

Carnegie Mellon University
15-826 – Multimedia Databases and Data Mining
Fall 2014, C. Faloutsos

Homework 2

Due Date: Oct 30th, 3:00pm, in class

By: Chenying Hou (Q1, Q2) & Peixin Zheng(Q3, Q4, Q5).

IMPORTANT REMINDERS

- All homeworks are to be done **INDIVIDUALLY**.
- All written answers should be **TYPED**.
- **Code submission:** create an (`[andrew-id].tar.gz`) file, and submit it to blackboard. Please use the template at [<http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW2/template.tar.gz>] Notice that it has a `makefile` for all questions, and directories `/q1`, `/q2`, `/q3`, etc, each with an individual `makefile`, and place-holder code in Java.
 - you may use any language that runs on linux/`andrew`, as long as your `make` creates all the necessary output
 - Replace our place-holders with your code, `tar/ gzip` *only the absolutely necessary* files, into (`[andrew-id].tar.gz`), and submit to blackboard.

Other reminders, FYI

- Weight: 30% of total homeworks weight = 3% of course weight.
- Expected effort for this homework (order-of-magnitude):
 - Q1: $\approx$ 3-4 hours
 - Q2: $\approx$ 3-4 hours
 - Q3: $\approx$ 3-4 hours
 - Q4: $\approx$ 1 hours
 - Q5: $\approx$ 4-5 hours

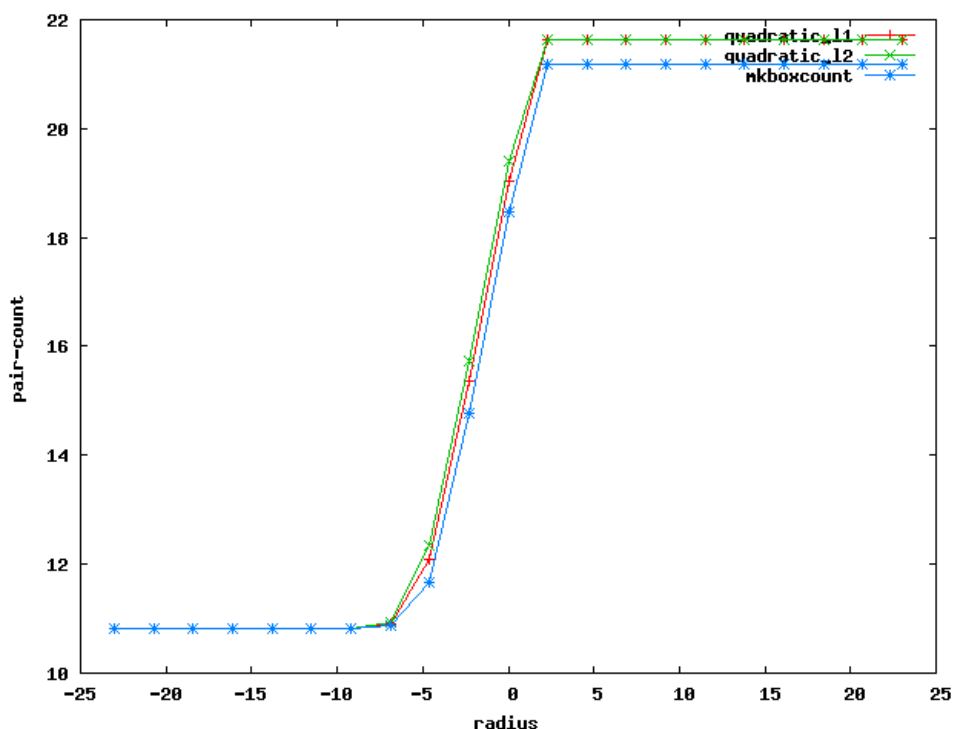
Q1 – Correlation Integral [25 pts]

Print answers on separate page, with '[course-id] [hw#] [question#] [andrew-id] [your name]'

Problem Description: We will implement an algorithm for calculating the correlation integral of a cloud of points and use it on a few datasets. We will compare your code against the approximate, but faster, box-counting method, included in the template tar-file, under `./q1/mkboxcount`.

1. [10 pt] Implement the naive $O(N^2)$ algorithm for computing the correlation integral. Your algorithm should include self-pairs and mirror pairs. Please submit your code.
Grading info: -2pt if they use wrong $l1$ or $l2$ formula
2. [5 pt] Plot and submit the correlation integral for mystery dataset *mystery1* using (a) your method and the $L1$ distance, (b) your method and the $L2$ distance and (c) the `mkboxcount` method.¹

Solution:

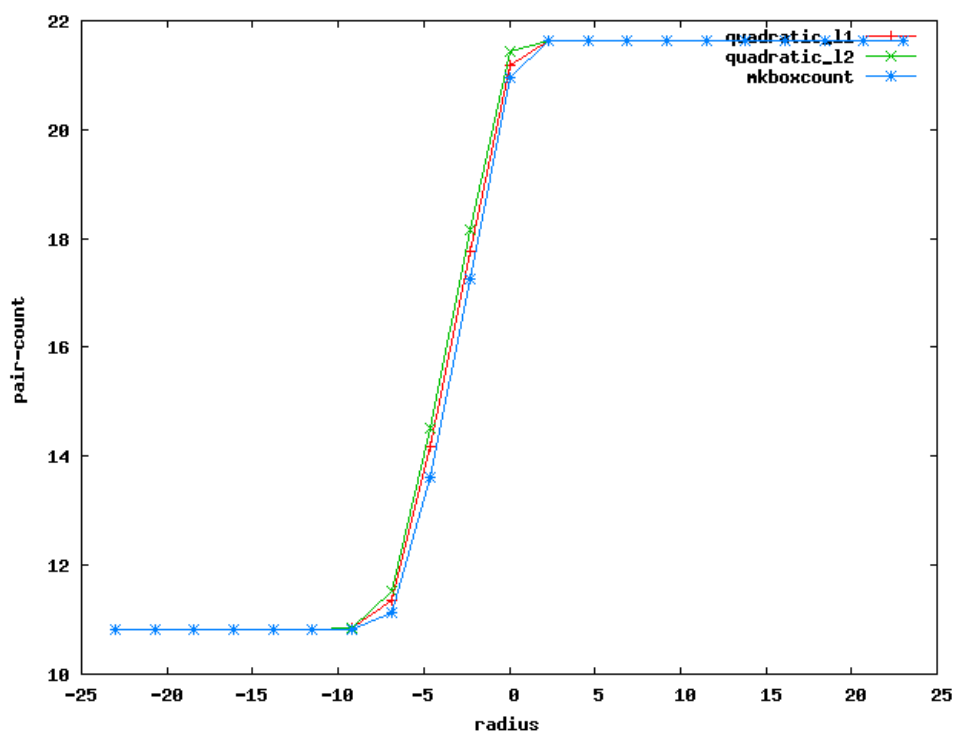


Grading info: -2pt if they miss one plot

¹FYI, $L2$ is the Euclidean Distance. Eg., for points $(x1, y1)$, and $(x2, y2)$, the $L2$ norm is $[(x1 - x2)^2 + (y1 - y2)^2]^{1/2}$. Similarly, $L1$ is given by $|x1 - x2| + |y1 - y2|$. And, in general, the L_p norm is given by $[|x1 - x2|^p + |y1 - y2|^p]^{1/p}$.

3. [5 pt] Plot and submit the correlation integral for mystery dataset *mystery2* using (a) your method and the $L1$ distance, (b) your method and the $L2$ distance and (c) the `mkboxcount` method.

Solution:



Grading info: -2pt if they miss one plot

4. [5 pt] Report the *wall-clock* time of your program on the *mystery2*, as well as the clock-time of running `mkboxcount`. Which one is longer? (Your program may run several minutes.)

Solution:

- slow quadratic(11): 538521ms
- slow quadratic(12): 1827264ms
- `mkboxcount`: 9114ms

Grading info: -2pt if they miss mkboxcount run time or your program run time

Output Format: For example, for the *mystery1* dataset, your program should generate the correlation integral in a file called `mystery1.result`. It should have lines like:

```
r1    #(pairs)
r2    #(pairs)
r3    #(pairs)
...
```

That is, two numbers per line, (separated by white space): the radius r and the count of pairs within r . The radius r should range from 10^{-10} to 10^{10} , by multiples of 10, that is:

10^{-10} , 10^{-9} , 10^{-8} , ... Thus, there should be 21 lines in your `.result` file.

What to turn in:

- **Code:** As mentioned in the beginning, please put your source files in the `/q1` directory in the template [<http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW2/template.tar.gz>]. The datasets are under `/data` directory. Please also modify the `makefile`, so that
 - `make` would generate all `.result` files and plots for the *mystery1* and *mystery2* datasets into the `/output` directory.
- **Answers:** On hard copy, please submit
 1. the code for Q1.1
 2. the plots for Q1.2 and Q1.3.
 3. the answer for Q1.4

SourceCode:

```
1 import java.io.BufferedReader;
2 import java.io.File;
3 import java.io.FileNotFoundException;
4 import java.io.FileReader;
5 import java.io.IOException;
6 import java.util.ArrayList;
7
8 /**
9  * @author Chenying Hou
10 */
11
12 public class ComputeCorrelationIntegral {
13
14     /**
15      * L2 distance
16      */
17     public static double l2Distance(double[] point1, double[] point2) {
18         double distance = 0;
19         for (int i = 0; i < point1.length; i++) {
20             distance += (point1[i] - point2[i]) * (point1[i] - point2[i]);
21         }
22         return Math.sqrt(distance);
23     }
24
25     /**
26      * L1 distance
27      */
28     public static double l1Distance(double[] point1, double[] point2) {
29         double distance = 0;
30         for (int i = 0; i < point1.length; i++) {
```

```

31     distance += Math.abs(point1[i] - point2[i]);
32 }
33 return distance;
34 }
35
36 public static void main(String[] args) {
37     long startTime = System.currentTimeMillis();
38     double rmin = Double.parseDouble(args[0]);
39     double rmax = Double.parseDouble(args[1]);
40     String l = args[3]; //l1 or l2 distance
41     try {
42         File file = new File(args[2]);
43         BufferedReader buffer=new BufferedReader(new FileReader(file));
44
45         try {
46             ArrayList<double[]> data = new ArrayList<double[]>();
47             String line;
48             //load data
49             while((line = buffer.readLine())!= null) {
50                 String[] parts = line.split("_");
51                 double[] values = new double[line.split("_").length];
52                 for (int i = 0; i < parts.length; i++) {
53                     values[i] = Double.parseDouble(parts[i]);
54                 }
55                 data.add(values);
56             }
57             //count pairs
58             for (double r = rmin; r <= rmax; r *= 10 ) {
59                 long count = 0;
60                 for (int i = 0; i < data.size(); i++) {
61                     for (int j = 0; j < data.size(); j++) {
62                         if (l.equals("l1")) {
63                             if (l1Distance(data.get(i), data.get(j)) <= r) {
64                                 count++;
65                             }
66                         } else {
67                             if (l2Distance(data.get(i), data.get(j)) <= r) {
68                                 count++;
69                             }
70                         }
71                     }
72                 }
73                 System.out.println(r + "_" + count);
74             }
75             //report the time elapse

```

```
76         long endTime = System.currentTimeMillis();
77         // System.out.println("Time elapse: " + (endTime - startTime) + "ms");
78     } catch (IOException e) {
79         // TODO Auto-generated catch block
80         e.printStackTrace();
81     }
82 } catch (FileNotFoundException e) {
83     // TODO Auto-generated catch block
84     e.printStackTrace();
85 }
86 }
87 }
```

Q2 – Chinese Restaurant Process [20 pts]

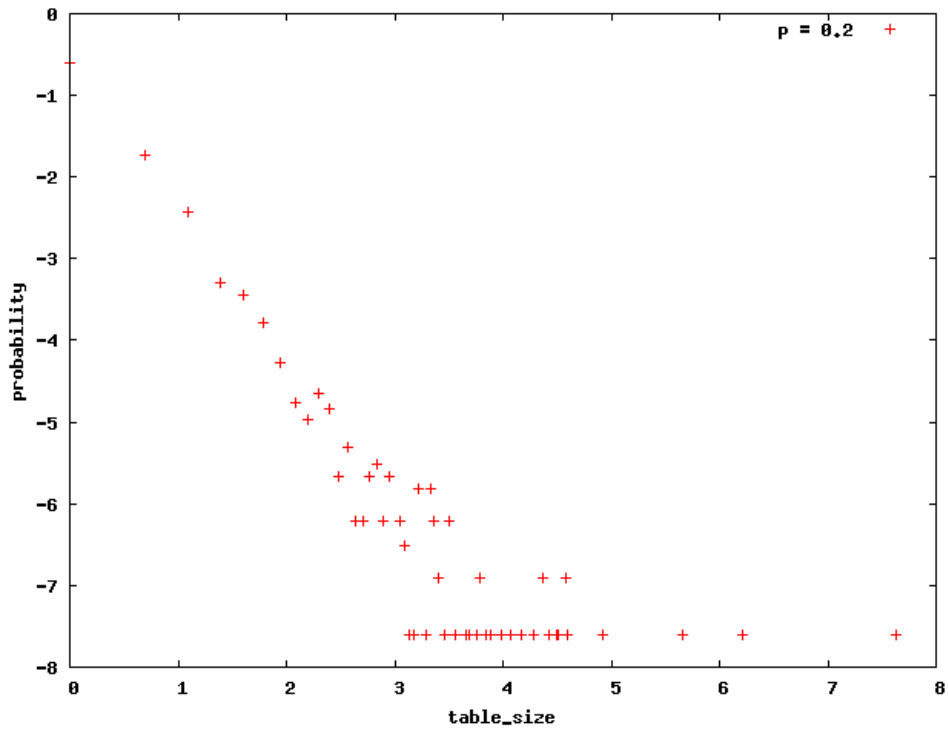
Print answers on separate page, with ‘[course-id] [hw#] [question#] [andrew-id] [your name]’

Problem Description: As was mentioned in the lectures, the “Chinese restaurant process” is a case of rich-get-richer setting, which results in power-law distributions. The goal is to get familiar with this process, and verify its properties.

Implementation Details: Imagine a (Chinese) restaurant with an infinite number of tables. Customers come into the restaurant one by one (one customer at each timestamp) and no one leaves. For each customer, he/she faces two choices: with probability p , this person will open a new table; otherwise, he/she will join a table with k customers, with probability proportional to k . Assume that initially, there is only one table with one customer. The probability distribution of the size of tables exhibits a power law tail [Newman05]. Please write code to verify this result.

1. **[10 pt]** Implement this process. Your program should take in two parameters: t and p where t is the total time steps and p is the probability for one customer to open a new table. Please submit your code. (Any output format is fine, as long as you can generate the plots that are required, below). Grading info: -2 if they calculate the probability incorrectly
2. **[5 pt]** Plot the PDF for the size of tables in the restaurant in log-log scale with $p = 0.2$ and $t = 10000$.

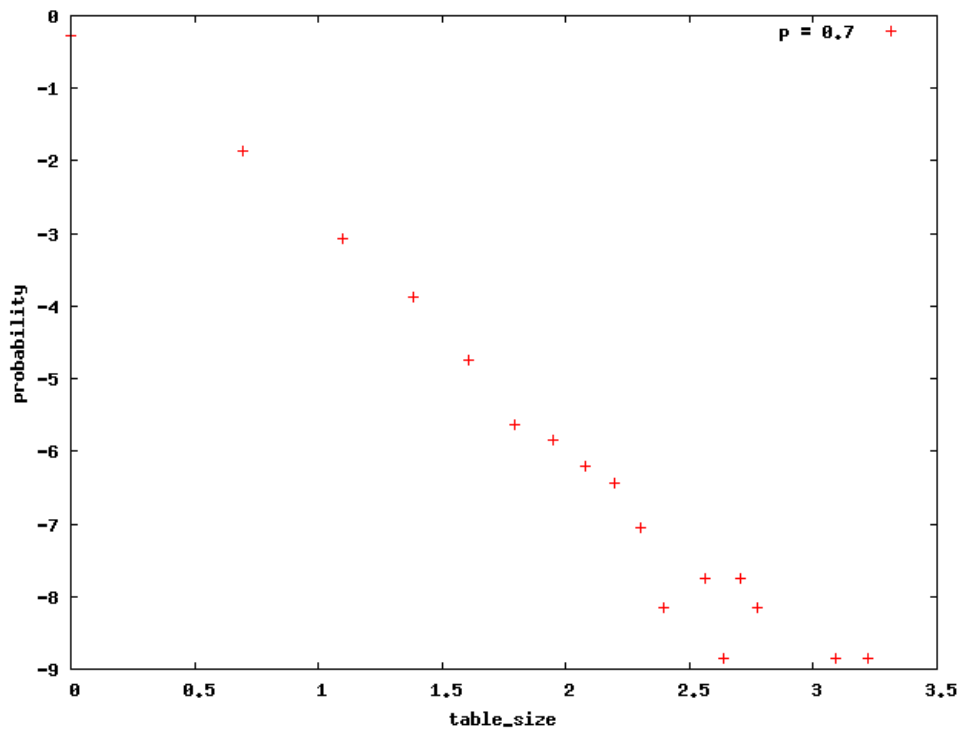
Solution:



Grading info: -5pt if the plot is wrong

3. **[3 pt]** Plot the PDF for the size of tables in the restaurant in log-log scale with $p = 0.7$ and $t = 10000$.

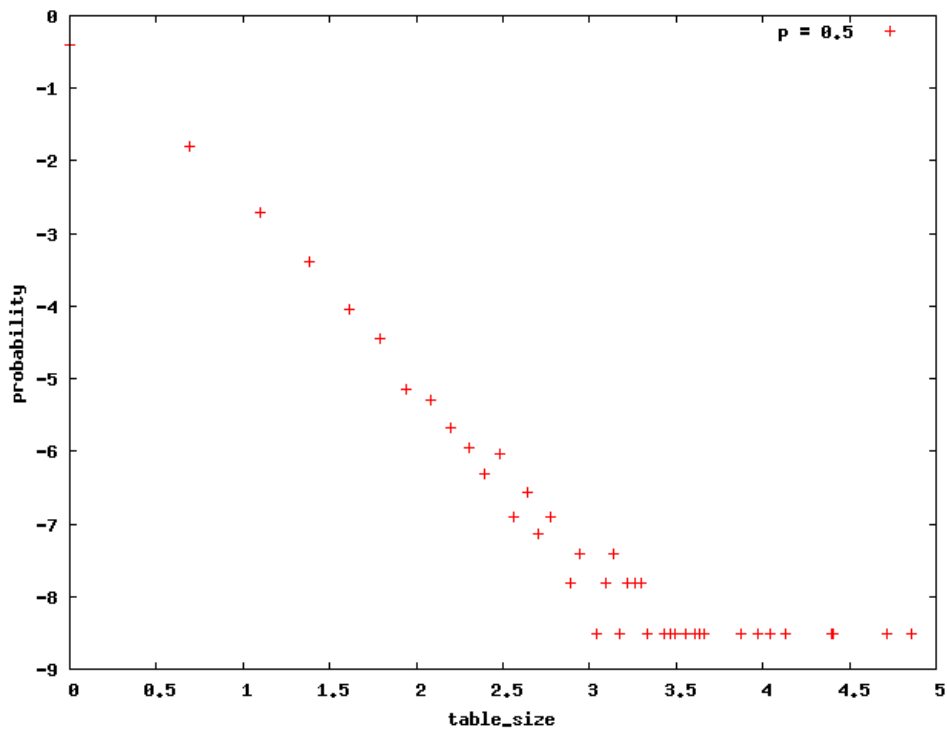
Solution:



Grading info: -3pt if the plot is wrong

4. [2 pt] Plot the PDF for the size of tables in the restaurant in log-log scale with $p = 0.5$ and $t = 10000$.

Solution:



Grading info: -2pt if the plot is wrong

What to turn in:

- **Code:** As mentioned earlier, please put your source code under the /q2 directory in the template [<http://www.cs.cmu.edu/~christos/courses/826.F14/HOMEWORKS/HW2/template.tar.gz>]. Please also modify the makefile, so that
 - make would generate all plots for Q2.2, Q2.3, Q2.4 under /output directory.
- **Answers:** On hard copy, please submit
 1. the code for Q2.1
 2. the plots for Q2.2, Q2.3, Q2.4

SourceCode:

```
1 import java.util.HashMap;
2 import java.util.List;
3 import java.util.ArrayList;
4 import java.util.Map;
5 import java.util.Random;
6
7 /**
8  * @author Chenying Hou
9  */
10 public class CRP {
```

```

11  /*
12   * find a table for user to join according to the number of customers on each
      table
13   */
14  public static int findATable(List<Integer> tables, int totalsize) {
15      Random rn = new Random();
16      int sharedCustomerIndex = rn.nextInt(totalsize) + 1; //based on 1
17      int i = 1; //based on 1
18      int accumulate = 0;
19      while (sharedCustomerIndex > accumulate) {
20          accumulate += tables.get(i - 1);
21          i++;
22      }
23      return i - 2;
24  }
25  /*
26   * print out the table size distribution
27   */
28  public static void printOutResult(List<Integer> tables, int totalCustomer) {
29      Map<Integer, Integer> pl = new HashMap<Integer, Integer>();
30      for (Integer i: tables) {
31          if (pl.containsKey(i)) {
32              pl.put(i, pl.get(i) + 1);
33          } else {
34              pl.put(i, 1);
35          }
36      }
37      /*A really naive way to get the final sorted result*/
38      for (int i = 0; i <= totalCustomer; i++) {
39          if (pl.containsKey(i)) {
40              System.out.println(i + "┘" + (double)((double)pl.get(i)/(double)(tables.
                  size())));
41          }
42      }
43  }
44
45  public static void main (String[] args) {
46      int t = Integer.parseInt(args[0]);
47      double p = Double.parseDouble(args[1]);
48
49      List<Integer> tables = new ArrayList<>();
50
51      //Initially, there is one customer at one table
52      tables.add(1);
53

```

```

54     for (int i = 1; i <= t; i++) {
55         Random rn = new Random();
56         double d = rn.nextDouble();
57         if (d < p) { //create a new table
58             tables.add(1);
59         } else { //join existing tables
60             int assignedTableIndex = findATable(tables, i);
61             tables.set(assignedTableIndex, tables.get(assignedTableIndex) + 1);
62         }
63     }
64     printOutResult(tables, t);
65 }
66 }

```

Q3 – String Editing Distance [20 pts]

Print answers on separate page, with '[course-id] [hw#] [question#] [andrew-id] [your name]'

Problem Description: The goal is to become familiar with the computation of string editing distance, and its dynamic-programming implementation.

Implementation Details: Write code (a) to print out the string editing matrix, (b) to compute the string editing distance, and (c) to print out the string editing path, for the following string pairs. Compute all the pairs under these two set of costs (where C_I , C_S and C_D stand for insertion, substitution, and deletion cost).

- Setting (A) (Deletion-heavy): $C_I = 1$, $C_S = 2$, $C_D = 3$.
- Setting (B) (Insertion-heavy): $C_I = 5$, $C_S = 2$, $C_D = 1$.

Here are the pairs of strings to run your code on:

- (a) *database* *dutyfree*
- (b) *information* *infomatic*
- (c) *iowaisinmidwest* *iowaisinwideeast*

Note:

- For the distance $D("d", "di")$ under setting (A) above, we have $D("d", "di") = C_I = 1$ while $D("di", "d") = C_D = 3$.
- In case there are more than one optimal path, take either one.

Output Format: For the pairs (a), (b), and (c), your program should print

- the string editing distance,
- the string editing matrix,
- and the optimal string editing path.

To check that you have the correct output format, run your program on:

- *survey* *surgery*
- setting all costs $C_I = C_D = C_S = 1$
- and **diff** your output with `q3/hw2q3-sample-output.txt` in the template tar-file that we provide.

What to turn in:

- **Code:** [10 pts] As we said, please replace our place-holder code in `./q3` of the template tar-file.
- **Answers:** [10 pts] On hard copy, please submit
 - your code, and,
 - your output for the three word-pairs (a), (b), and (c), and for both Settings (A) and (B).

SourceCode:

```
1 import java.io.BufferedReader;
2 import java.io.FileReader;
3 import java.io.IOException;
4
5
6 public class StringEditingDistance {
7     static int[][] calcMatrix(String str1, String str2, int insCost, int subCost,
8         int delCost) {
9         int len1 = str1.length();
10        int len2 = str2.length();
11        int[][] matrix = new int[len1 + 1][len2 + 1];
12        matrix[0][0] = 0;
13        for (int i = 1; i <= len1; i++) {
14            matrix[i][0] = i * delCost;
15        }
16        for (int j = 1; j <= len2; j++) {
17            matrix[0][j] = j * insCost;
18        }
19        for (int i = 1; i <= len1; i++) {
20            for (int j = 1; j <= len2; j++) {
21                int insResult = matrix[i][j - 1] + insCost;
22                int delResult = matrix[i - 1][j] + delCost;
23                int subResult = matrix[i - 1][j - 1] + ((str1.charAt(i - 1) == str2.charAt
24                    (j - 1)) ? 0 : subCost);
25                matrix[i][j] = Math.min(subResult, Math.min(insResult, delResult));
26            }
27        }
28        return matrix;
29    }
30    static void printMatrix(int[][] matrix) {
31        System.out.println("Matrix:");
32        for (int i = 0; i < matrix.length; i++) {
33            StringBuilder sb = new StringBuilder();
34            for (int j = 0; j < matrix[0].length - 1; j++) {
35                sb.append(matrix[i][j]);
36                sb.append("\t");
37            }
38            sb.append(matrix[i][matrix[0].length - 1]);
39            System.out.println(sb.toString());
40        }
41    }
42    static void printPath(int[][] matrix, String str1, String str2, int insCost,
43        int subCost, int delCost) {
44        StringBuilder sb = new StringBuilder();
```

```

42     int pos1 = str1.length();
43     int pos2 = str2.length();
44     System.out.println("String_editing_path:");
45     while (pos1 > 0 && pos2 > 0) {
46         if (matrix[pos1][pos2] == matrix[pos1 - 1][pos2 - 1] && str1.charAt(pos1 -
47             1) == str2.charAt(pos2 - 1)) {
48             sb.insert(0, "_,"");
49             pos1--;
50             pos2--;
51         } else if (matrix[pos1][pos2] - matrix[pos1 - 1][pos2 - 1] == subCost){
52             sb.insert(0, "S(" + str1.charAt(pos1 - 1) + "," + str2.charAt(pos2 - 1) +
53                 "),");
54             pos1--;
55             pos2--;
56         } else if (matrix[pos1][pos2] - matrix[pos1][pos2 - 1] == insCost) {
57             sb.insert(0, "I(" + str2.charAt(pos2 - 1) + "),");
58             pos2--;
59         } else if (matrix[pos1][pos2] - matrix[pos1 - 1][pos2] == delCost) {
60             sb.insert(0, "D(" + str1.charAt(pos1 - 1) + "),");
61             pos1--;
62         } else {
63             System.out.println("The_matrix_is_calculated_wrong._Exiting...");
64             return;
65         }
66     }
67     if (pos1 > 0) {
68         for (; pos1 > 0; pos1--) {
69             sb.insert(0, "D(" + str1.charAt(pos1 - 1) + "),");
70         }
71     } else if (pos2 > 0) {
72         for (; pos2 > 0; pos2--) {
73             sb.insert(0, "I(" + str2.charAt(pos2 - 1) + "),");
74         }
75     }
76     System.out.println(sb.substring(0, sb.length() - 2));
77 }
78 public static void main(String[] args) throws IOException {
79     int insCost = Integer.parseInt(args[0]);
80     int subCost = Integer.parseInt(args[1]);
81     int delCost = Integer.parseInt(args[2]);
82     System.out.println("Insertion_Cost:" + insCost);
83     System.out.println("Substitution_Cost:" + subCost);
84     System.out.println("Deletion_Cost:" + delCost);
85     BufferedReader br = new BufferedReader(new FileReader("input.txt"));
86     String inputString = null;

```

```

85     while ((inputString = br.readLine()) != null) {
86         String[] strArray = inputString.split("_");
87         System.out.println("String1:_ " + strArray[0] + ",_String2:_ " + strArray[1]);
88         int[][] matrix = calcMatrix(strArray[0], strArray[1], insCost, subCost,
            delCost);
89         printMatrix(matrix);
90         System.out.println("String_Editing_Distance:_ " + matrix[matrix.length - 1][
            matrix[0].length - 1]);
91         printPath(matrix, strArray[0], strArray[1], insCost, subCost, delCost);
92     }
93     br.close();
94 }
95
96 }

```

Result:

Insertion Cost: 1

Substitution Cost: 2

Deletion Cost: 3

String1: database, String2: dutyfree

Matrix:

0	1	2	3	4	5	6	7	8
3	0	1	2	3	4	5	6	7
6	3	2	3	4	5	6	7	8
9	6	5	2	3	4	5	6	7
12	9	8	5	4	5	6	7	8
15	12	11	8	7	6	7	8	9
18	15	14	11	10	9	8	9	10
21	18	17	14	13	12	11	10	11
24	21	20	17	16	15	14	11	10

String Editing Distance: 10

String editing path:

-, S(a, u), -, S(a, y), S(b, f), S(a, r), S(s, e), -

String1: information, String2: infomatic

Matrix:

0	1	2	3	4	5	6	7	8	9
3	0	1	2	3	4	5	6	7	8
6	3	0	1	2	3	4	5	6	7
9	6	3	0	1	2	3	4	5	6
12	9	6	3	0	1	2	3	4	5
15	12	9	6	3	2	3	4	5	6
18	15	12	9	6	3	4	5	6	7
21	18	15	12	9	6	3	4	5	6
24	21	18	15	12	9	6	3	4	5
27	24	21	18	15	12	9	6	3	4
30	27	24	21	18	15	12	9	6	5
33	30	27	24	21	18	15	12	9	8

String Editing Distance: 8

String editing path:

$\rightarrow, \rightarrow, \rightarrow, \rightarrow, D(r), \rightarrow, \rightarrow, \rightarrow, \rightarrow, D(o), S(n, c)$

String1: iowaisinmidwest, String2: iowaisinwideeast

Matrix:

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	3	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
9	6	3	2	3	4	5	6	7	6	7	8	9	10	11	12	13
12	9	6	5	2	3	4	5	6	7	8	9	10	11	10	11	12
15	12	9	8	5	2	3	4	5	6	7	8	9	10	11	12	13
18	15	12	11	8	5	2	3	4	5	6	7	8	9	10	11	12
21	18	15	14	11	8	5	2	3	4	5	6	7	8	9	10	11
24	21	18	17	14	11	8	5	2	3	4	5	6	7	8	9	10
27	24	21	20	17	14	11	8	5	4	5	6	7	8	9	10	11
30	27	24	23	20	17	14	11	8	7	4	5	6	7	8	9	10
33	30	27	26	23	20	17	14	11	10	7	4	5	6	7	8	9
36	33	30	29	26	23	20	17	14	11	10	7	6	7	8	9	10
39	36	33	32	29	26	23	20	17	14	13	10	7	6	7	8	9
42	39	36	35	32	29	26	23	20	17	16	13	10	9	8	7	8
45	42	39	38	35	32	29	26	23	20	19	16	13	12	11	10	7

String Editing Distance: 7

String editing path:

$\rightarrow, \rightarrow, S(w, v), \rightarrow, \rightarrow, \rightarrow, \rightarrow, \rightarrow, S(m, w), \rightarrow, \rightarrow, S(w, e), \rightarrow, I(a), \rightarrow, -$

Insertion Cost: 5

Substitution Cost: 2

Deletion Cost: 1

String1: database, String2: dutyfree

Matrix:

0	5	10	15	20	25	30	35	40
1	0	5	10	15	20	25	30	35
2	1	2	7	12	17	22	27	32
3	2	3	2	7	12	17	22	27
4	3	4	3	4	9	14	19	24
5	4	5	4	5	6	11	16	21
6	5	6	5	6	7	8	13	18
7	6	7	6	7	8	9	10	15
8	7	8	7	8	9	10	9	10

String Editing Distance: 10

String editing path:

-, S(a, u), -, S(a, y), S(b, f), S(a, r), S(s, e), -

String1: information, String2: infomatic

Matrix:

0	5	10	15	20	25	30	35	40	45
1	0	5	10	15	20	25	30	35	40
2	1	0	5	10	15	20	25	30	35
3	2	1	0	5	10	15	20	25	30
4	3	2	1	0	5	10	15	20	25
5	4	3	2	1	2	7	12	17	22
6	5	4	3	2	1	4	9	14	19
7	6	5	4	3	2	1	6	11	16
8	7	6	5	4	3	2	1	6	11
9	8	7	6	5	4	3	2	1	6
10	9	8	7	6	5	4	3	2	3
11	10	9	8	7	6	5	4	3	4

String Editing Distance: 4

String editing path:

-, -, -, -, D(r), -, -, -, -, D(o), S(n, c)

String1: iowaisinmidwest, String2: iowaisinwideeast

Matrix:

0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75	80
1	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75
2	1	0	5	10	15	20	25	30	35	40	45	50	55	60	65	70
3	2	1	2	7	12	17	22	27	30	35	40	45	50	55	60	65
4	3	2	3	2	7	12	17	22	27	32	37	42	47	50	55	60
5	4	3	4	3	2	7	12	17	22	27	32	37	42	47	52	57
6	5	4	5	4	3	2	7	12	17	22	27	32	37	42	47	52
7	6	5	6	5	4	3	2	7	12	17	22	27	32	37	42	47
8	7	6	7	6	5	4	3	2	7	12	17	22	27	32	37	42
9	8	7	8	7	6	5	4	3	4	9	14	19	24	29	34	39
10	9	8	9	8	7	6	5	4	5	4	9	14	19	24	29	34
11	10	9	10	9	8	7	6	5	6	5	4	9	14	19	24	29
12	11	10	11	10	9	8	7	6	5	6	5	6	11	16	21	26
13	12	11	12	11	10	9	8	7	6	7	6	5	6	11	16	21
14	13	12	13	12	11	10	9	8	7	8	7	6	7	8	11	16
15	14	13	14	13	12	11	10	9	8	9	8	7	8	9	10	11

String Editing Distance: 11

String editing path:

→, →, S(w, v), →, →, →, →, →, S(m, w), →, →, S(w, e), →, I(a), →, -

Q4 – Signature Files [10 pts]

Print answers on separate page, with '[course-id] [hw#] [question#] [andrew-id] [your name]'

Problem Description: The goal is to become familiar with the generation and usage of signatures. Consider the following words and their signatures:

Word	Signature
database	1000 0001 0100 0100 0010
multimedia	0100 1000 0010 1001 0000
science	0000 0000 1010 1010 0100
computer	0100 1001 0000 0010 0001
signature	0000 1000 1100 0010 0100

1. [6 pt] Suppose document D only has 3 words: 'database', 'multimedia', and 'science'.

(a) What is the signature S of Document D ?

Solution: 1100 1001 1110 1111 0110

(b) Using the signature S of the above document, answer the questions:

- i. Does 'computer' lead to a 'false drop' for D ?

Solution: No.

- ii. Does 'signature' lead to a 'false drop' for D ?

Solution: Yes.

2. [4 pt] Consider another document, D' , with signature S' :

$$S' = 1100\ 1001\ 0010\ 1011\ 0111$$

Among the 5 words above, how many qualify for the signature test, that is, how many will say *maybe*? which ones are they?

Solution: 2 will qualify. "multimedia" and "computer".

What to turn in:

- **Code:** : No code to turn in.
- **Answers:** : On hard copy, your answers to all the questions above.

Q5 – Forest Fire Problem [25 pts]

Print answers on separate page, with '[course-id] [hw#] [question#] [andrew-id] [your name]'

Problem Description: The goal is to use the 'Forest Fire' idea, to become familiar with the percolation, and its ability to generate power laws and phase transitions.

Implementation Details: Please refer to the foils: [http://www.cs.cmu.edu/~christos/courses/826.F14/F0ILS-pdf/190_powerLaws.pdf] starting from page 13.

The following is a repetition of the definitions:

- *Fire:* Suppose you have a square field ('forest') with some trees on it. Suppose that at $t = 0$, a lightning hits a tree and the tree starts to burn immediately. A tree burns for 1 time-tick, and then it lights up all the trees (if any), in the 4 neighboring positions - North, South, East, West.
- *Example:* Figure 1 shows a 4 * 4 field with some trees. (Empty cells are white). After the lightning strikes the tree on the top, it burns, burns its neighbors, recursively, and after $B=4$ time-ticks, all the blue-labeled trees are burnt.
- *Cluster:* This is a set of trees that will all burn, if any of them gets hit. E.g., in 1(a), the occupied cells form two clusters (the red, of size 2 cells, and the blue, of size 6 cells).

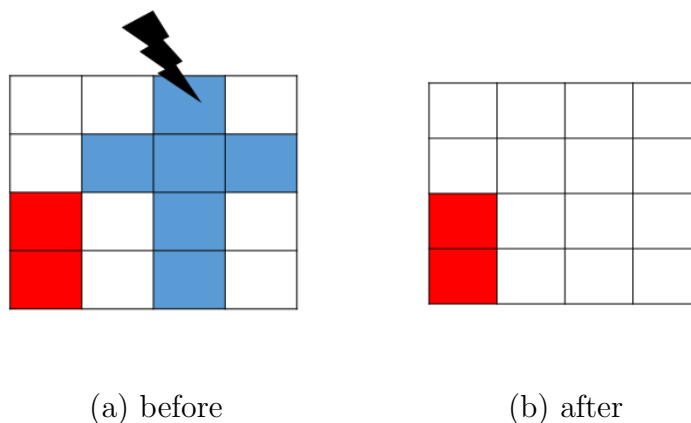


Figure 1: A field of trees (a) before the lightning strikes and (b) after the fire has stopped. The tree struck by the lightning is shown.

Implementation Details: Write code to simulate the burning of the forest on a 50 * 50 matrix. That is, create forests of varying densities p (p is the probability that a cell contains a tree). Then, choose a non-empty cell at random (where the lightning hits), and report the burning time B . For each density p , run the experiment $r = 5$ runs (i.e., 5 "lightnings").

As in the lecture foils, let p_c be the critical density, that is, the density that results in the longest burning times B . From your experiments above, estimate p_c

1. **[5 pt]** For each density p (p ranging from 0.10 to 0.90, by step of 0.05), print out the burning time $B(p)$ of all 5 runs, for each p .
2. **[2 pt]** For each density p , take the largest burning time $\hat{B}(p)$ among the 5 runs, plot $\hat{B}(p)$ against p , and estimate the critical density p_c .
3. **[3 pt]** For low, high, and critical density ($p = 0.10, 0.90$ and p_c) print out the 50×50 matrix ('forest'), before the lightning hit.
4. **[5 pt]** At critical density, analysis claims that there are clusters of all sizes, following a power law. Verify this claim: For low, high, and critical density p ($p = 0.10, 0.90$ and p_c), plot the histogram of cluster sizes (count c_s of clusters of size s , versus the size s). Please put the 3 graphs in the same plot, and make sure you plot is in log-log scales.

Note:

- The matrices ('forests') that you use in part 3 and 4 may be different that the ones in part 1 and part 2.

What to turn in:

- **Code:** **[10 pts]** As we said, please follow the provided `tar.gz` template, and put your code and results under the directory `./q5` inside the tar file you submit. Again, when we type `make`, we should see your answers.
- **Answers:** On hard copy, handin
 1. the code you use to generate the answers, and,
 2. the answers and plots to all questions Q5.1-Q5.4

SourceCode:

```

1
2 public class Point {
3     int x;
4     int y;
5     public Point(int _x, int _y) {
6         x = _x;
7         y = _y;
8     }
9 }

1 import java.text.DecimalFormat;
2 import java.util.ArrayList;
3 import java.util.Collections;
4 import java.util.HashMap;
5 import java.util.LinkedList;
6 import java.util.List;
7 import java.util.Queue;
8 import java.io.FileWriter;

```

```

9 import java.io.BufferedWriter;
10 import java.io.IOException;
11
12 public class ForestFire {
13     static void addTrees(boolean[][] matrix, double p, List<Point> trees) {
14         for (int i = 0; i < matrix.length; i++) {
15             for (int j = 0; j < matrix[0].length; j++) {
16                 if (Math.random() <= p) {
17                     matrix[i][j] = true;
18                     trees.add(new Point(i, j));
19                 }
20             }
21         }
22     }
23
24     static Point selectLightning(List<Point> trees) {
25         int index = (int) (Math.floor(Math.random() * trees.size()));
26         return trees.get(index);
27     }
28
29     static int burn(Point lightning, boolean[][] matrix) {
30         Queue<Point> queue = new LinkedList<Point>();
31         matrix[lightning.x][lightning.y] = false;
32         queue.add(lightning);
33         int time = 0;
34         ArrayList<Point> tempList = new ArrayList<Point>();
35         while (!queue.isEmpty()) {
36             tempList.clear();
37             while (!queue.isEmpty()) {
38                 Point point = queue.remove();
39                 tempList.add(point);
40             }
41             for (int i = 0; i < tempList.size(); i++) {
42                 Point point = tempList.get(i);
43                 int x = point.x;
44                 int y = point.y;
45                 if (x > 0 && matrix[x - 1][y]) {
46                     matrix[x - 1][y] = false;
47                     queue.add(new Point(x - 1, y));
48                 }
49                 if (y > 0 && matrix[x][y - 1]) {
50                     matrix[x][y - 1] = false;
51                     queue.add(new Point(x, y - 1));
52                 }
53                 if (x < matrix.length - 1 && matrix[x + 1][y]) {

```

```

54         matrix[x + 1][y] = false;
55         queue.add(new Point(x + 1, y));
56     }
57     if (y < matrix.length - 1 && matrix[x][y + 1]) {
58         matrix[x][y + 1] = false;
59         queue.add(new Point(x, y + 1));
60     }
61 }
62 time++;
63 }
64 return time;
65 }
66 static void printMatrix(boolean[][] matrix, double p) {
67     DecimalFormat df = new DecimalFormat("0.00");
68     System.out.println("Plot for p=" + df.format(p));
69     for (int i = 0; i < matrix.length; i++) {
70         StringBuilder sb = new StringBuilder();
71         for (int j = 0; j < matrix[0].length; j++) {
72             if (matrix[i][j]) {
73                 sb.append("*");
74             } else {
75                 sb.append("_");
76             }
77         }
78         System.out.println(sb.toString());
79     }
80 }
81 static void printClusterDistribution(boolean[][] matrix, double p, String
    filename) throws IOException {
82     HashMap<Integer, Integer> dist = new HashMap<Integer, Integer>();
83     for (int i = 0; i < matrix.length; i++) {
84         for (int j = 0; j < matrix[0].length; j++) {
85             if (matrix[i][j]) {
86                 int clusterSize = getClusterSize(matrix, new Point(i, j));
87                 if (!dist.containsKey(clusterSize)) {
88                     dist.put(clusterSize, 1);
89                 } else {
90                     int freq = dist.get(clusterSize);
91                     dist.remove(clusterSize);
92                     dist.put(clusterSize, freq + 1);
93                 }
94             }
95         }
96     }
97     DecimalFormat df = new DecimalFormat("0.00");

```

```

98     System.out.println("p=" + df.format(p));
99     ArrayList<Integer> keyList = new ArrayList<Integer>(dist.keySet());
100     Collections.sort(keyList);
101     BufferedWriter bw = new BufferedWriter(new FileWriter(filename));
102     for (int i = 0; i < keyList.size(); i++) {
103         System.out.println("cluster_size=" + keyList.get(i) + ",freq=" + dist.get(
            keyList.get(i)));
104         bw.write(keyList.get(i) + "," + dist.get(keyList.get(i)));
105         bw.newLine();
106     }
107     bw.close();
108 }
109 static int getClusterSize(boolean[][] matrix, Point start) {
110     Queue<Point> queue = new LinkedList<Point>();
111     matrix[start.x][start.y] = false;
112     queue.add(start);
113     int numPoints = 0;
114     while (!queue.isEmpty()) {
115         Point point = queue.remove();
116         numPoints++;
117         int x = point.x;
118         int y = point.y;
119         if (x > 0 && matrix[x - 1][y]) {
120             matrix[x - 1][y] = false;
121             queue.add(new Point(x - 1, y));
122         }
123         if (y > 0 && matrix[x][y - 1]) {
124             matrix[x][y - 1] = false;
125             queue.add(new Point(x, y - 1));
126         }
127         if (x < matrix.length - 1 && matrix[x + 1][y]) {
128             matrix[x + 1][y] = false;
129             queue.add(new Point(x + 1, y));
130         }
131         if (y < matrix.length - 1 && matrix[x][y + 1]) {
132             matrix[x][y + 1] = false;
133             queue.add(new Point(x, y + 1));
134         }
135     }
136     return numPoints;
137 }
138
139 public static void main(String[] args) throws IOException {
140     double epsilon = 0.00000001;
141     int maxTime = 0;

```

```

142     double maxP = 0.0;
143     for (double p = 0.1; p - 0.9 <= epsilon; p += 0.05) {
144         for (int i = 0; i < 5; i++) {
145             boolean[][] matrix = new boolean[50][50];
146             List<Point> trees = new ArrayList<Point>();
147             addTrees(matrix, p, trees);
148             Point lightning = selectLightning(trees);
149             int time = burn(lightning, matrix);
150             DecimalFormat df = new DecimalFormat("0.00");
151             System.out.println("Possibility:␣" + df.format(p)
152                 + ",␣num␣trees:␣" + trees.size() + ",␣time:␣" + time);
153             if (time > maxTime) {
154                 maxTime = time;
155                 maxP = p;
156             }
157         }
158     }
159     System.out.println("Peak␣possibility:␣" + maxP + ",␣peak␣time:␣" + maxTime);
160
161     boolean[][] matrixLow = new boolean[50][50];
162     boolean[][] matrixHigh = new boolean[50][50];
163     boolean[][] matrixPeak = new boolean[50][50];
164     List<Point> trees = new ArrayList<Point>();
165     addTrees(matrixLow, 0.1, trees);
166     printMatrix(matrixLow, 0.1);
167     printClusterDistribution(matrixLow, 0.1, "bottom.dat");
168     trees.clear();
169     addTrees(matrixHigh, 0.9, trees);
170     printMatrix(matrixHigh, 0.9);
171     printClusterDistribution(matrixHigh, 0.9, "top.dat");
172     trees.clear();
173     addTrees(matrixPeak, maxP, trees);
174     printMatrix(matrixPeak, maxP);
175     printClusterDistribution(matrixPeak, maxP, "peak.dat");
176     trees.clear();
177 }
178
179 }

1 set logscale xy
2 set style data linesp
3 set xrange [0.5:10000]
4 set yrange [0.5:10000]
5 set terminal png
6 set xlabel "size"
7 set ylabel "count"

```

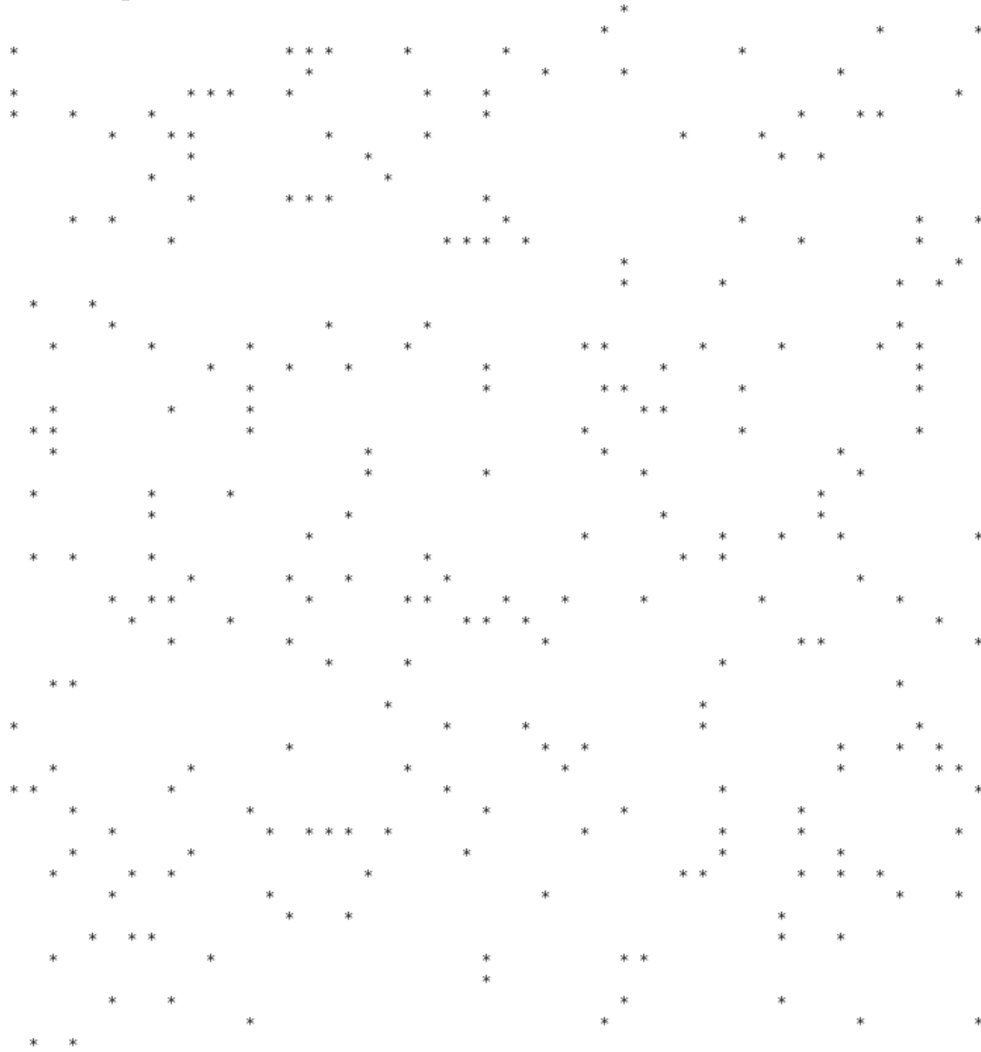
```
8 set output 'plot.png'
9 plot "bottom.dat", "peak.dat", "top.dat"
```

Result:

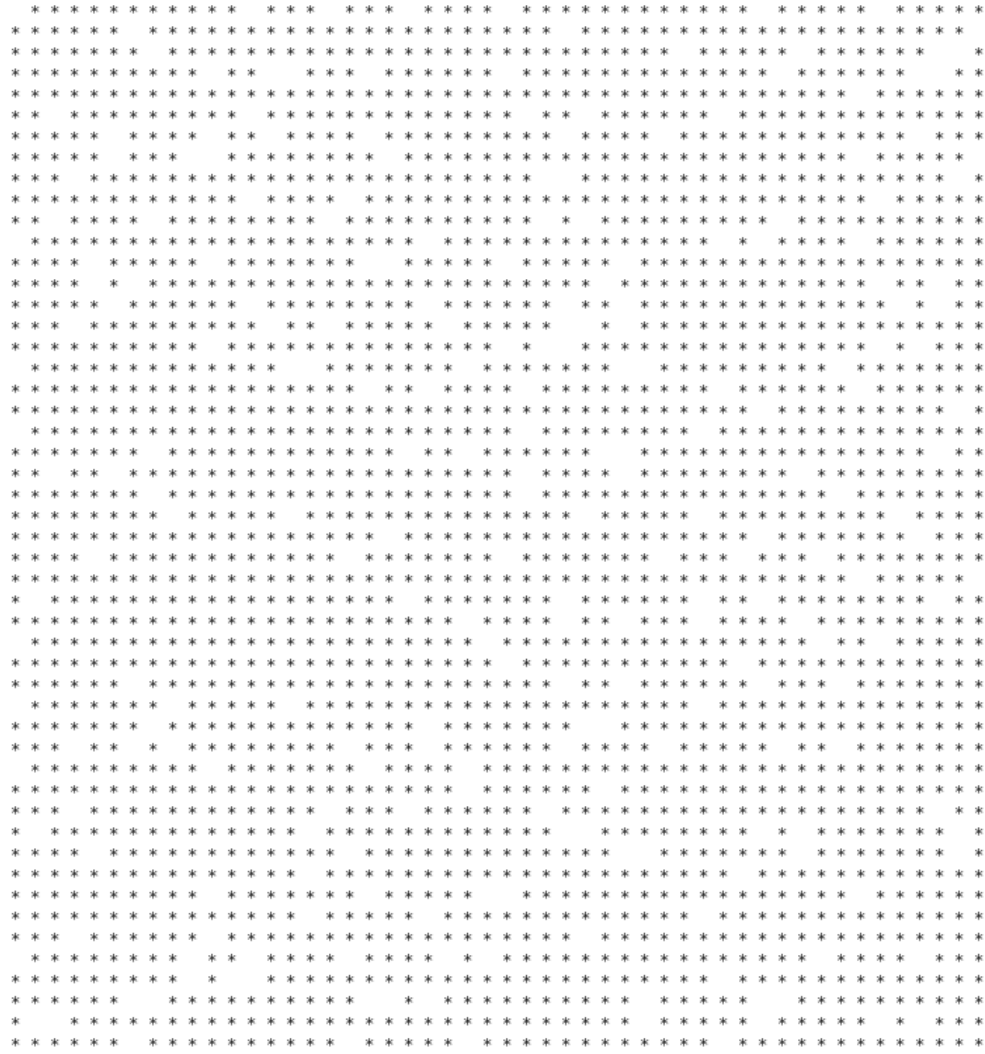
Possibility: 0.10, num trees: 269, time: 1
Possibility: 0.10, num trees: 251, time: 2
Possibility: 0.10, num trees: 250, time: 1
Possibility: 0.10, num trees: 278, time: 4
Possibility: 0.10, num trees: 250, time: 2
Possibility: 0.15, num trees: 383, time: 2
Possibility: 0.15, num trees: 363, time: 1
Possibility: 0.15, num trees: 407, time: 2
Possibility: 0.15, num trees: 363, time: 3
Possibility: 0.15, num trees: 372, time: 5
Possibility: 0.20, num trees: 511, time: 1
Possibility: 0.20, num trees: 482, time: 3
Possibility: 0.20, num trees: 491, time: 1
Possibility: 0.20, num trees: 521, time: 9
Possibility: 0.20, num trees: 531, time: 1
Possibility: 0.25, num trees: 648, time: 1
Possibility: 0.25, num trees: 587, time: 2
Possibility: 0.25, num trees: 638, time: 1
Possibility: 0.25, num trees: 636, time: 2
Possibility: 0.25, num trees: 623, time: 1
Possibility: 0.30, num trees: 750, time: 3
Possibility: 0.30, num trees: 750, time: 2
Possibility: 0.30, num trees: 725, time: 4
Possibility: 0.30, num trees: 766, time: 1
Possibility: 0.30, num trees: 766, time: 2
Possibility: 0.35, num trees: 873, time: 4
Possibility: 0.35, num trees: 852, time: 4
Possibility: 0.35, num trees: 890, time: 3
Possibility: 0.35, num trees: 873, time: 2
Possibility: 0.35, num trees: 876, time: 3
Possibility: 0.40, num trees: 987, time: 1
Possibility: 0.40, num trees: 1028, time: 3
Possibility: 0.40, num trees: 981, time: 4
Possibility: 0.40, num trees: 1025, time: 1
Possibility: 0.40, num trees: 1018, time: 13
Possibility: 0.45, num trees: 1157, time: 8
Possibility: 0.45, num trees: 1099, time: 20
Possibility: 0.45, num trees: 1100, time: 3
Possibility: 0.45, num trees: 1115, time: 8

Possibility: 0.45, num trees: 1168, time: 35
 Possibility: 0.50, num trees: 1237, time: 1
 Possibility: 0.50, num trees: 1271, time: 8
 Possibility: 0.50, num trees: 1211, time: 17
 Possibility: 0.50, num trees: 1243, time: 3
 Possibility: 0.50, num trees: 1249, time: 12
 Possibility: 0.55, num trees: 1415, time: 88
 Possibility: 0.55, num trees: 1338, time: 29
 Possibility: 0.55, num trees: 1372, time: 53
 Possibility: 0.55, num trees: 1365, time: 9
 Possibility: 0.55, num trees: 1374, time: 7
 Possibility: 0.60, num trees: 1520, time: 24
 Possibility: 0.60, num trees: 1550, time: 94
 Possibility: 0.60, num trees: 1530, time: 84
 Possibility: 0.60, num trees: 1465, time: 4
 Possibility: 0.60, num trees: 1474, time: 113
 Possibility: 0.65, num trees: 1647, time: 10
 Possibility: 0.65, num trees: 1610, time: 93
 Possibility: 0.65, num trees: 1661, time: 93
 Possibility: 0.65, num trees: 1573, time: 145
 Possibility: 0.65, num trees: 1628, time: 3
 Possibility: 0.70, num trees: 1717, time: 85
 Possibility: 0.70, num trees: 1800, time: 83
 Possibility: 0.70, num trees: 1772, time: 93
 Possibility: 0.70, num trees: 1737, time: 85
 Possibility: 0.70, num trees: 1758, time: 70
 Possibility: 0.75, num trees: 1892, time: 87
 Possibility: 0.75, num trees: 1876, time: 78
 Possibility: 0.75, num trees: 1868, time: 72
 Possibility: 0.75, num trees: 1842, time: 81
 Possibility: 0.75, num trees: 1894, time: 73
 Possibility: 0.80, num trees: 2020, time: 67
 Possibility: 0.80, num trees: 1995, time: 105
 Possibility: 0.80, num trees: 1983, time: 98
 Possibility: 0.80, num trees: 2006, time: 92
 Possibility: 0.80, num trees: 1974, time: 78
 Possibility: 0.85, num trees: 2112, time: 82
 Possibility: 0.85, num trees: 2105, time: 77
 Possibility: 0.85, num trees: 2142, time: 80
 Possibility: 0.85, num trees: 2112, time: 66
 Possibility: 0.85, num trees: 2104, time: 71
 Possibility: 0.90, num trees: 2249, time: 69

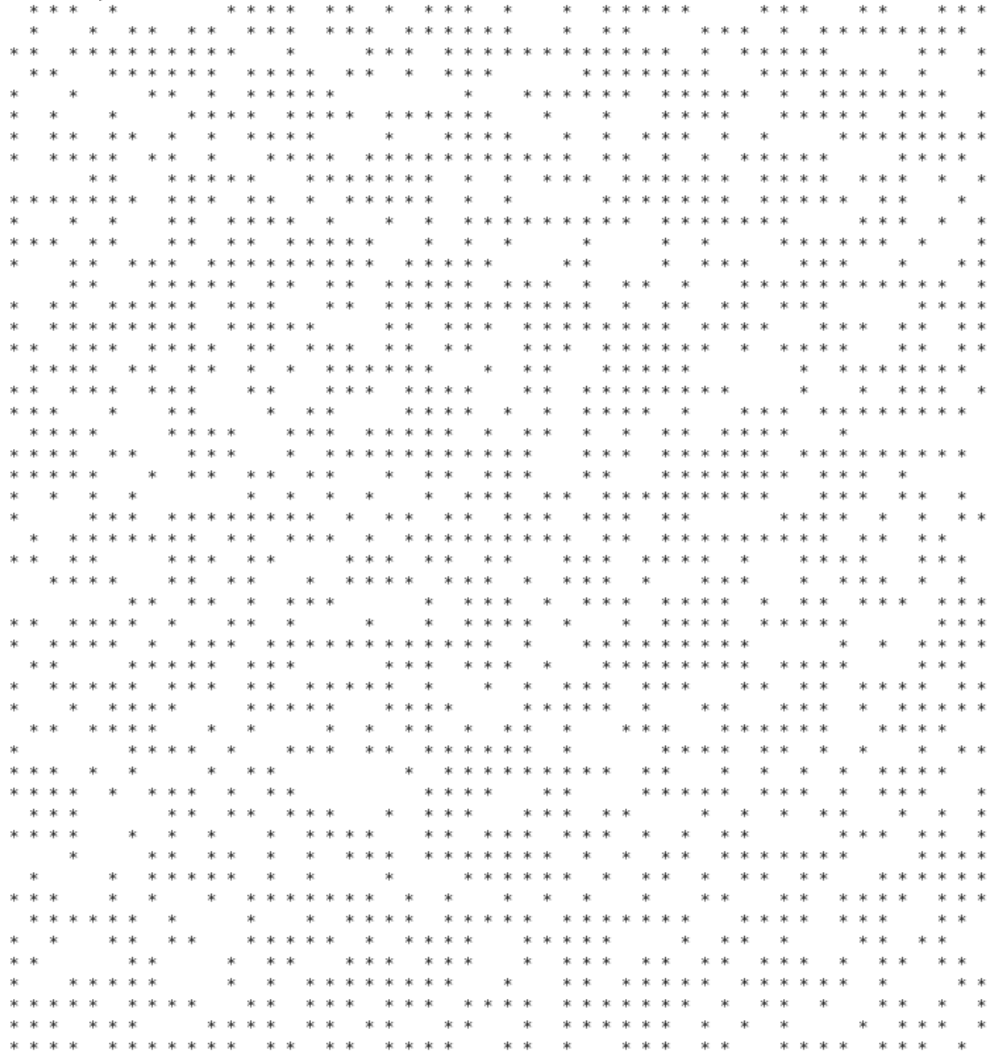
Possibility: 0.90, num trees: 2244, time: 70
 Possibility: 0.90, num trees: 2250, time: 79
 Possibility: 0.90, num trees: 2253, time: 82
 Possibility: 0.90, num trees: 2262, time: 81
 Peak possibility: 0.65, peak time: 145
 Plot for $p = 0.10$



Plot for $p = 0.90$



Plot for $p = 0.65$



Plot of cluster size:

