Computer Science 220S1C (2019)

Assignment 3 (hashing and basic graph algorithms)

Due date May 21, 2018, 10pm
100 Marks in total

This assignment requires you to submit programs in Python that you have written yourself to the automarker, `http://www.cs.auckland.ac.nz/automated-marker`. Your implementation must be from first principles and cannot use an existing library methods that might solve the problem (eg hashes, performs graph operations etc).

The automarker runs on a Linux box. Read the automarker help and FAQ for more details.

Please submit only Python source code named a3qX.py where X is 1, 2 or 3 corresponding to the problems below.

1. **Hashing** *30 marks*

   You are given an input file with multiple lines. On each line there is:

   (a) the size, m, of the hash table which is a prime

   (b) a second prime number, $p < m-1$, used for open addressing with double hashing

   (c) a sequence of integer keys

   Your program will create a hash table of size m and insert each key into the table. Use the primary hash function

   $$h(x) = x \bmod m$$

   and the secondary hash function for collision resolution

   $$\delta(x) = (x \bmod p) + 1.$$

   We use the convention that probes are made to the left of the initial probe.

   It will output

   (a) the total number of keys that are out of place in the table,

(b) the number of probes required to look up the final item inserted in the hash table, and

(c) the expected number of probes to find a random item, $T_{ss}(\lambda) = \frac{1}{\lambda} \ln\left(\frac{1}{1-\lambda}\right)$ rounded to 3 decimal places.

Input format: On each line is a sequence of at least 3 comma separated integers where the first integer is m, the second is p and the remaining integers are keys.

For example:

```
11,3,2,14,3
7,2,1,2
```

Output format: For each line of input, the output should have a line with the three values described in (a)-(c) above, in that order, separated by commas. The last number should have exactly 3 decimal places. Use Python str.format rounding behaviour.

For the example input above, output would be:

```
1,3,1.168
0,1,1.178
```

2. **Reverse of a digraph** *30 Marks*

For a given set of digraphs, write a program that prints out the reverse of each digraph.

Input format: described below under the heading "Digraph input format". Note that adjacency lists are sorted.

Output format: use the input format to output your result ensuring that the output adjacency lists are sorted.

For the example input shown below, the output would be

```
4
2
0
1
0 1
3

0 2
0
0
```

3. **Tree and back arcs in a DFS** *40 Marks*

   For a given set of digraphs, write a program that performs DFS on each digraph starting at node 0 and prints out the total number of tree arcs and back arcs resulting from the traversal. Use our standard convention that when there is a choice of white or grey nodes, the one with the lowest index should be chosen.

   Input format: described below under the heading, "Digraph input format".

   Output format: For each input digraph, print out a line with the number of tree arcs and the number of back arcs separated by a comma.

   For the example input shown below, the output would be

   ```
   3,1
   2,0
   ```

Digraph input format

A sequence of one or more digraphs is taken from the keyboard (System.in). Each graph is represented by an adjacency list. The first line is an integer n indicating the order of the graph. This is followed by n white space separated lists of adjacencies for nodes labeled 0 to n - 1. The input will be terminated by a line consisting of one zero (0). This line should not be processed. The sample input below shows two digraphs, the first has node set $\{0, 1, 2, 3\}$ and arc set $\{(0, 1), (0, 3), (1, 2), (1, 3), (2, 0)\}$, the second has node set $\{0, 1, 2\}$ and arc set $\{(0, 1), (0, 2), (2, 1)\}$.

```
4
1 3
2 3
0

3
1 2

1
0
```

Marking

The maximum number of submissions for each problem is fixed at 10.

Each problem has five test cases associated with it worth one fifth of the marks for that problem. Some of the test cases will be large to test for speed. You get full marks if you pass all test cases.