Software Engineering 284 (2021)

Assignment 3 (Hashing and traversals)

Due date October 5, 2021, 10pm

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

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

You can use any of the languages accepted by the automarker (Java, C, C++, Python 3, Go, Rust, F#, Javascript).

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, $p_1 < m - 1$, used for open addressing with double hashing

   (c) a sequence of integer keys

   Your program will create two hash tables of size m and insert each key once into each table. Both tables use the primary hash function

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

   . The first table (called the LP table) use linear probing for collision resolution. The second hash table (called the DH table) uses double hashing where the secondary hash function is

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

   Throughout, we use the convention that probes are made to the left of the initial probe.

For each input line, your program should calculate

(a) the total number of keys that have a collision upon insertion in the LP table,

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

(c) the total number of keys that have a collision upon insertion in the DH table, and

(d) the number of probes required to look up the final item inserted in the DH table.

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:

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

Output format: For each line of input, the output should have a line with the four values described in (a)-(d) above, in that order, separated by commas.

For the example input above, output would be:

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

2. **Reverse of a digraph** *20 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".

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. **Finding cycles in a digraph** *30 Marks*

   For a given set of digraphs, write a program that determines whether or not the digraph contains a cycle.

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

   Output format: For each input digraph, print out a line with a one (1) if the digraph contains a cycle and a zero (0) otherwise.

   For the example input shown below, the output would be

   ```
   1
   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 12.

Each problem has 3 test cases associated with it worth one third of the marks for that problem. Some of the test cases will be large. You get full marks if you pass all test cases.