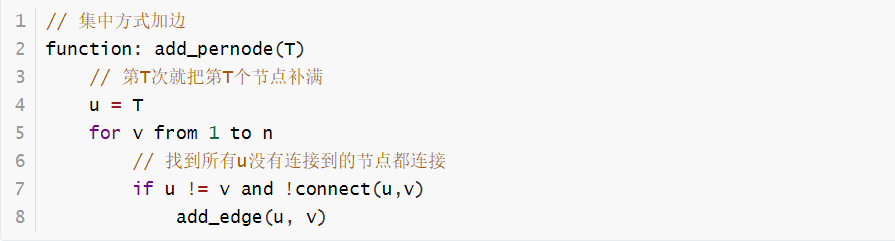
这里构建树的方法非常简单，即对第i个节点，随机一个节点j (j < i)，然后将他们连通，这样就能保证每个节点连通，而且没有环，因为前面的节点不可能有回边



1. 第二类是探究逐步加边的方法，我们先构建一棵树，并且用邻接表记录这棵树，然后使用两种方法逐步加边：
2. 平均加边：我们从每个点的度数入手实现平均加边，因为最后一个完全图的所有点的度数都为n-1,那么在加边的第T轮，我们只给度数<T的节点之间连边，这样就可以让整个图平均加边，这样只需要进行n-1轮，所有的节点就都互相来都有边了



1. 集中加边：我们从每个节点入手，直接让这个节点与所有的节点连边，这样我们执行n-1轮，也就让这个图变成完全图了。



三、进行实验：我们为了方便实验，编写了测试脚本，根据设置的节点大小，自动执行我们的测试任务。

从脚本中也可以看到我们枚举的各个方面：

枚举节点N

根据N得到M

并根据M生成边权

枚举不同的边权

根据当前边权生成图

枚举不同的图

枚举进行的轮数

根据当前的图执行prim

计算平均值

枚举进行的轮数

根据当前的图执行kruskal

执行平均值

