TSP.java

```java
package tsp;

import java.util.List;
import myLib.utils.Utils;

/**
 * TSPをsimulated annealingで解く
 *
 * @author tadaki
 */
public class TSP {

    private final int N;//都市数
    private final double d[][];//都市間の距離
    private final List<Integer> path;//経路
    private double beta = 0.;//温度の逆数

    /**
     * コンストラクタ
     *
     * @param N 都市数
     * @param d 都市間の距離
     */
    public TSP(int N, double[][] d) {
        this.N = N;
        this.d = d;
        path = Utils.createList();
        for (int i = 0; i < N; i++) {
            path.add(i);
        }
    }

    /**
     * 経路の距離
     *
     * @return
     */
    public double evalLength() {
        double len = 1.;
        for (int i = 0; i < N; i++) {
            int j = (i + 1) % N;
            len += d[path.get(i)][path.get(j)];
        }
        return len;
    }
```

TSP.java

```
    /**
     * 部分経路を反転する
     *
     * @return
     */
    public double oneFlip() {
        int p[] = Utils.createRandomNumberList(N, 2);
        return oneFlip(p[0], p[1]);
    }

    /**
     * 部分経路を場所を指定して反転する
     *
     * @param p
     * @param q
     * @return
     */
    public double oneFlip(int p, int q) {
        //evaluate difference in length
        double de = -d[path.get((p - 1 + N) % N)][path.get(p)]
                - d[path.get(q)][path.get((q + 1) % N)]
                + d[path.get((p - 1 + N) % N)][path.get(q)]
                + d[path.get(p)][path.get((q + 1) % N)];
        if (de > 0) {
            if (Math.exp(-beta * de) < Math.random()) {
                return evalLength();
            }
        }
        return flipAt(p, q);
    }

    /**
     * 実際に場所を指定して反転する
     *
     * @param p
     * @param q
     * @return
     */
    private double flipAt(int p, int q) {
        List<Integer> subPath = Utils.createList();
        for (int i = p; i <= q; i++) {
            subPath.add(path.get(i));
        }
        for (int i = p; i <= q; i++) {
            int j = i - p;
            path.set(i, subPath.get(q - p - j));
```

```
        }
        return evalLength();
    }

    public void setBeta(double beta) {
        this.beta = beta;
    }

    public String showPath() {
        StringBuilder sb = new StringBuilder();
        path.stream().forEachOrdered(k -> {
            sb.append(k).append("->");
        });
        return sb.toString();
    }
}
```

TspMain.java

```
import java.io.BufferedWriter;
import java.io.IOException;
import myLib.utils.FileIO;
import tsp.TSP;

/**
 *
 * @author tadaki
 */
public class TspMain {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException {
        int N = 50;
        double d[][] = new double[N][N];
        int tOne = 1000;
        double beta = 0.001;
        for (int i = 0; i < N - 1; i++) {
            for (int j = i; j < N; j++) {
                double r = 10 * Math.random();
                d[i][j] = r;
                d[j][i] = r;
            }
        }
        TSP tsp = new TSP(N, d);
        tsp.setBeta(beta);
        int count =0;
        BufferedWriter out = FileIO.openWriter("tsp-out.txt");
        for (int i = 0; i < 20; i++) {
            for (int t = 0; t < tOne; t++) {
                for (int tt = 0; tt < N; tt++) {
                    tsp.oneFlip();
                }
                double len = tsp.evalLength();
                if(i>0){
                StringBuilder sb=new StringBuilder();
                sb.append(count).append(" ").append(len);
                out.write(sb.toString());
                out.newLine();
                }
                count ++;
            }
        beta *= 2;
```

```
            tsp.setBeta(beta);
            }
            out.close();
        }
}
```