Data Generations

The i values are subject to change, depending on the density size. Method to generate a random number within the given range

```
public int generateRandomInteger(Random r, int low, int high) {

    return r.nextInt(high - low + 1) + low;

}
```

Removing a vertex and their edges

```
//populating the incidence matrix with nodes
for(int i = 0; i < 2000; i++)
{
    addVertex(String.valueOf(i));
}

//generates random edges within the range of vertexes
for(int i = 0; i < 2000; i++)
{
    addEdge(String.valueOf(generateRandomInteger(r,0,2000)), String.valueOf(generateRandomInteger(r,0,2000)),
            generateRandomInteger(r,0,2000));
}

//creates a start time before removing the vertexes
starttime = System.currentTimeMillis();

//choosing a random node to remove
int k = generateRandomInteger(r,0,2000);


//removing the vertex
removeVertex(String.valueOf(k));


endtime = System.currentTimeMillis();

//returns the time taken to perform given task
System.out.println(endtime - starttime);
```

Retrieving K nearest neighbours

```
//populating the incidence matrix with nodes
for(int i = 0; i < 2000; i++)
{
    addVertex(String.valueOf(i));
}

//generates random edges within the range of vertexes
for(int i = 0; i < 2000; i++)
{
    addEdge(String.valueOf(generateRandomInteger(r,0,2000)), String.valueOf(generateRandomInteger(r,0,2000)),
            generateRandomInteger(r,0,2000));
}

//creates a start time before removing the vertexes
starttime = System.currentTimeMillis();

//choosing a random node to remove
int k = generateRandomInteger(r,0,2000);

//retrieving all k nearest neighbours
inNearestNeighbours(-1, String.valueOf(k));

endtime = System.currentTimeMillis();

//returns the time taken to perform given task
System.out.println(endtime - starttime);
```

Updating the edge of a random source node -> target node

```
//populating the incidence matrix with nodes
for(int i = 0; i < 2000; i++)
{
    addVertex(String.valueOf(i));
}

//generates random edges within the range of vertexes
String ran = String.valueOf(generateRandomInteger(r,0,2000));
String dom = String.valueOf(generateRandomInteger(r,0,2000));

addEdge(ran,dom,2);

for(int i = 0; i < 2000; i++)
{
    addEdge(String.valueOf(generateRandomInteger(r,0,2000)), String.valueOf(generateRandomInteger(r,0,2000)),
            generateRandomInteger(r,0,2000));
}

//creates a start time before removing the vertexes
starttime = System.currentTimeMillis();

//updating edge of source -> target with a weight of 5 (the weight shouldn't matter).
updateWeightEdge(ran,dom,5);

//retrieving all k nearest neighbours


endtime = System.currentTimeMillis();

//returns the time taken to perform given task
System.out.println(endtime - starttime);
```

Expected output

```
Davids-MacBook-Pro:aa-a1 davidvo$
[Davids-MacBook-Pro:aa-a1 davidvo$ java -cp .:jopt-simple-5.0.2.jar:sample.jar Gr]
aphEval adjlist
Interative mode.
1882
```