الجامعـــة الإسلاميـــة للتكنولوجيا
UNIVERSITE ISLAMIQUE DE TECHNOLOGIE
ISLAMIC UNIVERSITY OF TECHNOLOGY
DHAKA, BANGLADESH
ORGANISATION OF ISLAMIC COOPERATION

Co-ordination of Streams of Data

Authors

Abdullah Al Mamun	**Student Id: 104425**
Tahmeed Hasan	**Student Id: 104431**

Supervisor

Dr. Mohammad Rezwanul Huq

Department of Computer Science & Engineering (CSE)
Islamic University of Technology (IUT)

A Thesis submitted to the Department of Computer Science and Engineering (CSE), Islamic University of Technology in Partial Fulfillment of the requirements for the degree of Bachelor of Science in CSE (Computer Science & Engineering)

September, 2014

CERTIFICATE OF RESEARCH

This is to certify that the work presented in this thesis paper is the outcome of the research carried out by the candidates under the supervision of **Dr. Mohammad Rezwanul Huq, Assistant Professor, Department of Computer Science and Engineering, IUT Gazipur.** It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree or any judgment.

Authors

_____ _____

Signature of Supervisor

_____

Signature of the Head of the Department

_____

ABSTRACT

Here in this thesis work we have proposed a model for some issues from a new data processing model. In this type of model data we are dealing with does not possess any relation that is persistent, rather this data have some different properties. They arrive in real time, continuous, time varying, rapid and infinite *streams of data*. This paper introduces a model for the delayed and unordered arrival of streams. In addition, this thesis work also introduces the concept of using buffers and algorithmic issues for stream management purpose reviewing the past related works and also the ongoing research in this area.

This thesis proposes a model and an algorithm that will deal with the coordination of the tuples that do not arrive in due time and in some cases they just lost forever. The model introduces some modules that will depict the total work flow and the algorithm will propose some data set that will enable one to choose the trade-off depending on the system requirements implementing the ordering mechanism.

ACKNOWLEDGMENT

At the very beginning we express our heartiest gratitude to Almighty Allah for His divine blessings which allowed us to do this research work to life.

We are grateful and indebted to our supervisor **Dr. Mohammad Rezwanul Huq,** Assistant Professor**,** Department of Computer Science and Engineering, IUT. Due to his supervision, knowledge and relentless support allowed us to complete this endeavour successfully.

We are thankful to **Prof. Dr. M. A. Mottalib**, Head of the department, Computer Science and Engineering, IUT. Also our appreciation extends to all the respected faculty members of the Department of Computer Science and Engineering, IUT.

Finally we would like to extend our thanks to our friends, students, staffs and everyone else who have contributed to this work in their own way.

CONTENTS

Chapter 1

INTRODUCTION

With the advancement of the technology we have to deal with more data set efficiently. In many cases now a whole lot of data should be dealt with minimum amount of time (e.g. reading from sensors, radars). Hence the concept of streams comes into being. Management of streams of data leads to the concept of streaming database where large amount of data are management within a minimum amount of time. With the requirement of time the traditional query processing will not be enough for this kind of data which are constantly in a motion. These data are needed to be processed according to the adaptive dataflow.

In emerging networking systems, the commodity of interest is data. Like any usual commodity, its value is understood when it is moved to different places as per need. Unlike the traditional data processing systems where data is predicted to reside in known locations statically, data in recent days for many applications are continuously changing and dynamic. This fluidity nature of data leads us to consider management of data for these recent applications as an adaptive dataflow processing that must observe, monitor and react to information streams as they pass through any kind of network.

Streaming databases are mostly used now a days for managing continuous arriving of infinite timely data which we are referring to as streams. These Databases offer flexible query processing and continuous queries. Continuous queries are those which generates the output based on the rapid incoming values. Processing dataflow for applications often has an active data monitoring or filtering aspect for the query. Whenever new data arrives at the module that processes queries, it is then routed through the set of queries which are continuously active. Such continuous query processing needs a different approach rather a traditional one in case of managing database. What happens in the traditional system is that the queries are allowed access to a collection of data while in this case the arrival of streams of data initiates access to a predefined set of queries. As expected this deviation prompts a very different approach to query processing system.

Another important property of these continuous queries is that the streams they deal with can be infinite. Queries that process infinite streams of data require

different semantics, non-blocking operators and exception handling. Operators should return the incremental results continuously and need to be defined in such a way that can process the window of the input. Continuous queries can be long lived so, care must be taken to reduce the states that are required in processing.

For error free output, tuples in the input stream must be in correct order. For databases that manages streams of data, co-ordination of streams can be a big challenge, as the output can become erroneous for that moment of time. When the incoming streams contain unordered data, the output will correspond to the result for those incoming data. But hence data arriving in a specific moment of time must get processed accordingly to generate output that will be related to the arrival time. So not only producing output yields a correct result but generating output referring to the incoming time will be considered as an errorfree outcome.

As expected there are two parameters that we have identified, one is time and other one is accuracy. When we allow the time to expand over infinite range, accuracy also increases but waiting for infinite amount of time is not feasible. Again if we want increased performance with respect to time that means lower processing latency, it also degrades the accuracy level. So, our goal is to find the best trade-off between time and accuracy for an optimize performance.

Chapter 2

RELATED WORKS

In the STEAM framework in [1] authors have proposed a model for a sensor based streaming system. They have described the functionality and the key components of the STEAM system. They have introduced a mechanism for calling individual streams. Also they have described the mechanism for handling multiple streams and sharing queries between them. They have also examined the scheduling of the query operators and also gave a detail description of a class of join algorithms, the window join for joining multiple data streams. Their research work is basically based on video streaming offering comprehensive and efficient database management for real time query managing and processing.

The key contribution of their research are:

• A stream processing framework with a powerful stream management system that is capable of working and performing the basic operations of a multisensory network.

• A class of algorithm for joining multiple streams that are non-blocking and can be easily implemented in the pipeline system.

• Application development based on the stream database framework for online data mining and analysis techniques,

In their proposed model they have used some components like stream manager for managing the buffer and the streams, a scheduler for scheduling the queries and a stream scan for scanning the streams and send it to the stream manager. In their research work they have introduced a double buffer system that will reduce the chance of buffer overflowing.

The major sectors of improvements or we can say disadvantages of their research work are:

• We have reduced the component number of the model to reduce inter latency time (discussed in the parameters part.).

• They have not specify how the main buffer and temporary storage will be manageable, that is specified in our work.

• Also their fetching data's from the temporary storage separately that will also cost more time but in our model we have used the buffer always for incoming or outgoing of the data.

In search based buffer management policies in [2] authors have introduced buffer replacement and management policies for CM (continuous media) servers based on the information from both searching and streaming process. They have tested their approach in the real time video database system. They have also compared the buffer policy and search policy. Their work is also based on the video streaming.

The key contribution of their research are:

• Introducing some background work based on the CM (continuous media) server.
• Introduction of the weight based system applying the generalized CLOCK (G-CLOCK) algorithm for buffer replacement.

The major sectors of improvements or we can say disadvantages of their research work are:

• They have used too many separate module in their model of CL server that will cause huge latency. Our model is more simplified.
• They have not given any algorithm or real time example for the buffer management. We have given that.
• We have showed how the weight value is assigned and how the tuples are managed that is not described in their work.

Chapter 3

PROBLEM DEFINITION

In case of data streaming correctness of the tuples is very much important. There can be some cases where tuples may be lost or they may come later as unordered. We have to deal with certain issues when this cases occurs. This such phenomena creates problem that any kind of database must deal with. When the tuple order is not corrected the outcome of the result becomes less accurate. As it is quite obvious that if some tuple in stream one is in order and some tuple in the second stream are out of order, then for those tuples in the second stream which are not aligned with the first one may produce the erroneous result. Which will affect the outcome. And this may also cause a dramatic debase in the accuracy of the result processing. In this situation getting the correct result which is defined as cent percent accurate is not possible. Also there are some other issues which has been discussed later on.

If the tuples are out of order then we need to store the current tuples till the correct ordered tuple come. Not only that, we need find a way to order the tuples before they are sent to the processing unit. We must develop some algorithm that is suitable and working to find the results. There will be delays if the tuples are unordered. So the algorithm should have the ability to count and process the delay and find the accuracy of getting correct data according to the delay. There should be a better trade-off between the delay and accuracy as per as the system requirements and demand.

Chapter 4

CHALLENGES & PARAMETERS

4.1 Challenges:

The problem we are dealing with definitely had some challenges by fighting which we were expected to get a desired research outcome. In any research work soon after the problem has defined, next comes the challenges we need to face to achieve the proper model for this one. So after we are done with our problem definition that is statement of the problem that is on which of the issues we are targeting and trying to get a way around the problem. We have specified the major obstacles for this issue and here some key challenges are discussed at length.

1. **Designing a model for the whole process:** Well as we are discussing the coordination of the tuples that is in the streams, so first we need an appropriate model which will depict the processing of the streams and will also show the workflow of how the streams are handled and the different modules are working. So designing was the first challenge we faced for developing a suitable model.

2. **Generate and discuss some case studies:** A case study is an account of an activity, event or problem that contains a real or hypothetical situation and includes the complexities that might occur during actual workplace. Case studies are used to plot the complexities to prepare necessary measures. Here we have introduced some cases regarding the co-ordination issue that we want to reduce. These case studies covers every possible actions a system might face. This was also challenging because it is not at all easy to plot a real world scenario by just assuming. There are situations which cannot be predicted earlier and those cases are defined as the special cases. Yet we covered every possible phase that may occur in regular system.

3. **Developing some ordering techniques:** Now that system model and the taste cases are generated now the real problem comes named how the ordering should be managed. So to say this challenge was the most difficult one to deal with. Because this was the main point of how the coordination between the tuples should be done. The ordering techniques must be in such a way that it must be faster and efficient. The ordering part of the algorithm should have the property of optimized run time. This ordering might take less time and with the efficient algorithm it should be an easy procedure so that it can be implemented easily. To order the delayed tuple we must mark them first and store them, then the comparison begins with the first stream. That's where we have introduced a weight value to mark and swap the tuples.

4. **Developing the algorithm:** An algorithm is a process or set of rules to be followed in calculations or other problem-solving operations. Algorithms are mainly used for calculating complex problems which needs step by step debugging. It is an effective method expressed as a finite list of well-defined instructions for calculating a function. Starting from an initial state and initial input that can also be empty, the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate random input. The algorithm is the key to devise a perfect model for this coordination issue. So we have given the topmost priority to the algorithm as it describes the total flow.

5. **Evaluation of the results:** Evaluating the outcome and compare the performances at different parameters was another important aspect for choosing the trade-off we were looking for. The algorithm, by evaluating the sample data and graph, enables the user to choose the appropriate trade-off between delay and accuracy in any particular system.

4.2 Parameters:

There are some key terms that needed to be understood in order to get a clear idea about the research area. These key terms are considered as the parameter.in this section we are describing some common and must needed parameters related to our work.

Data stream [S_i]: A continuous, ordered and infinite sequence of tuples. When large amount of data comes at a time, then it is called streams of data. In Connection-oriented communication, a data stream is a sequence of digitally encoded coherent signals (packets of data or data packets) used to transmit or receive information that is in the process of being transmitted.

In electronics and computer architecture, a data flow determines for which time which data item is scheduled to enter or leave which port of a systolic array, a Reconfigurable Data Path Array or similar pipe network, or other processing unit or block.

Often the data stream is seen as the counterpart of an instruction stream, since the von Neumann machine is instruction-stream-driven, whereas its counterpart, the Anti-machine, is data stream driven.

Tuple [t_i]: Incoming records residing in streams. Streams of data is basically is a summation of tuples. They can be considered as a unit of data. A tuple is an ordered list of elements.

Tuples are often used to describe other mathematical objects, such as vectors. In computer science, tuples are directly implemented as product types in most functional programming languages. More commonly, they are implemented as record types, where the components are labeled instead of being identified by position alone. This approach is also used in relational algebra. Tuples are also used in relation to programming the semantic web with Resource Description Framework or RDF.

Buffer: Temporary memory storage for stream processing. Buffer is a storage that is used to store data for a short period to use it in processing the query.

In computer science, a data buffer (or just buffer) is a region of a physical memory storage used to temporarily store data while it is being moved from one place to another. Typically, the data is stored in a buffer as it is retrieved from an input device (such as a microphone) or just before it is sent to an output device (such as speakers). However, a buffer may be used when moving data between

processes within a computer. This is comparable to buffers in telecommunication. Buffers can be implemented in a fixed memory location in hardware—or by using a virtual data buffer in software, pointing at a location in the physical memory. In all cases, the data stored in a data buffer are stored on a physical storage medium. A majority of buffers are implemented in software, which typically use the faster RAM to store temporary data, due to the much faster access time compared with hard disk drives. Buffers are typically used when there is a difference between the rate at which data is received and the rate at which it can be processed, or in the case that these rates are variable, for example in a printer spooler or in online video streaming.

A buffer often adjusts timing by implementing a queue (or FIFO) algorithm in memory, simultaneously writing data into the queue at one rate and reading it at another rate.

Window (W): A bounded set of tuples over which the operation takes place. A set of tuples that is suitable for the given operation is set, and used as a limit of the processing of the queries.

Instead of using synopses to compress the characteristics of the whole data streams, window techniques only look on a portion of the data. This approach is motivated by the idea that only the most recent data are relevant. Therefore, a window continuously cuts out a part of the data stream, e.g. the last ten data stream elements, and only considers these elements during the processing.

There are different kinds of such windows like sliding windows that are similar to FIFO lists or tumbling windows that cuts out disjoints parts. Furthermore, the windows can also be differentiated into element based to consider, e.g. the last ten elements, or time based windows, e.g. to consider the last ten seconds of data. Additionally, there are also different approaches to implement windows. There are, for example, approaches that use timestamps or time intervals for system-wide windows or buffer-based windows for each single processing-step.

Window type [W_{type}]: Window can be of different types. We are considering tuple/time based window. This window is measured by the size of the tuples.

Window size [W_{size}]: The maximum number of tuples in a stream. Here we are considering the window size 3 of type tuple-based window.

Buffer size: Maximum number of tuples it can store. Buffer size can be anything. But to find a better tradeoff between time and accuracy we need to set a proper sized buffer.

Processing latency [PL]: the time required for the processing of tuples while in buffer. Processing time can be different depending on tuple number, buffer size and waiting time for the tuple arrival. Latency is a time interval between the stimulation and response, or, from a more general point of view, as a time delay between the cause and the effect of some physical change in the system being observed. Latency is physically a consequence of the limited velocity with which any physical interaction can propagate. This velocity is always lower than or equal to the speed of light. Therefore every physical system that has spatial dimensions different from zero will experience some sort of latency, regardless of the nature of stimulation that it has been exposed to.

The precise definition of latency depends on the system being observed and the nature of stimulation. In communications, the lower limit of latency is determined by the medium being used for communications. In reliable two-way communication systems, latency limits the maximum rate that information can be transmitted, as there is often a limit on the amount of information that is "in-flight" at any one moment. In the field of human–machine interaction, perceptible latency has a strong effect on user satisfaction and usability.

Inter arrival time [IAT]: the time difference between two incoming tuples. It can be changed for tuple loss or late arrival. Inter arrivals provide a method of creating object arrivals at regular intervals without the need to specify detailed arrival profiles. Inter arrivals require that the total number of objects (or unlimited) to arrive is specified along with how often one object (or a batch) arrive. Variables can be defined as either constant values, distributions (using the distribution wizard) or predefined variables. The value type list box determines which value is applied to the simulation.

When defining an Inter Arrival Time the following Simulation propertieswill apply:

4.2.1 Maximum Arrivals:

Maximum number of objects that can arrive throughout the simulation run. Maximum Value Type-- you may specify that the Maximum Value Type is Constant, Distribution, or Variable. Note that if you specify one of these choices, then you only need to fill in the accompanying property from the set below:

Constant: Type in the number of Objects that you want to run the experiment for. For example, when simulating a Car Wash, you might want to look at 500 cars. You would enter the number 500 in this field.

4.2.2 First Arrival At:

The time of the first object to arrive in the model. You may specify that the First Arrival At property is Constant, Distribution, or Variable.

Constant: the Constant Arrival Time by default is set to 0. If you set it to 60 by putting the number 60 in the Constant field of the First Arrival At property, the model will be simulated for 60 time units before the first Object would be brought into the system.

4.2.3 Inter Arrival Time:

Time between the objects arrivals. The time of the first object to arrive in the model. You may specify that the Inter Arrival Time property is Constant, Distribution, or Variable.

4.2.4 Lot Size:

Number of objects that arrive together. The time of the first object to arrive in the model. You may specify that the Lot Size property is Constant, Distribution, or Variable.

Valid/explicit time [$E_{xplicit}$]: the time when the data was originated. The time is count when the data was created from any input source. Valid time is the time period during which a fact is true with respect to the real world. Transaction time is the time period during which a fact stored in the database is considered to be true. Temporal data combines both Valid and Transaction Time.

Transaction/implicit time [$I_{mplicit}$]: the time when the data get processed. The time from one processing unit to another is measured as the transaction time. Transaction time (TT), a concept originated by Richard T. Snodgrass and his doctoral student, is used in temporal databases.[1] It denotes the time period during which a fact stored in the database is considered to be true. In a database table transaction time is often represented by two extra table-columns StartTT and EndTT. The time interval is closed at its lower bound and open at its upper bound.

When the ending transaction time is unknown, it may be considered as "Until Changed". Academic researchers and some RDBMS have represented "Until Changed" with the largest timestamp supported or the keyword "forever". This convention is not technically precise.

Inter latency time (T_l): the time needed for a tuple or data unit to go form one module to another.it can be different based on module number, size and type and path it is travelling. Regardless of the speed of the processor or the efficiency of the software, it takes a finite amount of time to manipulate and present data. Whether the application is a web page showing the latest news or a live camera shot showing a traffic jam, there are many ways in which an application can be affected by latency. Four key causes of latency are: propagation delay, serialization, data protocols, routing and switching, and queuing and buffing.

Chapter 5

PROPOSED MODEL

The model we are proposing is basically an improvised model from the models that has been discussed in the related works. There are five basic components in the model:

A. Stream reader: stream reader is used to take the input streams and send them to the ordering module and also take the output stream from the buffer and deliver it to the query manager. This is the module where the incoming streams are received. The tuples in the stream may occur in any order. Here it is important to mention that Stream Reader only receives the streams, it does not sort or convey other information about the streams. It only passes the incoming streams to the next unit which we named the Ordering module.

B. Ordering module: the ordering module takes input stream from the stream reader and order it. It also assign a weight value (Boolean value 0 or 1) to the tuples and send them to buffer for storing. Ordering module is the central and most important unit in our model. Ordering module takes the streams passed by the stream reader and then it processes them. Ordering module detects the tuple which are unordered and also keeps track of the in time streams. It has a buffer connected with it which is used for the correct ordering of the tuples. Ordering module actually sends the timely stream to the buffer from where the streams are retrieved when the operation is performed. The Ordering module also sends the delayed stream to the buffer and the stream stays there until the delayed tuple is arrived or time out occurs. Well time out is defined by the specific amount of time set by the user to wait for the delayed tuple to arrive. If that time limit is exceeded then the operation for those streams are aborted as well. So time out is the maximum delay time an operation can sustain. The tuples in the delayed streams are stored in the buffer, but what happens when the delayed tuples arrive?

A Weight value is introduced for this kind of scenario, Out of order tuples are identified by the weight value and thus the operation waits till ordered streams are processed. If the desired tuple takes too much time to arrive then there has been a chance for buffer overflowing, and so for that purpose we have introduced a

temporary storage. Where the tuples that are not in order will stay. And after the arrival of delayed tuple. The stream's tuple gets ordered on the basis of the weight value. Then the contents of the temporary storage also get updated. When both the streams in the buffer has the same numbered tuple, they both are popped from the buffer for the query execution. They are again passed to the stream reader which forwards those streams to query manager where the desired query is performed. This time the stream reader gets the ordered streams to forward.

C. Current buffer: Current buffer is used to store the tuples with the assigned weight values. Based on the weight values, the ordering is done. Current buffer holds the tuples that are in order in the streams. The out of order streams are instantly replaced from the temporary storage. So the buffer waits for the tuples to get sorted in the streams. Then it passes the sorted streams to the query manager for the operation.

D. Temporary storage: Temporary storage is used to store the tuples if the current buffer is overflowed. Temporary storage is the final repository for the tuples. The delayed tuple waits here and only called when the desired tuple arrives. If the delayed and desired tuple does not arrive within the time out then the contents of the current buffer is replaced by the contents of the temporary storage. And then the further processing is continued. The size of the temporary storage should be considerably large.

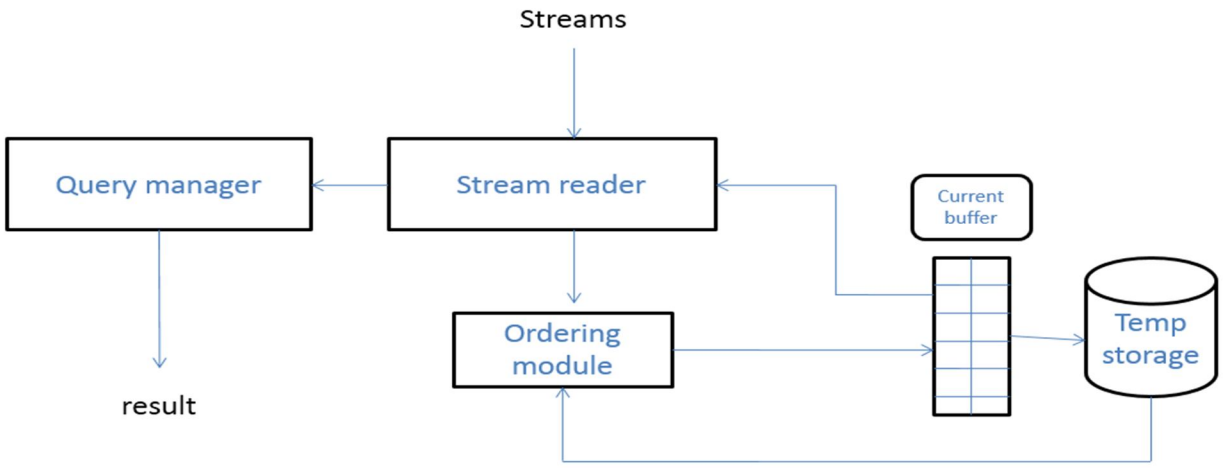


Figure 1: proposed model for ordering tuples of streaming data

E. Query manager: Query manager processes the query with the tuples that stream reader takes from the current buffer. Query manager always receives the streams which have the tuples ordered. As they have been processed by the ordering module and the buffer, the query manager takes the corrected streams from the stream reader as inputs and then it performs the desired user defined or predefined queries.

5.1 Ordering module: All about the 'weighting'

One possible solution is to assign weight values to the incoming tuples. Here in the figure 2 we can see how the tuples are coming and weight value is assigned to them when they are inserted into the database. We can see form the figure that tuple 6, 7, 8 from stream s1 are in order so they have the weight value 1. Tuple 9 from stream s2is out of order and so it has the value 0. When the buffer is overflowed the next tuple will go to the secondary storage. Now as tuple 8 from stream s2 has the weight value 1 it will replace the tuple 9 in the main buffer and tuples will be rearranged in order again. The change is shown in the figure 3.

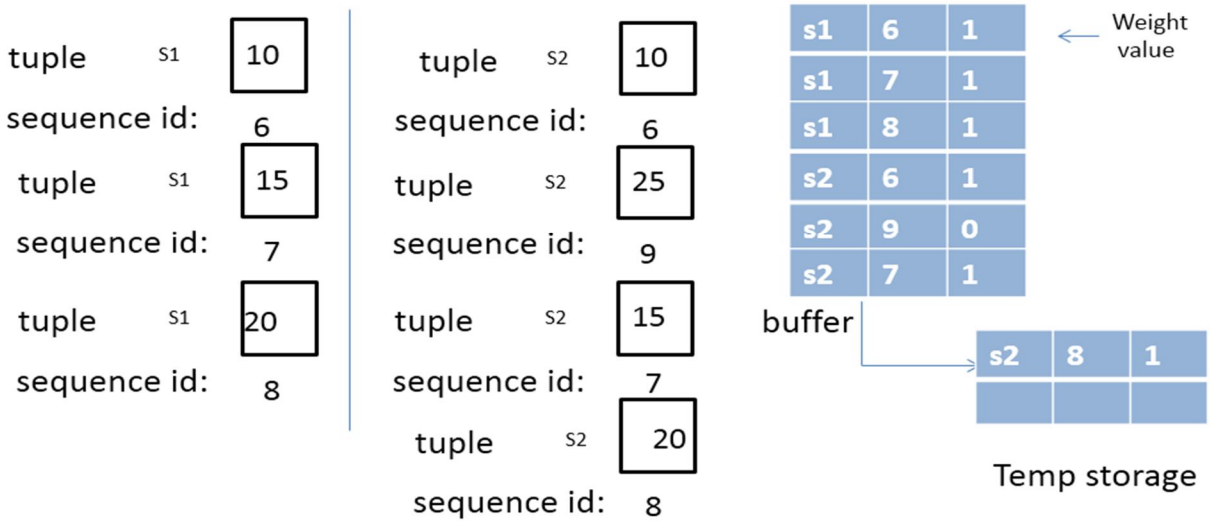


Figure 2: before swapping the tuple

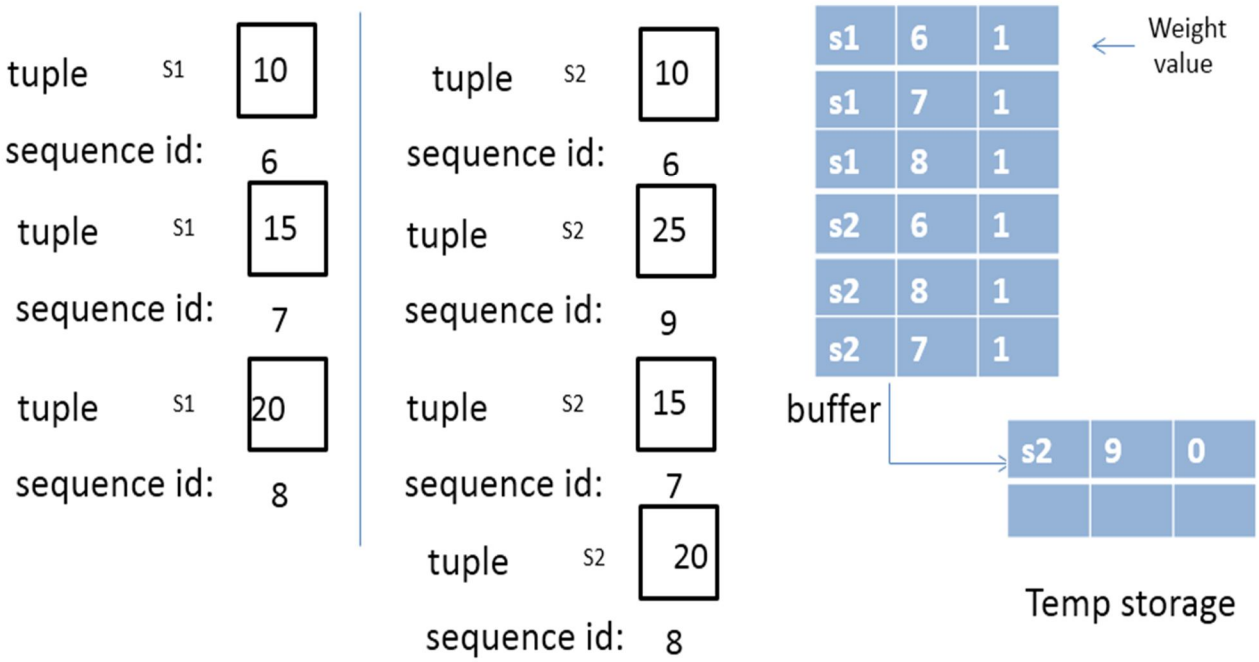


Figure 3: after swapping the tuple

Chapter 6

ALGORITHM:

In mathematics and computer science, an algorithm is a step-by-step procedure for calculations. Algorithms are used for calculation, data processing, and automated reasoning.

An algorithm is an effective method expressed as a finite list [of well-defined instructions for calculating a function. Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate random input.

In computer systems, an algorithm is basically an instance of logic written in software by software developers to be effective for the intended "target" computer(s) for the target machines to produce output from given input (perhaps null).

"Elegant" (compact) programs, "good" (fast) programs: The notion of "simplicity and elegance" appears informally in Knuth and precisely in Chaitin:

Knuth: ". . . we want good algorithms in some loosely defined aesthetic sense. One criterion . . . is the length of time taken to perform the algorithm. ... Other criteria are adaptability of the algorithm to computers, its simplicity and elegance, etc"
Chaitin: “. . . a program is 'elegant,' by which I mean that it's the smallest possible program for producing the output that it does"
Chaitin prefaces his definition with: "I'll show you can't prove that a program is 'elegant'"—such a proof would solve the Halting problem (ibid).

Algorithm versus function computable by an algorithm: For a given function multiple algorithms may exist. This is true, even without expanding the available instruction set available to the programmer. Rogers observes that "It is . . . important to distinguish between the notion of algorithm, i.e. procedure and the notion of function computable by algorithm, i.e. mapping yielded by procedure. The same function may have several different algorithms".

Unfortunately there may be a trade-off between goodness (speed) and elegance (compactness)—an elegant program may take more steps to complete a computation than one less elegant. An example that uses Euclid's algorithm appears below.

Developing a good algorithm to solve the problem was not easy. We have developed an algorithm that will order the tuples and also evaluate the result.

There are basically 4 parts of the algorithm:

- Initialization
- Delay
- Ordering the tuple
- Total execution time and accuracy

Details of these are discussed below:

6.1 Initialization:

```
DECLARE INTEGER stream
DECLARE INTEGER tuple
DECLARE INTEGER t_number[stream][tuple]
DECLARE INTEGER temp_arr[stream][tuple]
DECLARE INTEGER temp, q=0,ch
DECLARE FLOAT iat=0.1, accuracy, t_delay, In_time=0, t_time=0, delay[tuple
```

t_number[stream][tuple] is a double array for the incoming tuples of data.
temp_arr[stream][tuple] is a temporary array that will store data in the time of ordering.

6.2 Delay:

```
PRINT "randomly generated delays:\n“
FOR i=0 TO i LESS THAN tuple
delay[i]=(double)(rand()%55)/100
PRINT "for tuple %d: %.2f\n",i+1,delay[i]
END
NEXT i
PRINT ”\n”
```

In the delay part we basically generate random delays for the tuples using a **for loop** and **random** function. And next we are printing the delays generated.

6.3 Ordering the tuples:

```
FOR i=0 TO i LESS THAN stream
weight=0
FOR j=tuple-1 TO j GREATER THAN 0
IF t_number[i][j]<t_number[i][j-1]
        THEN weight=1
END
END IF
PRIVIOUS j
IF weight
PRINT "tuples of stream %d are out-of-order\n",i+1
s_no=i+1
FOR k=0 TO k LESS THAN tuple-1
FOR j=0 TO j LESS THAN tuple-1
IF t_number[i][j] > t_number[i][j+1]
THEN
 temp=t_number[i][j+1]
 t_number[i][j+1]=t_number[i][j]
 t_number[i][j]=temp
END
END IF
NEXT j
END
NEXT k
FOR i=0 TO i LESS THAN stream
FOR j=0 TO j LESS THAN tuple
IF   t_number[i][j]!=temp_arr[i][j]
THEN NEXT q
END
END IF
NEXT j
END
NEXT i
ELSE PRINT "stream %d is in-time\n",i+1
END
END IF
```

Ordering the tuples is the crucial part of our algorithm. Here comparing the **tuple number** using **for loop** we are assigning **weight** values to the ordered and unordered tuples. We are also printing how many tuples are out of order and how many are synchronized.

6.4 Total execution time and accuracy:

```
PRINT "\n"
   PRINT "%d tuples of stream %d are out-of-order\n",q,s_no
   PRINT "%d tuples of both streams are synchronized\n",tuple-q
    PRINT ("\n")
    a=tuple-q
    accuracy=(double)a/(double)tuple *100
    PRINT "\n"
FOR j=0 TO j LESS THAN tuple
in_time = (double)tuple * iat
END
NEXT j
PRINT "total processing time for in-time stream  = %.2f\n",in_time
PRINT "Accuracy: %.2f%%  (no delay is considered)\n ",accuracy
IF s_no=1
THEN
FOR j=0 TO j LESS THAN tuple
t_time= (double)tuple * (double)iat
END
NEXT j
t_delay=0
PRINT "If consecutive delays are considered:\n"
FOR j=0 TO j LESS THAN q
t_time+=(double)delay[j]
 t_delay+=(double)delay[j]
 accuracy=(double)++a/(double)tuple *100
PRINT "Accuracy: %.2f%% \t processing time: %.2f (Delay: %.2f)\n",accuracy,t_time,delay[j]
END
NEXT j
PRINT total processing time for out-of-order stream = %.2f \n",t_time
 PRINT "Accuracy: %.2f%% (total delay= %.2f )\n",accuracy,t_delay
 t_time=0
END
END IF
```

In this part of the algorithm some basic **printing** part was done. Here we have calculated the **processing time** and the **accuracy** of the tuples both in total and individual cases.

Here is the full code given below:

Source code:

In computing, source code is any collection of computer instructions (possibly with comments) written using some human-readable computer language, usually as text. The source code of a program is specially designed to facilitate the work of computer programmers, who specify the actions to be performed by a computer mostly by writing source code. The source code is often transformed by a compiler program into low-level machine code understood by the computer. The machine code might then be stored for execution at a later time. Alternatively, an interpreter can be used to analyze and perform the outcomes of the source code program directly on the fly.

Most computer applications are distributed in a form that includes executable files, but not their source code. If the source code were included, it would be useful to a user, programmer, or system administrator, who may wish to modify the program or understand how it works.

Aside from its machine-readable forms, source code also appears in books and other media; often in the form of small code snippets, but occasionally complete code bases; a well-known case is the source code of PGP.

The notion of source code may also be taken more broadly, to include machine code and notations in graphical languages, neither of which are textual in nature. An example from an article presented on the annual IEEE conference and on Source Code Analysis and Manipulation.

For the purpose of clarity ‘source code’ is taken to mean any fully executable description of a software system. It is therefore so construed as to include machine code, very high level languages and executable graphical representations of systems.

```
#include<stdio.h>
int main()
{
   int stream=2;
   int tuple=100;
   int t_number[stream][tuple];
   int temp_arr[stream][tuple];
   float iat=0.1;
   float accuracy,t_delay,in_time=0,t_time=0;
   int i,j,k,num,weight,s_no,a;
   float delay[10];
   int temp,q=0;

//input
////////////////////////////////////////////////////////////////////////////////////////////////////////////
   printf("Enter the number of tuples in each stream:");
   scanf("%d",&num);

   for(i=0;i<stream;i++)
   {
      printf("Enter the tuple order for stream %d: ",i+1);
      for(j=0;j<num;j++)
      {
         scanf("%d",&t_number[i][j]);
      }
   }
////////////////////////////////////////////////////////////////////////////////////////////////////////////



//copy to a temp array and print the input
////////////////////////////////////////////////////////////////////////////////////////////////////////////
   for(i=0;i<stream;i++)
   {
      for(j=0;j<num;j++)
      {
         temp_arr[i][j]= t_number[i][j];
      }

    printf("\n");
```

```
     printf("Inserted values are:\n");

        for(i=0;i<stream;i++)
        {
           printf("stream %d : ",i+1);
           for(j=0;j<num;j++)
              {
                 printf("%d ",t_number[i][j]);
              }
           printf("\n");
        }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////


        printf("\n");
        printf("\n");


//delay
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

    printf("randomly generated delays:\n");
    for ( i=0; i<num; i++)
    {
       delay[i]=(double)(rand()%55)/100;
       printf("for tuple %d: %.2f\n",i+1,delay[i]); //random delay
    }
    printf("\n");


//ordering the delayed tuple
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
    for(i=0;i<stream;i++)
    {
       weight=0;
       for(j=num-1;j>0;j--)
       {
          if (t_number[i][j]<t_number[i][j-1])
          {
             weight=1;
```

```
      }
   }

   if (weight)
   {
      printf("tuples of stream %d are out-of-order\n",i+1); s_no=i+1;
      for(k=0;k<num-1;k++)
      {
         for(j=0;j<num-1;j++)
         {
            if   (t_number[i][j] > t_number[i][j+1])
            {
               temp=t_number[i][j+1];
               t_number[i][j+1]=t_number[i][j];
               t_number[i][j]=temp;
            }
         }
      }

      for(i=0;i<stream;i++)
      {
         for(j=0;j<num;j++)
         {
           if (t_number[i][j]!=temp_arr[i][j]) q++;
         }
      }
   }

      else printf("stream %d is in-time\n",i+1);

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////



      printf("\n");
      printf("\n");
```

```
//print after ordering
////////////////////////////////////////////////////////////////////////////////////////////////////////////////

/*    printf("values after ordering:\n");
   for(i=0;i<2;i++)
   {
      printf("stream %d : ",i+1);
      for(j=0;j<num;j++)
      {
         printf("%d ",t_number[i][j]);
      }
      printf("\n");
   }*/
   printf("\n");
   printf("%d tuples of stream %d are out-of-order\n",q,s_no);
   printf("%d tuples of both streams are synchronized\n",num-q);
   printf("\n");
   a=num-q;
   accuracy=(double)a/(double)num *100;
   printf("\n");
      for(j=0;j<num;j++)
      {
         in_time = (double)num * iat;
      }

        printf("total processing time for in-time stream  = %.2f\n",in_time);
        printf("Accuracy: %.2f%%  (no delay is considered)\n ",accuracy);



////////////////////////////////////////////////////////////////////////////////////////////////////////////////
      printf("\n");
      printf("\n");


//total execution time
////////////////////////////////////////////////////////////////////////////////////////////////////////////////
   if (s_no=1)
   {
      for(j=0;j<num;j++)
```

```
        {
           t_time= (double)num * (double)iat;
        }

        t_delay=0;
        printf("If consecutive delays are considered:\n");

        for(j=0;j<q;j++)
        {
           t_time+=(double)delay[j];
           t_delay+=(double)delay[j];
           accuracy=(double)++a/(double)num *100;
           printf("Accuracy: %.2f%% \t processing time: %.2f
(Delay: %.2f)\n",accuracy,t_time,delay[j]);
        }

        printf("\n");
        printf("\n");


        printf("total processing time for out-of-order stream = %.2f \n",t_time);
        printf("Accuracy: %.2f%% (total delay= %.2f )\n",accuracy,t_delay);

        t_time=0;

        printf("\n");
        printf("\n");
    }
////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
}
```

Chapter 7

RESULT ANALYSIS & PERFORMANCE EVALUATION

Evaluation is a systematic determination of a subject's merit, worth and significance, using criteria governed by a set of standards. It can assist an organization, program, project or any other intervention or initiative to assess any aim, realizable concept/proposal, or any alternative, to help in decision-making; or to ascertain the degree of achievement or value in regard to the aim and objectives and results of any such action that has been completed. The primary purpose of evaluation, in addition to gaining insight into prior or existing initiatives, is to enable reflection and assist in the identification of future change.

Evaluation is often used to characterize and appraise subjects of interest in a wide range of human enterprises, including the arts, criminal justice, foundations, non-profit organizations, government, health care, and other human services.

Evaluation is the structured interpretation and giving of meaning to predict or actual impacts of proposals or results. It looks at original objectives, and at what is either predicted or what was accomplished and how it was accomplished. So evaluation can be formative that is taking place during the development of a concept or proposal, project or organization, with the intention of improving the value or effectiveness of the proposal, project, or organization. It can also be assumptive, drawing lessons from a completed action or project or an organization at a later point in time or circumstance.

Evaluation is inherently a theoretically informed approach (whether explicitly or not), and consequently any particular definition of evaluation would have be tailored to its context – the theory, needs, purpose, and methodology of the evaluation process itself. Having said this, evaluation has been defined as:

A systematic, rigorous, and meticulous application of scientific methods to assess the design, implementation, improvement, or outcomes of a program. It is a resource-intensive process, frequently requiring resources, such as, evaluate expertise, labor, time, and a sizable budget

"The critical assessment, in as objective a manner as possible, of the degree to which a service or its component parts fulfills stated goals" (St Leger and

Wordsworth-Bell). The focus of this definition is on attaining objective knowledge, and scientifically or quantitatively measuring predetermined and external concepts.

"A study designed to assist some audience to assess an object's merit and worth" (Shuffleboard). In this definition the focus is on facts as well as value laden judgments of the programs outcomes and worth.

Depending on the topic of interest, there are professional groups that review the quality and rigor of evaluation processes.

Evaluating programs and projects, regarding their value and impact within the context they are implemented, can be ethically challenging. Evaluators may encounter complex, culturally specific systems resistant to external evaluation. Furthermore, the project organization or other stakeholders may be invested in a particular evaluation outcome. Finally, evaluators themselves may encounter "conflict of interest (COI)" issues, or experience interference or pressure to present findings that support a particular assessment.

General professional codes of conduct, as determined by the employing organization, usually cover three broad aspects of behavioral standards, and include inter-collegial relations (such as respect for diversity and privacy), operational issues (due competence, documentation accuracy and appropriate use of resources), and conflicts of interest (nepotism, accepting gifts and other kinds of favoritism). However, specific guidelines particular to the evaluator's role that can be utilized in the management of unique ethical challenges are required. The Joint Committee on Standards for Educational Evaluation has developed standards for program, personnel, and student evaluation. The Joint Committee standards are broken into four sections: Utility, Feasibility, Propriety, and Accuracy. Various European institutions have also prepared their own standards, more or less related to those produced by the Joint Committee. They provide guidelines about basing value judgments on systematic inquiry, evaluator competence and integrity, respect for people, and regard for the general and public welfare.

The American Evaluation Association has created a set of Guiding Principles for evaluators. The order of these principles does not imply priority among them; priority will vary by situation and evaluator role. The principles run as follows:

7.1 Systematic Inquiry: evaluators conduct systematic, data-based inquiries about whatever is being evaluated. This requires quality data collection, including a defensible choice of indicators, which lends credibility to findings. Findings are credible when they are demonstrably evidence-based, reliable and valid. This also pertains to the choice of methodology employed, such that it is consistent with the

aims of the evaluation and provides dependable data. Furthermore, utility of findings is critical such that the information obtained by evaluation is comprehensive and timely, and thus serves to provide maximal benefit and use to stakeholders.

7.2 Competence: evaluators provide competent performance to stakeholders. This requires that evaluation teams comprise an appropriate combination of competencies, such that varied and appropriate expertise is available for the evaluation process, and that evaluators work within their scope of capability.
Integrity/Honesty: evaluators ensure the honesty and integrity of the entire evaluation process. A key element of this principle is freedom from bias in evaluation and this is underscored by three principles: impartiality, independence, and transparency.

Independence is attained through ensuring independence of judgment is upheld such that evaluation conclusions are not influenced or pressured by another party, and avoidance of conflict of interest, such that the evaluator does not have a stake in a particular conclusion. Conflict of interest is at issue particularly where funding of evaluations is provided by particular bodies with a stake in conclusions of the evaluation, and this is seen as potentially compromising the independence of the evaluator. Whilst it is acknowledged that evaluators may be familiar with agencies or projects that they are required to evaluate, independence requires that they not have been involved in the planning or implementation of the project. A declaration of interest should be made where any benefits or association with project are stated. Independence of judgment is required to be maintained against any pressures brought to bear on evaluators, for example, by project funders wishing to modify evaluations such that the project appears more effective than findings can verify.

Impartiality pertains to findings being a fair and thorough assessment of strengths and weaknesses of a project or program. This requires taking due input from all stakeholders involved and findings presented without bias and with a transparent, proportionate, and persuasive link between findings and recommendations. Thus evaluators are required to delimit their findings to evidence. A mechanism to ensure impartiality is external and internal review. Such review is required of significant (determined in terms of cost or sensitivity) evaluations. The review is based on quality of work and the degree to which a demonstrable link is provided between findings and recommendations.

Transparency requires that stakeholders are aware of the reason for the evaluation, the criteria by which evaluation occurs and the purposes to which the findings will be applied. Access to the evaluation document should be facilitated

through findings being easily readable, with clear explanations of evaluation methodologies, approaches, sources of information, and costs incurred.

7.3 Respect for People: Evaluators respect the security, dignity and self-worth of the respondents, program participants, clients, and other stakeholders with whom they interact. This is particularly pertinent with regards to those who will be impacted upon by the evaluation findings. Protection of people includes ensuring informed consent from those involved in the evaluation, upholding confidentiality, and ensuring that the identity of those who may provide sensitive information towards the program evaluation is protected. Evaluators are ethically required to respect the customs and beliefs of those who are impacted upon by the evaluation or program activities. Examples of how such respect is demonstrated is through respecting local customs e.g. dress codes, respecting people's privacy, and minimizing demands on others' time. Where stakeholders wish to place objections to evaluation findings, such a process should be facilitated through the local office of the evaluation organization, and procedures for lodging complaints or queries should be accessible and clear.

Responsibilities for General and Public Welfare: Evaluators articulate and take into account the diversity of interests and values that may be related to the general and public welfare. Access to evaluation documents by the wider public should be facilitated such that discussion and feedback is enabled.

We have considered three different cases of unsorted tuples and compares the results for the three cases.

7.4 Case study:

An average, or typical, case is often not the richest in information. In clarifying lines of history and causation it is more useful to select subjects that offer an interesting, unusual or particularly revealing set of circumstances. A case selection that is based on representativeness will seldom be able to produce these kinds of insights. When selecting a subject for a case study, researchers will therefore use information-oriented sampling, as opposed to random sampling. Outlier cases (that is, those which are extreme, deviant or atypical) reveal more information than the potentially representative case. Alternatively, a case may be selected as a key case, chosen because of the inherent interest of the case or the circumstances surrounding it. Or it may be chosen because of researchers' in-depth local knowledge; where researchers have this local knowledge they are in a position to "soak and poke" as Fenno puts it,

and thereby to offer reasoned lines of explanation based on this rich knowledge of setting and circumstances.

Three types of cases may thus be distinguished:

- Key cases
- Outlier cases
- Local knowledge cases

Whatever the frame of reference for the choice of the subject of the case study (key, outlier, local knowledge), there is a distinction to be made between the subjestorical unity through which the theoretical focus of the study is being viewed. The object is that theoretical focus – the analytical frame. Thus, for example, if a researcher were interested in US resistance to communist expansion as a theoretical focus, then the Korean War might be taken to be the subject, the lens, the case study through which the theoretical focus, the object, could be viewed and explicated.

Beyond decisions about case selection and the subject and object of the study, decisions need to be made about purpose, approach and process in the case study. Thomas thus proposes a typology for the case study wherein purposes are first identified (evaluative or exploratory), then approaches are delineated (theory-testing, theory-building or illustrative), then processes are decided upon, with a principal choice being between whether the study is to be single or multiple, and choices also about whether the study is to be retrospective, snapshot or diachronic, and whether it is nested, parallel or sequential. It is thus possible to take many routes through this typology, with, for example, an exploratory, theory-building, multiple, nested study, or an evaluative, theory-testing, single, retrospective study. The typology thus offers many permutations for case study structure.

A closely related study in medicine is the case report, which identifies a specific case as treated and/or examined by the authors as presented in a novel form. These are, to a differentiable degree, similar to the case study in that many contain reviews of the relevant literature of the topic discussed in the thorough examination of an array of cases published to fit the criterion of the report being presented. These case reports can be thought of as brief case studies with a principal discussion of the new, presented case at hand that presents a novel interest.

7.4.1 Case 1:

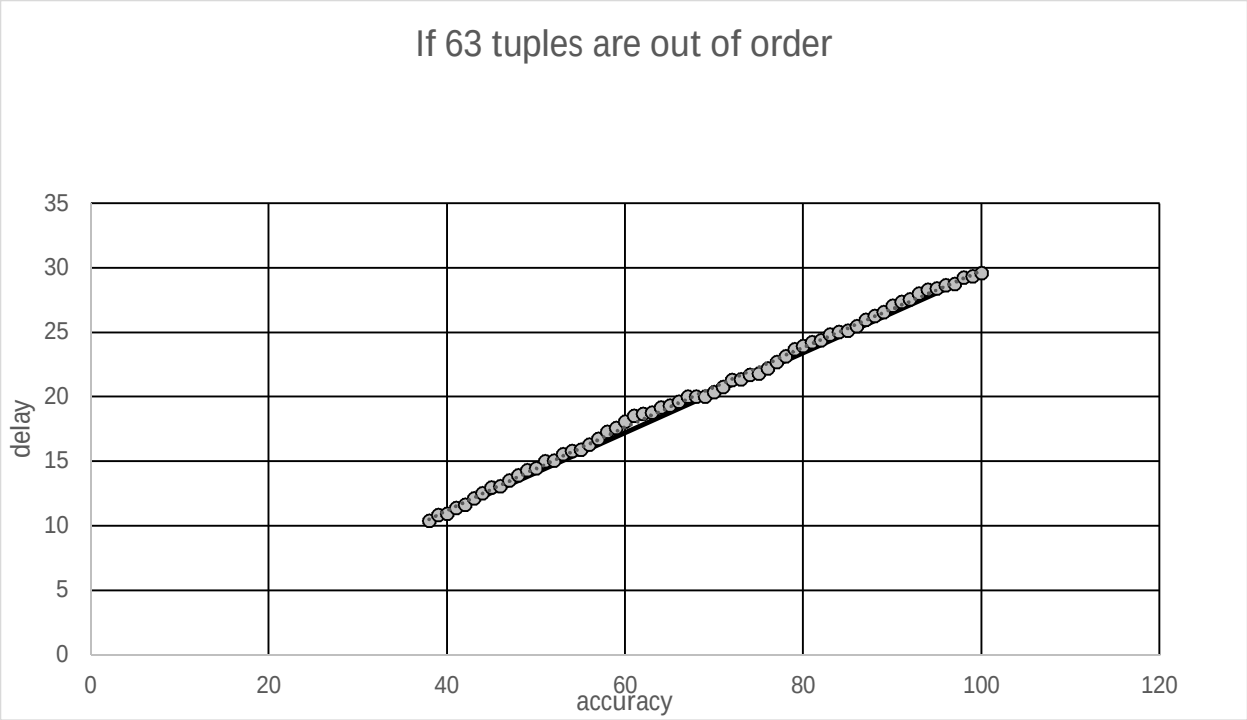
If 63 tuples are out of order:

- Accuracy is between 38% to 100%
- Processing time is between 10.41 sec to 29.61 sec
- Total processing time is 29.61 Sec
- Total delay for 100% accuracy 19.61sec

In the screen shots below we can see the detail for unsorted tuples.

```
If consecutive delays are considered:
Accuracy: 38.00%          processing time: 10.41 (Delay: 0.41)
Accuracy: 39.00%          processing time: 10.83 (Delay: 0.42)
Accuracy: 40.00%          processing time: 10.92 (Delay: 0.09)
Accuracy: 41.00%          processing time: 11.37 (Delay: 0.45)
Accuracy: 42.00%          processing time: 11.66 (Delay: 0.29)
Accuracy: 43.00%          processing time: 12.15 (Delay: 0.49)
Accuracy: 44.00%          processing time: 12.53 (Delay: 0.38)
Accuracy: 45.00%          processing time: 12.96 (Delay: 0.43)
Accuracy: 46.00%          processing time: 13.08 (Delay: 0.12)
Accuracy: 47.00%          processing time: 13.52 (Delay: 0.44)
Accuracy: 48.00%          processing time: 13.92 (Delay: 0.40)
Accuracy: 49.00%          processing time: 14.32 (Delay: 0.40)
Accuracy: 50.00%          processing time: 14.48 (Delay: 0.16)
Accuracy: 51.00%          processing time: 15.00 (Delay: 0.52)
Accuracy: 52.00%          processing time: 15.06 (Delay: 0.06)
Accuracy: 53.00%          processing time: 15.57 (Delay: 0.51)
Accuracy: 54.00%          processing time: 15.82 (Delay: 0.25)
Accuracy: 55.00%          processing time: 15.89 (Delay: 0.07)
Accuracy: 56.00%          processing time: 16.31 (Delay: 0.42)
Accuracy: 57.00%          processing time: 16.77 (Delay: 0.46)
Accuracy: 58.00%          processing time: 17.28 (Delay: 0.51)
Accuracy: 59.00%          processing time: 17.57 (Delay: 0.29)
Accuracy: 60.00%          processing time: 18.09 (Delay: 0.52)
Accuracy: 61.00%          processing time: 18.52 (Delay: 0.43)
Accuracy: 62.00%          processing time: 18.69 (Delay: 0.17)
Accuracy: 63.00%          processing time: 18.76 (Delay: 0.07)
Accuracy: 64.00%          processing time: 19.17 (Delay: 0.41)
Accuracy: 65.00%          processing time: 19.33 (Delay: 0.16)
Accuracy: 66.00%          processing time: 19.61 (Delay: 0.28)
Accuracy: 67.00%          processing time: 20.01 (Delay: 0.40)
Accuracy: 68.00%          processing time: 20.03 (Delay: 0.02)
Accuracy: 69.00%          processing time: 20.04 (Delay: 0.01)
Accuracy: 70.00%          processing time: 20.35 (Delay: 0.31)
Accuracy: 71.00%          processing time: 20.78 (Delay: 0.43)
Accuracy: 72.00%          processing time: 21.32 (Delay: 0.54)
Accuracy: 73.00%          processing time: 21.34 (Delay: 0.02)
Accuracy: 74.00%          processing time: 21.71 (Delay: 0.37)
Accuracy: 75.00%          processing time: 21.80 (Delay: 0.09)
Accuracy: 76.00%          processing time: 22.20 (Delay: 0.40)
```


Accuracy: 77.00% processing time: 22.69 (Delay: 0.49)
Accuracy: 78.00% processing time: 23.17 (Delay: 0.48)
Accuracy: 79.00% processing time: 23.68 (Delay: 0.51)
Accuracy: 80.00% processing time: 23.95 (Delay: 0.27)
Accuracy: 81.00% processing time: 24.23 (Delay: 0.28)
Accuracy: 82.00% processing time: 24.41 (Delay: 0.18)
Accuracy: 83.00% processing time: 24.85 (Delay: 0.44)
Accuracy: 84.00% processing time: 25.01 (Delay: 0.16)
Accuracy: 85.00% processing time: 25.12 (Delay: 0.11)
Accuracy: 86.00% processing time: 25.50 (Delay: 0.38)
Accuracy: 87.00% processing time: 25.98 (Delay: 0.48)
Accuracy: 88.00% processing time: 26.25 (Delay: 0.27)
Accuracy: 89.00% processing time: 26.59 (Delay: 0.34)
Accuracy: 90.00% processing time: 27.06 (Delay: 0.47)
Accuracy: 91.00% processing time: 27.38 (Delay: 0.32)
Accuracy: 92.00% processing time: 27.55 (Delay: 0.17)
Accuracy: 93.00% processing time: 27.99 (Delay: 0.44)
Accuracy: 94.00% processing time: 28.32 (Delay: 0.33)
Accuracy: 95.00% processing time: 28.38 (Delay: 0.06)
Accuracy: 96.00% processing time: 28.67 (Delay: 0.29)
Accuracy: 97.00% processing time: 28.75 (Delay: 0.08)
Accuracy: 98.00% processing time: 29.26 (Delay: 0.51)
Accuracy: 99.00% processing time: 29.36 (Delay: 0.10)
Accuracy: 100.00% processing time: 29.61 (Delay: 0.25)
total processing time for out-of-order stream = 29.61
Accuracy: 100.00% (total delay= 19.61)



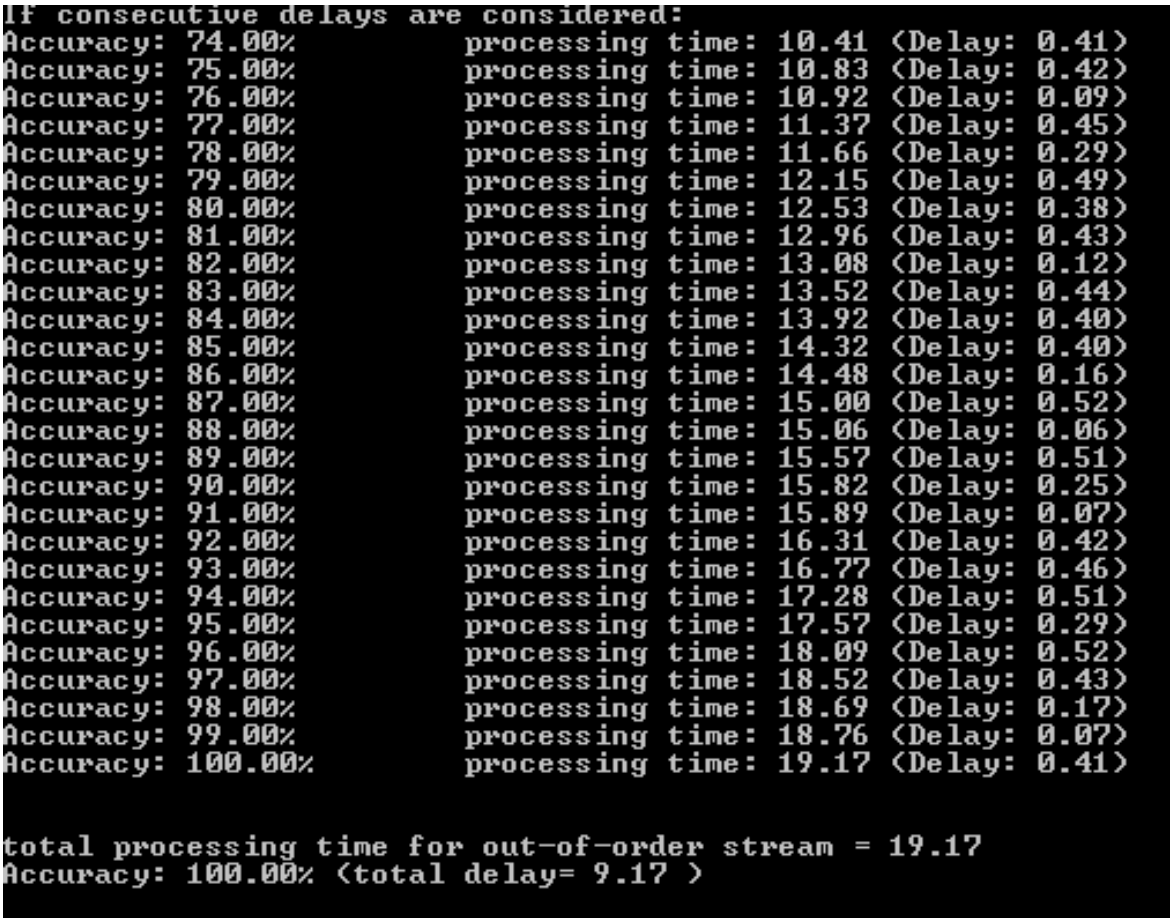
The graphs shows the accuracy in the x axis and delay in y axis. We can see the line is polynomial and with the delay accuracy is increasing.

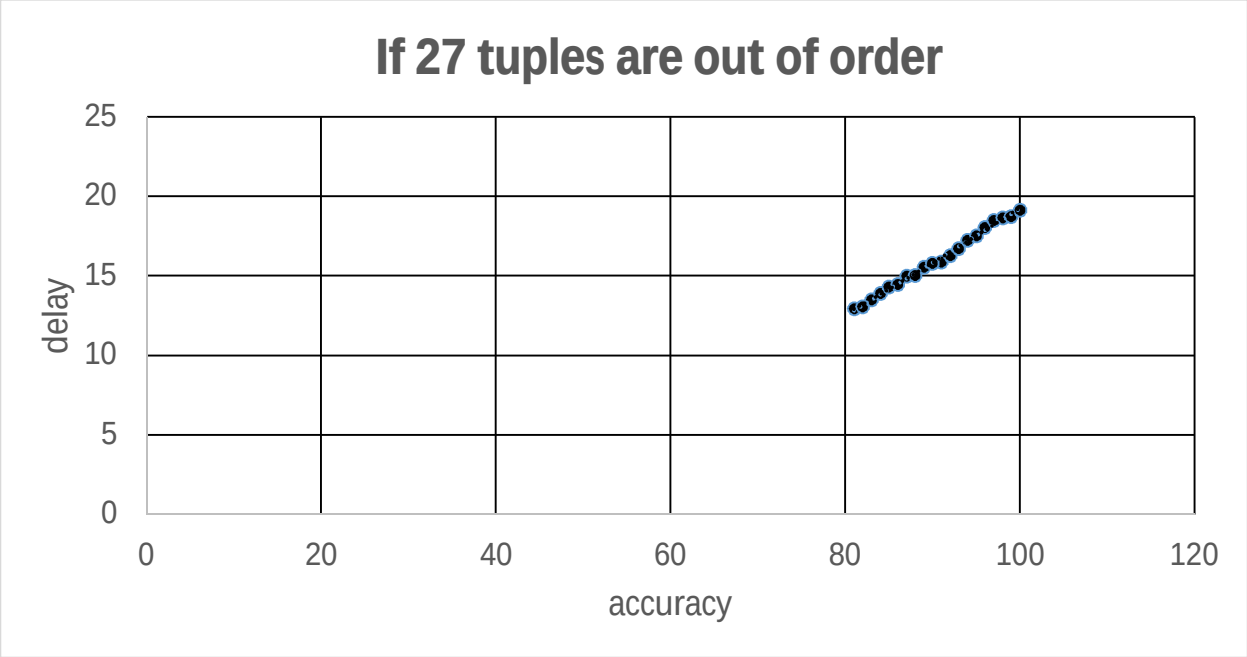
7.4.2 Case 2:

If 27 tuples are out of order:

- Accuracy is between 74% to 100%
- Processing time is between 10.41 sec to 19.17 sec
- Total processing time is 19.17 sec
- Total delay for 100% accuracy 9.17 sec

In the screen shots below we can see the detail for unsorted tuples.





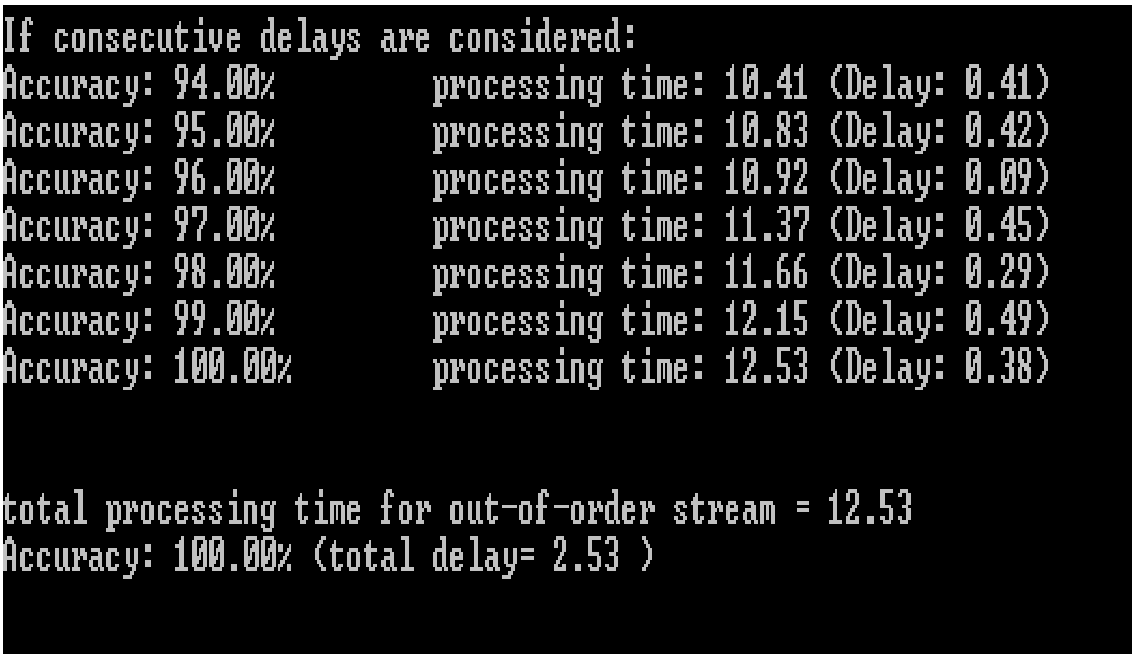
The graphs shows the accuracy in the x axis and delay in y axis. We can see the line is polynomial and with the delay accuracy is increasing.

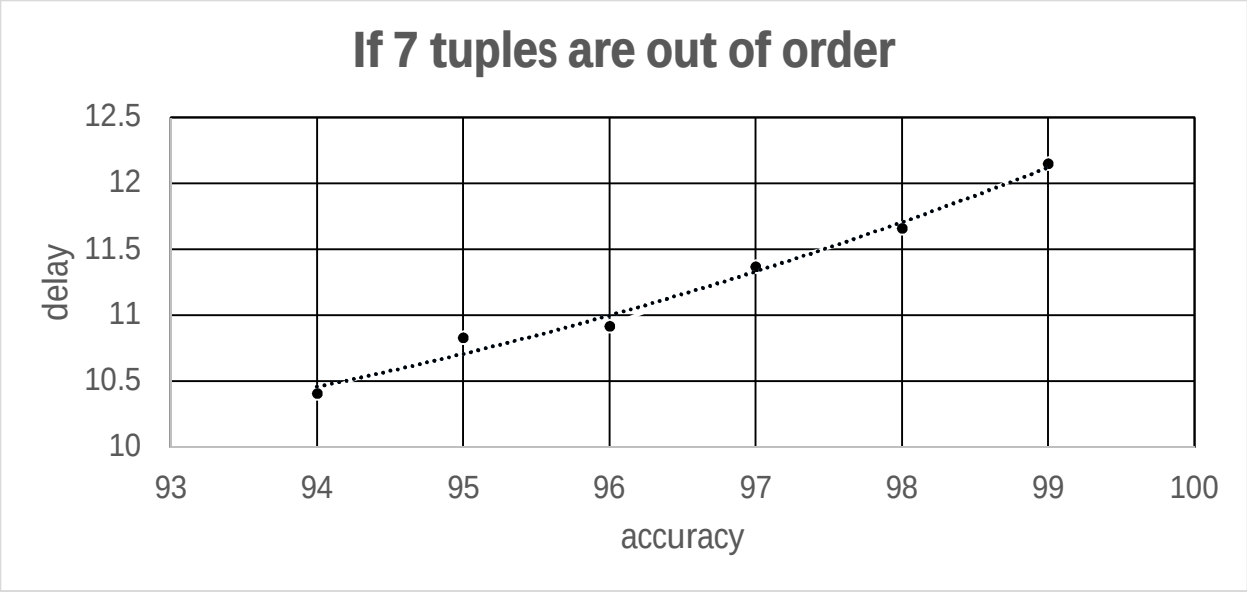
7.4.3 Case 3:

If 7 tuples are out of order:

- Accuracy is between 94% to 100%
- Processing time is between 10.41 sec to 12.53 sec
- Total processing time is 12.53 sec
- Total delay for 100% accuracy 2.53 sec

In the screen shots below we can see the detail for unsorted tuples.





The graphs shows the accuracy in the x axis and delay in y axis. We can see the line is polynomial and with the delay accuracy is increasing.

Chapter 8

CONCLUSION

The algorithm, by evaluating the sample data and graph, enables the user to choose the appropriate trade-off between delay and accuracy in any particular system.it depends on user choice as for different accuracy delay will be different. From the graphs user easily can see the variation of data and decide which he will choose. The main purpose of the algorithm is to enable the user to analyze the sample data that will be taken as the input and it will show the delay and the accuracy values. If the values are plotted in the graph, then the trade-off becomes easy to pick. As they have the properties similar to each other like if the processing time is increased then the accuracy also increases but just the opposite happens when the processing time is decreased then the accuracy also degrades. So this creates a constraint about picking the right point between them. Again, the delay and the accuracy has a completely similar relationship. When we want to increase the accuracy measure, the delay time also gets increased. So there is apparently no perfect solution but there can be an optimized point chosen depending on the system environment.

But in case of special situations (e.g. too many tuples within inter arrival time) the performance of the algorithm might degrade. So further improvements will surely been done after evaluating some user cases and more study. When the model gets fully functioning then based on the feedbacks and other bugs that will be revealed after the usage the improvisation shall be made.

REFERENCES AND BIBLIOGRAPHY

- [1] Moustafa A. Hammad, Walid G. Aref, Ann C. Catlin, Mohamed G. Elfeky and Ahmed K. Elmagarmid, “A Stream Database Server for Sensor Applications”.

- [2] Moustafa A. Hammad Walid G. ArefAhmed K. Elmagarmid, “Search based buffer management policies for streaming in continuous media servers”.

- [3] watsawee sansrimahachai , “tracing fine grained provenance in stream processing systems using a reverse mapping method”.

- [4] Irina Botan, Gustavo Alonso, Peter M. Fischer, Donald Kossmann, Nesime Tatbul ,”Flexible and Scalable Storage Management for Data-intensive Stream Processing”.

- [5] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, Jennifer Widom ,” Models and Issues in Data Stream Systems”