1. Algorithm for the method buildDataStructure()

```
hashTable = new HashTable(ceil(points.size() * 1.5))
for each p in S:
    key = floor(p)
    tuple = new Tuple(key, p)
    hashTable.add(tuple)
```

2. Algorithm for the method npHashNearestPoints(float p)

```
npHashNearestPoints(p):
    results = new ArrayList()
    k = floor(p)
    for each key in {k-1, k, k+1}:
        tuples = hashTable.search(key)
        for each tuple in tuples:
            point = tuple.getValue()
            if (point - p) <= 1:
                results.add(point)

    return results
```

3. Create an instance of the class NearestPoints using the data from the file points.txt. Run the methods allNearestPointsNaive() and allNearestPointsHash() (note that you must build the data structure) before running allNearestPointsHash(). Report the run times of both methods. Use System.currentTimeMillis() to measure run times.

Code to measure run times:

```
        NearestPoints nearestPoints = new NearestPoints("points.txt");
        nearestPoints.buildDataStructure();
        long start1 = System.currentTimeMillis();
        nearestPoints.allNearestPointsNaive();
        long end1 = System.currentTimeMillis();
        System.out.println("allNearestPointsNaive(): " + (end1 -
start1) / 1000.0 + " s");
        long start2 = System.currentTimeMillis();
        nearestPoints.allNearestPointsHash();
        long end2 = System.currentTimeMillis();
        System.out.println("allNearestPointsHash(): " + (end2 -
start2) / 1000.0 + " s");
```

Results:

```
        allNearestPointsNaive(): 31.693 s
        allNearestPointsHash(): 0.124 s
```