1. 主要算法：

1.1 最大流最小费用算法

来源于运筹学中的最大流算法，给定图G(V,E,C)，V和E代表顶点集和边集，C代表每条边的容量，要求在每一条边上布置流量大小与方向，除了源点和汇点外其他顶点的输入等于输出，在以上的条件约束下使得流量最大。最大流最小费用问题是该问题的变种，在保证最大流的前提下每条边给定了一个流量单价，问题由最大流问题转化为使总价最小的问题。

最大流问题的算法在算法导论26章有具体阐述，主要思路是在残量网络上搜索可行流直到搜索不到可行流为止。为了保证每次搜索到的费用最小，将边的权值设置为单价，反向边的单价设置为负值。如此一来每次运行最短路径算法都可以搜索到费用最小的可行流，这次比赛使用的也是这一个思路。但是最终效果与最佳方案有一定偏差，特别是小规模搜索。

对于多源多汇的情况设置超级源点和超级汇点可以解决，对于无向图，可以通过增加节点转化为有向图，具体流程可以参考算法导论第26章。

1.2 最短路径搜索算法

最先使用的算法是dijkstra算法，该算法本质上是一种广度优先的搜索方式，使用最小堆存储中间节点可以提升算法的效率，该算法是不适用于负权值情况的，在1.1提到的算法中，负权值是一定会出现的，但是根据实验效果，dijkstra算法可以寻找到可行解，甚至在小规模的搜索中，dijkstra算法搜索的可行解优于SPFA算法，dijkstra算法的运行效率和稳定性也一定是优于SPFA算法的。

SPFA算法全称是Shortest Path Fast Algorithm，该算法用于处理带负权值的最短路径搜索问题，其中的松弛操作在我的理解中依然是使用广度优先，因为该算法的实现离不开队列这种数据结构，通过对队列的改进可以进一步提升该算法的效率，与dijkstra算法相比唯一的优势就在于解决负权值问题，理论上的时间复杂度比不上dijkstra算法，在最大流最小费用算法中使用必须遍历到整张图，而dijkstra算法只需要确定目标点的最短路径就可以提前结束，因此稳定性上SPFA算法被dijkstra算法完爆。在实验过程中可以明显感觉到SPFA算法执行时会出现卡顿现象。

1.3 启发式搜索

启发式搜索算法用于搜索服务器部署的位置，待选的算法如下：模拟退火、遗传、蚁群、粒子群、禁忌搜索。前两者的实现较为简单。启发式搜索的主要问题在于初始情况与参数的设置。特别是遗传算法的参数较多，例如种群的规模、交叉概率、变异概率、选择方式等。同时遗传算法极有可能陷入局部最优解，因此参数可以随着世代增加而改变，即自适应的遗传算法，遗传算法主要是选择、交叉、变异三个过程的不断循环，其中的细节非常之多，参数和方式的设置也没有固定的思路，这也是启发式搜索面临的问题。

2. Java实现细节

2.1 Collection类

该类是使用非常频繁的类，List相当于长度可变的数组，Map存储键值对，List还有队列和优先队列的实现，Collection的sort方法用于排序。可以说几乎所有的数据结构Java已经实现好了。但是List又分为ArrayList、LinkedList，Map分为HashMap、TreeMap。虽然操作大同小异，不同的数据结构对应相同操作的效率差别很大，这一点只能在使用过程中具体摸索，现在已经清楚的是可以把List看作线性表，ArrayList理解为顺序表，LinkedList理解为链表，这两者的差异套用数据结构的知识就行了，需要说明的是，如果能够用数组解决问题那么尽量使用数组，数组的效率比上述所有对象的效率都高。Map和Set的区别只能够在以后用到时具体分析。

2.2 序列化( Serializable )深复制

这是Java里面深复制的一种简易方式，对于小型对象非常好用，具体实现时要注意序列化的类成员变量如果是对象也必须实现序列化，常用的Collection类实现了序列化。但是当对象较大并且需要多次深复制时，重写clone方法是更好的选择。

3. 展望一下

44名感觉还行，反正准备明年再来一次，从刚开始上不了榜到之后想冲进32，这个过程还是很有意思的。感谢凡哥不停调试与提交代码。最后没有hold住的原因总结如下：1.编程语言的问题，Java跑的就是比C++慢，最大规模一次费用流可能需要500ms，C++讲道理只需要50ms，启发式搜索的次数如果是别人的1/10那劣势就很大了。2.费用流算法的问题，在给出最佳服务器分布的情况下得不到极限费用，其中的原因至今都未找到，最短路径算法是一个因素，费用流算法本来是应用于有向图，但是我们强制应用到无向图，本身的正确性没有验证，这可能是一个隐患。3.费用流算法是绝对能够改进的，最大流问题有更好的算法，算法导论26章中称为推送-重贴标签算法，甚至还有前置重贴标签算法，我们使用的图顶点数目远小于边的数目，因此推送-重贴标签算法在解决最大流问题上更具优势，以下是截取算法导论中的原文：

***到目前为止，许多渐近效率很高的最大流算法都是推送-重贴标签算法，最大流算法的最快实现也是基于推送-重贴标签方法。推送-重贴标签方法还能有效解决其他流问题，如最小成本流问题。***

有理由猜想最小费用流问题有更快捷并且更加准确的实现方法，有的队伍可以在10ms内完成一次费用流问题，这也佐证了猜想。（推送-重贴标签算法值得研究并且具体实现一下）

以后想做的工作：Java多线程、多态、推送-重贴标签算法具体实现（算法导论没有给出完整的算法，完成26章的习题或者查阅参考文献）、推送-重贴标签算法的最小费用流版本。