Java实现：

**private void backtrack(int i) {//i为初始深度**

**if (i > n) {**

**//TODO**

**} else {**

**//TODO**

**}**

**}**

**public boolean check(int pos) {**

**//TODO**

**return false;**

**}**

public int computeLB(Vector<Integer> liveNode, int level, int[][] cMatrix)

{

//TODO

return -1;

}

public int bb4TSP(int[][] cMatrix, int n)

{

//构造初始节点

Vector<Integer> liveNode = new Vector<Integer>() ;//城市排列

for(int i = 1; i<=n; i++) liveNode.add(i);

int level = 1;//0-level的城市是已经排好的

int lcost = computeLB(liveNode, level, cMatrix) ; //代价的下界

while(level != n)

{

//TODO

//参考优先队列，不停扩展节点,选取下一个节点

}

return minCost;

}

将“AlgorithmAnalyse--工程文件”解压后，导入Eclipse,实现Experiment2下的Back4TSP.java里面的**backtrack和check**方法，BB4TSP.java里面的**computeLB和bb4TSP**方法。实现后，运行Back4TSPTest.java，testBB4TSP.java测试文件，检测正确性，也可以添加更多的用例进行测试。

实现后，构造随机案例的产生方法，比较实现的两个方法。