Homework #4

(Due: May 4)

Task 1. [20 Points] Estimating the Number of Distinct Elements in a Sequence. Suppose we are scanning a long sequence $\mathcal{S}$ of integers, and we would like to count the number (say, m) of distinct integers appearing in $\mathcal{S}$. It turns out that if m is large, then it can be estimated reasonably well by scanning $\mathcal{S}$ only once, and using extra storage only for a constant number (say, K) of additional integers. For our current task each such additional integer will be an L-bit unsigned number, where $L = 32$. For $0 \leq k < K$, let $\mathcal{B}(k)$ be the k-th such integer, and for $0 \leq l < L$, let $\mathcal{B}_l(k)$ denote the l-th least significant bit of $\mathcal{B}(k)$. We will call $\mathcal{B}$ a *counting bitmap*.

We initialize each $\mathcal{B}_l(k)$ to 0, and then start scanning $\mathcal{S}$. For each integer $x \in \mathcal{S}$, we call the following update function.

```
UPDATE( B, x )
  1. h ← hash( x )
  2. k ← h mod K,  l ← ρ( h div K )
  3. if l < L then B_l(k) ← 1
```

The update function uses the following hash function[1]:

$$hash(\ x\ ) = (\ ax\ +\ b\ ) \bmod q,$$

where $a = 1073741827$, $b = 17179869209$, and $q = 4294967291$, and also the following function:

$$\rho(\ x\ ) = \begin{cases} L & \text{if } x = 0, \\ \min\limits_{i\ \in\ [0,\ L)} \{\ x_i = 1\ \} & \text{otherwise,} \end{cases}$$

where, x is an L-bit unsigned integer, and x_i is the i-th ($0 \leq i < L$) least significant bit of x.

After updating $\mathcal{B}$ for all integers in $\mathcal{S}$, we call EVALUATE($\mathcal{B}$) to get an estimate of m.

Observe that if $\mathcal{B}^u$ is a counting bitmap of a sequence $\mathcal{S}_u$, and $\mathcal{B}^v$ is that of another sequence $\mathcal{S}_v$, then the counting bitmap $\mathcal{B}$ of $\mathcal{S} = \mathcal{S}_u \cup \mathcal{S}_v$ can be constructed by setting $\mathcal{B}_l(k)$ to the bitwise OR of $\mathcal{B}^u_l(k)$ and $\mathcal{B}^v_l(k)$ for all $l \in [0, L)$ and $k \in [0, K)$.

Write a MapReduce program in Hadoop to estimate the number of distinct integers in any given sequence of integers. Scan each test sequence given for this task (see Appendix), and report your estimates for $K \in \{1, 2, 32, 64, 128\}$. For each estimate m' also include the ratio $\frac{m'}{m}$, where m is the actual number of distinct integers in the test sequence.

[1] Any hash function that outputs integers sufficiently distributed over the scalar range [0, 2^L) can be used.

EVALUATE($\mathcal{B}$)

1. $s \leftarrow 0$
2. ***for*** $k \leftarrow 0$ ***to*** $K-1$ ***do***
3. $\quad r \leftarrow \min_{l \in [0, L)} \{ \mathcal{B}_l(k) = 0 \}$
4. $\quad s \leftarrow s + r$
5. $m' \leftarrow \left\lfloor \frac{K}{0.77351} \times 2^{\frac{s}{K}} \right\rfloor$
6. ***return*** m'

Task 2. [30 Points] Estimating the Number of Reachable Nodes. Given a large directed graph $G = (V, E)$, we would like to estimate the number of vertices reachable from each vertex of V by scanning the edges in E only a few times. For this purpose we will use a counting bitmap from Task 1 for each vertex of the graph. The algorithm is given below (ESTIMATE-REACHABLES).

ESTIMATE-REACHABLES($G = (V, E)$) {*takes a directed graph $G = (V, E)$ as input, and for each $u \in V$, returns an estimate of the #nodes reachable from u*}

1. ***for*** each $u \in V$ ***do***
2. $\quad$ create a counting bitmap $\mathcal{B}^u$ with parameters K and L {*a counting bitmap from Task 1*}
3. $\quad \mathcal{B}^u_l(k) \leftarrow 0, \forall k \in [0, K)$ ***and*** $\forall l \in [0, L)$
4. $\quad$ UPDATE($\mathcal{B}^u$, u) {*initially, u is the only node reachable from u*}
5. $\quad \mathcal{N}(u) \leftarrow$ EVALUATE($\mathcal{B}^u$)
6. *done* $\leftarrow$ FALSE
7. ***while*** *done* = FALSE ***do***
8. $\quad$ ***for*** each $(u, v) \in E$ ***do***
9. $\quad\quad \mathcal{B}^u_l(k) \leftarrow$ BITWISE-OR($\mathcal{B}^u_l(k)$, $\mathcal{B}^v_l(k)$), $\forall k \in [0, K)$ ***and*** $\forall l \in [0, L)$ {*all nodes reachable from v are also reachable from u*}
10. $\quad$ *done* $\leftarrow$ TRUE
11. $\quad$ ***for*** each $u \in V$ ***do***
12. $\quad\quad t \leftarrow$ EVALUATE($\mathcal{B}^u$) {*in iteration $i \geq 0$, $\mathcal{B}^u$ estimates #nodes within distance 2^i from u*}
13. $\quad\quad$ ***if*** $t \neq \mathcal{N}(u)$ ***then*** *done* $\leftarrow$ FALSE {*if the estimate changes from previous iteration then we have not converged yet*}
14. $\quad\quad \mathcal{N}(u) \leftarrow t$
15. ***return*** $\mathcal{N}$ {*for each $u \in V$, $\mathcal{N}(u)$ is set to an estimate for the #nodes reachable from u*}

Implement a MapReduce version of ESTIMATE-REACHABLES in Hadoop. Run your program on each test case for this task and produce your estimates for $K \in \{1, 4, 16, 64\}$. For each test case report the number of times you had to scan the edges before convergence as well as the maximum, minimum and average values of the estimates for each K.

APPENDIX 1: Input/Output Format for Task 1

- **Input Format:** The first line of the input will contain one integer giving the number of integers (n) in the file. Each of the next n lines will contain one integer. Before running your program you are free to split the given input file into multiple files suitable for use by your program.
- **Output Format:** The output will consist of only number giving your estimate of the number of distinct integers.
- **Test Input/Output:** `/work/01905/rezaul/CSE590/HW4/T1/tests` on Lonestar.

APPENDIX 2: Input/Output Format for Task 2

- **Input Format:** The first line of the input will contain two integers giving the number of vertices (n), and the number of edges (m), respectively. Each of the next m lines will contain two integers u and v ($1 \leq u, v \leq n$) denoting a directed edge from vertex u to vertex v. Before running your program you are free to split the given input file into multiple files suitable for use by your program.
- **Output Format:** The output will consist of four integers: $\mathcal{N}_{max}$, $\mathcal{N}_{min}$, $\mathcal{N}_{avg}$ and c, where the first three numbers are the maximum, minimum and average (rounded to the nearest integer) of the estimates in $\mathcal{N}$, respectively, and c is the checksum value computed using the following function. Please make sure that you use a large enough datatype to avoid overflow during checksum computation.

```
CHECKSUM( G(V,E), 𝒩 )
1. c ← 0
2. for v ← 1 to |V| do
3.       c ← hash( c + 𝒩(v) )
4. return c
```

- **Test Input/Output:** `/work/01905/rezaul/CSE590/HW4/T2/tests` on Lonestar.

APPENDIX 3: What to Turn in

Please email one compressed archive file (e.g., zip, tar.gz) containing the following items to `cse590hw@cs.stonybrook.edu`.

- Source code (including the code for splitting input files, and for task 2 the code you used for computing checksums), makefiles and any other scripts required for running your code.
- Output files for test cases.
- A PDF document containing all answers.

APPENDIX 4: Things to Remember

- You can use Amazon Web Services (AWS) for your homework. You can access your account from the following page: `https://cse590.signin.aws.amazon.com/console`.
- The following is a tutorial prepared by our grader (Gaurav Menghani) on how to use Hadoop on AWS: `http://gaurav.im/files/aws-hadoop/presentation.html`.
- Please remember to terminate any instances on AWS that you no longer need. It turns out that the AWS credits burn pretty fast! So we need to be very careful with our usage. Please install Hadoop on your local machine for the development and debugging of you code, and use AWS only when you need a multi-node cluster.
- Please start working on the homework as early as possible so that you can identify any difficulties in using AWS and Hadoop as soon as possible, and solve them in time.