

```

1  MinTotErr[point(x,y), applicablePtSet[p1 ... pn], TotErrSum]
2  PossibleKStepPts[KstepFunctionPts[Points(x,y)], TotalErrorSum]
3
4  main(set of points P, k)
5  {
6      findStepFunctionError(P) and fill MinTotErr
7      return findMinimizedErr(k,P) to get the final result
8  }
9
10 findStepFunctionError(point set P){
11     if(P.size is zero) MinTotErr.add(p, point set[P], 0)
12     for(point p : point set P)
13     {
14         findStepFunctionError((P - p) - P.tail)
15         totalerror = previous element in MinTotErr arraylist + error[P.tail]
16         MinTotErr.add(p, point set[P], totalerror)
17     }
18 }
19
20 findMinimizedErr(int k,point set P, point set KStepPoints)
21 {
22     temp pointset = P;
23     //see if all points are covered in the union of set of points applicable of all k entries,
24     //by length or iterating over all pts (this current index's applicable point set + the recursive results point set)
25     if(k == 0 && P.isEmpty == true) // all points covered, k steps used
26     {
27         retrieve total errors from MinTotErr for each K point and sum
28         for(point p : KStepPoints)
29         {
30             TotalErrorSum += MinTotErr[p].TotErrSum
31         }
32         PossibleKStepPts.add[KStepPoints, TotalErrorSum];
33     }
34     else if (k == 0 && P.isEmpty == false) // k steps used, not all points covered
35         return
36     else if( k != 0 && P.isEmpty == true) // k steps not fully used, but all points covered
37         return
38     else // recurse
39     {
40         //subtract applicable point set from given point set,
41         tempptset = tempptset - MinTotErr[KStepPoints.last].applicablePtSet
42         for(point p : tempptset)
43         {
44             findMinimizedErr(k-1, tempptset - p, KStepPoints + p)
45         }
46     }
47     return Min(PossibleKStepPts.TotalErrorSum)
48 }

```