COMP5331 Knowledge Discovery in Databases (Fall Semester 2019)
Homework 3 Solution

Q1

(a)
Let N be the total number of items read so far.

The first 2t elements find their way into S.

When the sampling rate is $r \geq 2$,
we have $N = rt + \Delta$
where $\Delta \in [1, rt)$

Consider

$$N = rt + \Delta$$
$$\frac{1}{r} = \frac{t + \frac{\Delta}{r}}{N}$$
$$\frac{1}{r} \geq \frac{t}{N}$$

Consider an element e in S

The stored frequency $\tilde{f}$ of e is at most the true frequency f_0 of e

Why is the value of $\tilde{f}$ different from f_0? Why not the same?
This is because the stored frequency $\tilde{f}$ is deducted by tossing a coin repeatedly. If the outcome is "tail", the stored frequency will be decremented.

The reason why e still remains in S is that the outcome of the last toss is "head".

Note that

$$\text{P(Head)} = \frac{1}{r}$$
$$\text{and} \quad \text{P(Tail)} = 1 - \frac{1}{r}$$

Structure of the proof:

A. First, we prove the following

with confidence at least 1-δ,

$$f_0 - \tilde{f} \leq \varepsilon N$$

B. Second, with the above result, we prove the following

The algorithm returns the ε-deficient synopsis.

Part A

Lemma 1

Let C be a non-negative integer.

$\text{Prob}(f_0 - \tilde{f} = C) \leq \frac{1}{r}(1-\frac{1}{r})^C$

where r is the current sampling rate.

Proof:

Let x be the total number of the changes of sampling rate.

Let r_i be the sampling rate at the i-th time of changing sampling rate.

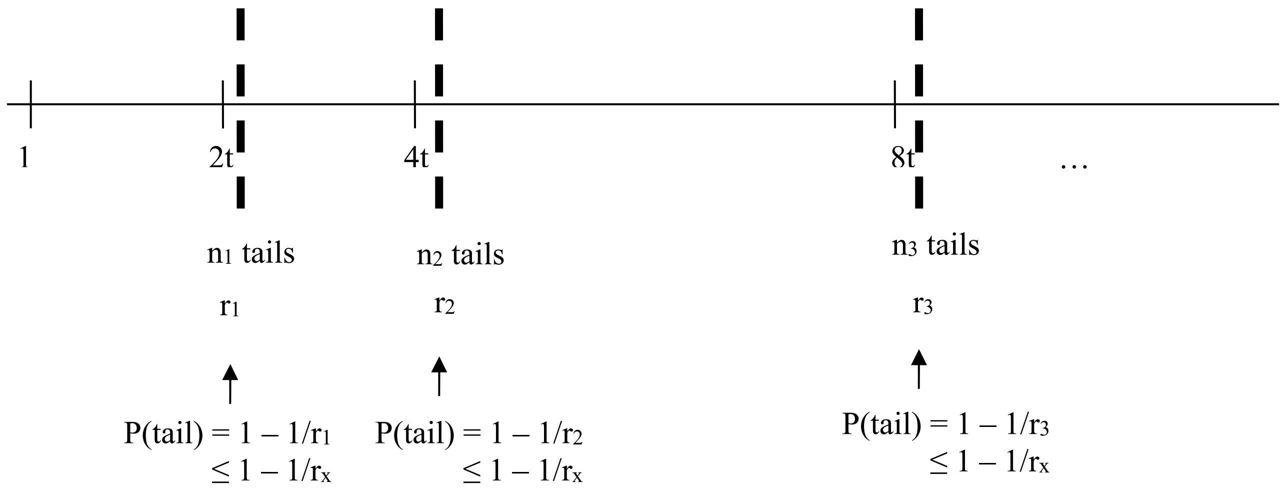
Consider the two scenarios.

The first scenario is that after each boundary, the entry for an item is still in the memory.

For the i-th time of changing sampling rate,

we decrement the stored frequency of e by n_i times

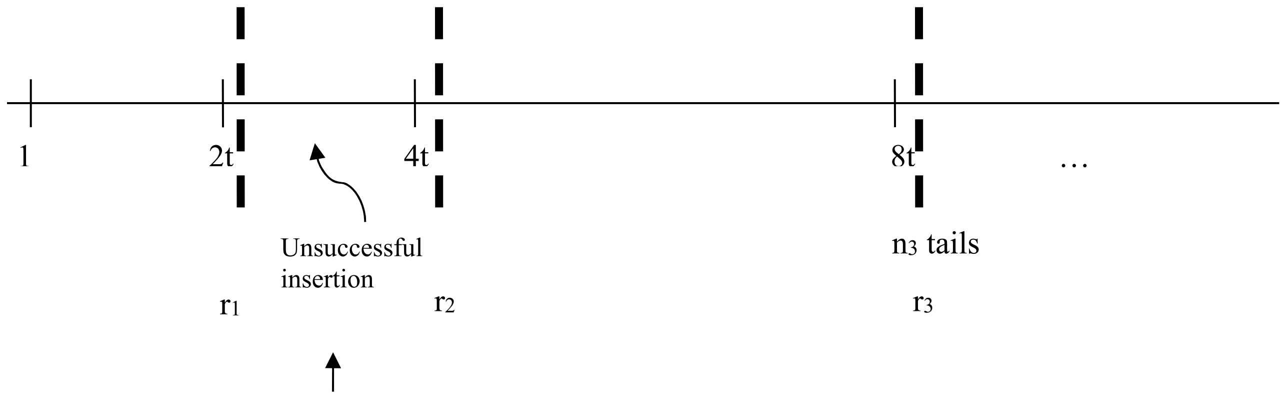
(i.e., there are n_i tails when we toss a coin repeatedly)



1
2t
4t
8t
…
n_1 tails
r_1
n_2 tails
r_2
n_3 tails
r_3
P(tail) = $1 - 1/r_1 \leq 1 - 1/r_x$
P(tail) = $1 - 1/r_2 \leq 1 - 1/r_x$
P(tail) = $1 - 1/r_3 \leq 1 - 1/r_x$

Thus, for each i in [1, x], we know that $P(\text{tail}) = 1 - 1/r_i$
$\leq 1 - 1/r_x$

The second scenario is that after the i-th boundary, the entry for an item is not in the memory.



$P(\text{Unsuccessful insertion}) = 1 - 1/r_1$
$\leq 1 - 1/r_x$

Thus, for each i in [1, x], we know that $P(\text{Unsuccessful insertion}) = 1 - 1/r_i$
$\leq 1 - 1/r_x$

Just after the x-th time of changing sampling rate,
$f_0 - \tilde{f}$ is equal to C

There are C times of frequency loss of f_o.

Each frequency loss is either (1) we observe a tail when we toss a coin (when we reach the boundary) or (2) we encounter an unsuccessful insertion (between two adjacent boundaries).

Since $P(\text{tail}) \leq 1 - 1/r_x$ and $P(\text{Unsuccessful insertion}) \leq 1 - 1/r_x$,
$P(\text{a frequency loss of } f_o) \leq 1 - 1/r_x$

Since there are C times of frequency loss of f_o
and
we know that the outcome of the last coin toss is head (since the entry for the item is still in the memory),
P(there are C times of frequency loss of f_o)
$$\leq \frac{1}{r_x}\left(1 - \frac{1}{r_x}\right)^C$$

Lemma 2

Prob (f_0 - $\tilde{f}$ ≥εN) ≤e$^{-εt}$

Proof:

We want to calculate the probability that the difference between f_0 and $\tilde{f}$ is at least εN (i.e., Prob (f_0 - $\tilde{f}$ ≥εN)).

Prob (f_0 - $\tilde{f}$ ≥εN)

= Prob (f_0 - $\tilde{f}$ =εN) + Prob (f_0 - $\tilde{f}$ =εN+1) + Prob (f_0 - $\tilde{f}$ =εN+2) + …

$$\le \frac{1}{r}(1-\frac{1}{r})^{\varepsilon N} + \frac{1}{r}(1-\frac{1}{r})^{\varepsilon N+1} + \frac{1}{r}(1-\frac{1}{r})^{\varepsilon N+2} + \dots$$

$$= \frac{1}{r}[(1-\frac{1}{r})^{\varepsilon N} + (1-\frac{1}{r})^{\varepsilon N+1} + (1-\frac{1}{r})^{\varepsilon N+2} + \dots]$$

$$= \frac{1}{r} \times \frac{(1-\frac{1}{r})^{\varepsilon N}}{1-(1-\frac{1}{r})}$$

$$= (1-\frac{1}{r})^{\varepsilon N}$$

$$\le (1-\frac{t}{N})^{\varepsilon N}$$ (because $\frac{1}{r} \ge \frac{t}{N}$)

≤e$^{-εt}$ (because $\lim_{N\to\infty}(1+\frac{-t}{N})^{N} = e^{-t}$)

Lemma 3

The probability that at least one of the frequent elements has $f_0 - \tilde{f} \geq \varepsilon N$ is at most δ

Proof:

The greatest no. of frequent elements (i.e., elements with frequency at least sN) is equal to

$= \frac{N}{sN}$

$= \frac{1}{s}$

Thus, the probability that at least one of the frequent elements has $f_0 - \tilde{f} \geq \varepsilon N$ is

$\leq e^{-\varepsilon t} \times \frac{1}{s}$

$= \frac{e^{-\varepsilon t}}{s}$

$= \frac{e^{-\varepsilon\left\lceil \frac{1}{\varepsilon}\log(s^{-1}\delta^{-1})\right\rceil}}{s}$

$\leq \frac{e^{-\varepsilon(\frac{1}{\varepsilon}\log(s^{-1}\delta^{-1}))}}{s}$

$= \delta$

Lemma 4

With confidence at least 1-δ,

$f_0 - \tilde{f} \leq \varepsilon N$…………………………..(1)

Proof:

The probability that each frequent element has $f_0 - \tilde{f} \leq \varepsilon N$

= 1- probability that at least one of the frequent elements has $f_0 - \tilde{f} \geq \varepsilon N$

(the boundary case (i.e., when $f_0 - \tilde{f} = \varepsilon N$) is not handled here for the sake of simplicity)

$\geq 1-\delta$

In other words,

Prob ($f_0 - \tilde{f} \leq \varepsilon N$) $\geq 1-\delta$

We can also write

with confidence at least 1-δ,

$f_0 - \tilde{f} \leq \varepsilon N$…………………………..(1)

Lemma 4 is the result for Part A.

Part B

Next, we check the three conditions of the ε-deficient synopsis.

The second condition is satisfied with confidence at least 1-δ.

Consider the first condition.

Consider an item with true frequency $f_0 \geq sN$

From (1),

$$\tilde{f} + \varepsilon N \geq f_0$$

$$\Rightarrow \quad \tilde{f} + \varepsilon N \geq sN$$

We conclude that

an item has true frequency $f_0 \geq sN$

➔ it has $\tilde{f} + \varepsilon N \geq sN$ (during the execution of the algorithm)

➔ this item is in the algorithm output

Thus, the first condition is satisfied.

Consider the third condition.

Consider an item with true frequency $f_0 \leq (s-\varepsilon)N$

Thus, we have $f_0 \leq sN - \varepsilon N$

$$f_0 + \varepsilon N \leq sN$$

$$\tilde{f} + \varepsilon N \leq sN \qquad \text{(since } \tilde{f} \leq f_0\text{)}$$

We conclude that this item is NOT in the algorithm output (i.e., it is classified as an infrequent item).

Thus, the third condition is satisfied.

(b) Differences:

1. Sticky Sampling Algorithm is a probabilistic algorithm.
   Lossy Counting Algorithm is a deterministic algorithm.
2. Storage of Sticky Sampling Algorithm is independent of ε
   Storage of Lossy Counting Algorithm is dependent on ε

(c) Differences:

1. Lossy Counting Algorithm requires an error parameter ε which we can deduce the memory storage
   Space-saving Algorithm requires the memory parameter M which we can deduce the error
2. Given the same memory (i.e., the max. memory that we can use), Space-Saving Algorithm has smaller error than Lossy Counting Algorithm.

Q2.

(a) (i)w= $\frac{1}{0.2} = 5$

(ii) (1) $(I_2,1,0)$

$(I_3,2,0)$

$(I_4,1,0)$

$(I_5,1,0)$

(2) $(I_3,2,0)$ (Remove I_2, I_4, I_5 since $f + \Delta \le 1$)

(iii)(1) $(I_2,2,1)$

$(I_3,2,0)$

$(I_4,3,1)$

(2) $(I_2,2,1)$

$(I_4,3,1)$ (Remove I_3 since $f + \Delta \le 2$)

(iv) (1) $(I_1,2,2)$

$(I_2,3,1)$

$(I_4,3,1)$

$(I_6,1,2)$

$(I_7 1,2)$

(ii) $(I_1,2,2)$

$(I_2,3,1)$

$(I_4,3,1)$ (Remove I_6, I_7 since $f + \Delta \le 3$)

(v) $\varepsilon N = 0.2 \times 15 = 3$
$sN = 0.4 \times 15 = 6$

Consider $(I_4,3,1)$

$\because f + \varepsilon N \ge sN$ (i.e. $3 + 3 \ge 6$)

$\therefore I_4$ is the output of the algorithm.

Consider $(I_2,3,1)$

$\because f + \varepsilon N \ge sN$ (i.e. $3 + 3 \ge 6$)

$\therefore I_2$ is the output of the algorithm.

Consider $(I_1,2,2)$

$\because f + \varepsilon N < sN$ (i.e. $2 + 3 < 6$)

$\therefore I_1$ is not the output of the algorithm.

Output:{ I_2, I_4 }

(b)

Optional solution 1 (mining frequent itemsets):

(i) memory $= \sum_{i=1}^{m} \binom{m}{i} \frac{1}{\varepsilon} (\log \varepsilon N)$

(OR $(2^M - 1)\frac{1}{\varepsilon}(\log \varepsilon N)$ where M is the total number of items.).

(ii) It is not nicely bounded because the memory consumption contains an exponential factor on m or M.
Optional solution 2 (mining frequent items):

(i) memory $= m \cdot \frac{1}{\varepsilon}(\log \varepsilon N)$

(ii) It is nicely bounded.

Q3

(a)(i)

Let M be the memory size (in *bytes* (not the *number of entries* in form of (e, f, Δ) discussed in class)).
We divide the memory into two parts:

1. Part 1: buffer which stores the incoming data
2. Part 2: summary of previous batches.

We allocate the memory for part 1 with B bytes. And the memory for part 2 is M-B bytes.

Since B is given, the memory size for Part 1 and the memory size for part 2 are fixed.
We divide the memory for Part 2 into 4 partitions of equal size (We do not divide the memory into 4 partitions of adjustable size because in the worst case, in order to minimize the error, we also need the equal-sized partitions.)
Thus, we have four batches. Each batch is associated with a summary. For each batch, we can update the summary (associated to the batch) like what the original space-saving algorithm does.
Given a set S of data points, the summary stored in the Space-Saving algorithm for this set is denoted by SpaceSaving(S). Let X = SpaceSaving(S). X contains two components. The first component denoted by X.E contains a list of entries each in form of (e, f, □Δ) where e is the item number, f is the frequency of the item (recorded after this entry is created) and Δ is the maximum possible error in f. The second component denoted by X.p is equal to the variable p_e used in algorithm Space-Saving discussed in class (i.e., the greatest possible frequency error).

Output:

Let f(e, B_i) be the count of element e stored in the summary of batch B_i.
If the entry (e, f, □Δ) for e exists in the summary of batch B_i, f(e, B_i) is equal to f. Otherwise, it is equal to 0.

Let Δ(e, B_i) be the Δ value of element e stored in batch B_i.
If the entry (e, f, □Δ) for e exists in the summary X of batch B_i, Δ (e, B_i) is equal to Δ. Otherwise, it is equal to X.p.

For each item e, calculate $f_e = \sum_{i=1}^{4} f(e, B_i)$ and $\Delta_e = \sum_{i=1}^{4} \Delta(e, B_i)$

Get a list of items where $f_e + \Delta_e \geq sN$ where N=4B

(ii)

The memory for part 2 is M-B. Since the memory for part 2 is divided into 4 equal-sized parts, each part occupies (M-B)/4.

Consider a part containing a summary X.
Assume that the second component of X (i.e., X.p) occupies 1 byte.
Since the second component of X occupies 1 bytes, the first component of X occupies (M-B)/4 - 1.

Let m be the size occupied by a single entry in form of (e, f, Δ).
The total number of entries each in form of (e, f, Δ) is equal to [(M-B)/4 – 1]/m.

For each batch, the greatest error in fraction = $\dfrac{1}{\dfrac{\frac{M-B}{4}-1}{m}} = \dfrac{4m}{M-B-4}$.

The overall greatest error in count = $\dfrac{4m}{M-B-4} \times 4B = \dfrac{16mB}{M-B-4}$

The overall greatest error in fraction = $\dfrac{16mB}{M-B-4} \times \dfrac{1}{4B} = \dfrac{4m}{M-B-4}$

(b)(i)The answer of (b)(i) is the same as the answer of (a)(i). But we need to update the output part.
Note that the incomplete batch contains all items over data streams instead of the summary.
Let f(e, B_i) be the count of element e stored in batch B_i. The definition is the same as (a)(i).

Let Δ(e, B_i) be the Δ value of element e stored in batch B_i. The definition is the same as (a)(i).

Let I be the incomplete batch.
Let f(e, I) be the exact count of element e in the incomplete batch I.

Consider batch B_1, for each item e, calculate Y_e=min{ f(e, B_1)+Δ(e, B_1),0.5B}
Consider batch B_2, B_3, B_4, I,

For each item e, calculate $f_e=\sum_{i=2}^{4} f(e, B_i)$ +f(e, I) and $\Delta_e=\sum_{i=2}^{4}\Delta(e, B_i)$ +0

Get a list of items where $f_e+\Delta_e+Y_e \geq sN$ where N = 4B.

(ii) For each of batches B_2, B_3, B_4,
the greatest error in fraction = $\dfrac{1}{\dfrac{\frac{M-B}{4}-1}{m}} = \dfrac{4m}{M-B-4}$.

For batch I, the greatest error = 0 in both count and fraction.
For batch B_1, the greatest error in fraction is 0.5, so the greatest error in count is 0.5B.

The overall greatest error in count = $\dfrac{4m}{M-B-4} \times 3B$ +0+0.5B= $\dfrac{12mB}{M-B-4}$ +0.5B

So, the overall greatest error in fraction = $\dfrac{1}{4B} \times (\dfrac{12mB}{M-B-4} + 0.5B) = \dfrac{3m}{M-B-4} + \dfrac{1}{8}$.

Q4

a. We propose a dominance graph G for the data structure.
V is a set of points in the sliding window which may become a skyline point currently or later.
E is a set of edges where each edge corresponds to a dominance relationship, e.g., an edge (u, v) corresponds to that point u dominates point v.

Besides, we maintain a list L containing all points (or nodes) with in-degree = 0.

We present the algorithm as follows.

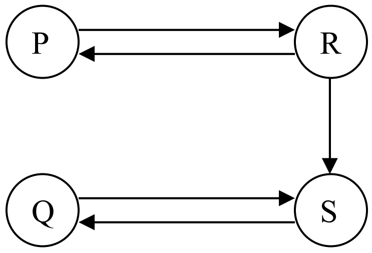
Algorithm

```
1.  -$L \leftarrow \Phi$
2.  -$V \leftarrow \Phi$
3.  -$E \leftarrow \Phi$
4.  -$G \leftarrow (V, E)$
5.  -When there is a new point $P_{new}$,
6.      -if there is a point $P_{old}$ which expires,
7.          -remove point $P_{old}$ from V
8.          -remove all edges involving $P_{old}$ from E
9.          -for each edge removal, update L if $\exists$ a point has in-degree = 0
10.             (by inserting this point with in-degree = 0 into L)
11.     -$V \leftarrow V \cup \{P_{new}\}$
12.     -for each point $q \in V$ which is dominated by $P_{new}$,
13.         -$E \leftarrow E \cup \{P_{new}, q\}$
14.         -for each edge addition, update L if $\exists$ a point in L has in-degree updated to non-zero
15.             (by deleting this point from L)
16.     -for each point $q \in V$ which dominates $P_{new}$,
17.         -$E \leftarrow E \cup \{q, P_{new}\}$
18. -Output all points (or nodes) with in-degree = 0 (by using L).
```

It is easy to see that the output cost is O(L) = O(|S|).

Q5

(a)



$$\begin{tikzcd} P \arrow[r, shift left] & R \arrow[l, shift left] \arrow[d] \\ Q \arrow[r, shift left] & S \arrow[l, shift left] \end{tikzcd}$$

Stochastic matrix:

$$\begin{array}{c|cccc} & P & Q & R & S \\ \hline P & 0 & 0 & 0.5 & 0 \\ Q & 0 & 0 & 0 & 1 \\ R & 1 & 0 & 0 & 0 \\ S & 0 & 1 & 0.5 & 0 \end{array}$$

(b) Yes. The spider trap corresponds to a set {Q, S}.

(c) Equation to be solved:

$$\begin{pmatrix} P \\ Q \\ R \\ S \end{pmatrix} = 0.8 \begin{pmatrix} 0 & 0 & 0.5 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0.5 & 0 \end{pmatrix} \begin{pmatrix} P \\ Q \\ R \\ S \end{pmatrix} + \begin{pmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.2 \end{pmatrix}$$

(d)

	1	2	3	4	5	6	7	…	13
P	1	0.6	0.6	0.47	0.47	0.43	0.43	…	0.41
Q	1	1.0	1.32	1.32	1.42	1.42	1.46	…	1.47
R	1	1.0	0.68	0.68	0.58	0.58	0.54	…	0.53
S	1	1.4	1.4	1.53	1.53	1.57	1.57	…	1.59

So the ranking is S, Q, R, P.