3.1节对应重要函数：（预计算距离）

Reader::calcClusterDistances：构建聚类间距离矩阵

Hausdorff::calcDistance函数：计算聚类间距离

3.2节对应重要函数：（构建可行初始解在cluster层）

BinPacking::run 装箱算法构建初始cluster层面可行解（满足容量约束前提）

ClusterSolution::getClosestFeasibleVehicle：找到当前cluster对应最近的车辆（每个车辆最后一个类）

参数randConstructProb：判断选择closest或仅按demand排序

3.3节对应重要函数：（如3.2无法获取可行初始解，在cluster层面进行聚类的Redistribution）

BinPacking::run 装箱算法构建初始cluster层面可行解（满足容量约束前提）

Redistribution::run 执行swapClusters操作（不同车辆间的cluster互换）（满足车辆容量）

Redistribution::swapClusters 具体clusters互换操作 含Move::swap

3.4节对应重要函数：（强化阶段/局部搜索）

CluVNS::run 3.4.1

NodeSolution\* ClusterSolution::convert 3.4.2

NodVNS::run 3.4.3

3.4.1节对应重要函数：（在cluster层通过7个intra/inter算子优化路线）

CluVNS::run ：VNS1算法

CluVNS::findNeighbour: 选择7个领域算子中的一个

CluVNS::7个算子的具体操作

3.4.2节对应重要函数：（将VNS1获得的最优cluster线路转换为customer对应的完整线路）

NodeSolution\* ClusterSolution::convert ：Convert算法

参数Params::RANDOM\_CONVERSION：判断最近/随机方式选择cluster内部点顺序

3.4.3节对应重要函数：（在customer层通过8个intra/inter算子优化路线）

NodVNS::run ： VNS2算法

NodVNS::findNeighbour: 选择8个领域算子中的一个

NodVNS::8个算子的具体操作

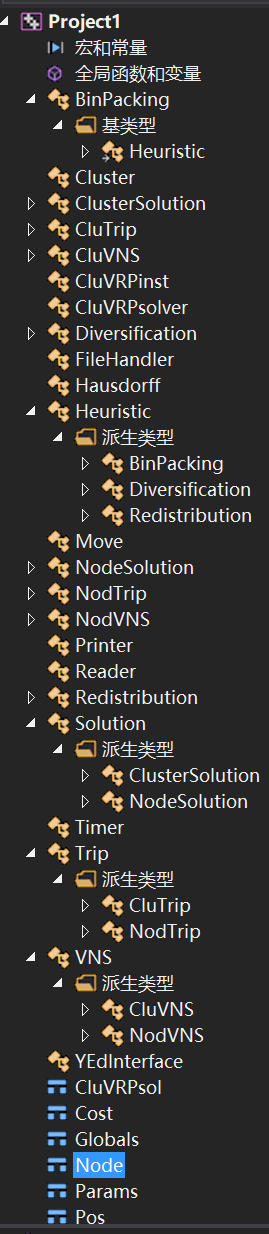
3.5节对应重要函数：（先评估后多样化阶段）

NodeSolution::evaluate ：评估是否有改进

Diversification::run ： perturbation+repair

Diversification::perturbationCluster

Diversification::repairCluster



1输出CluVRPinst\* cluVRPinst

cout << cluVRPinst;

2 CluVRPsolver：包含解: sCluCurret （ClusterSolution）and sNodCurrent(NodeSolution)

3 输出Solution：（ClusterSolution + NodeSolution）

包含cluVRPinst和totalDist + vTrips（cluster和node不同）

class Solution; //基础solution类,包含算例cluVRPinst\_和总距离totalDist\_

class ClusterSolution; //继承后的solution,增加CluTrip

class NodeSolution; //继承后的solution,增加NodTrip

Solution中，计算总距离totalDist\_:

totalDist\_ += vTrips\_.at(i)->calcDistance();

Trip类：CluTrip and NodTrip

Dist\_ size\_ totalDemand\_ 加 vClusters and vNodes

增加instances算例，选项8：简单自建cluster，类似选项0

增加showSol函数：方便对solution结果的输出展示

增加inst展示：方便对inst的输出展示