

Experiment No.: 05

Aim:

Data Stream Algorithms (any one): - Implementing DGIM algorithm using any Programming language.

Theory:

A data stream is an existing, continuous, ordered (implicitly by entrance time or explicitly by timestamp) chain of items. It is unfeasible to control the order in which units arrive, nor it is feasible to locally capture stream in its entirety. It is enormous volumes of data, items arrive at a high rate.

Data stream algorithms only have limited memory available but they may be able to defer action until a group of points arrive, while online algorithms are required to take action as soon as each point arrives.

In the data stream model, some or all of the input is represented as a finite sequence of integers (from some finite domain) which is generally not available for random access, but instead arrives one at a time in a "stream". If the stream has length n and the domain has size m , algorithms are generally constrained to use space that is logarithmic in m and n . They can generally make only some small constant number of passes over the stream, sometimes just one.

Types of Data Streams :

1. Data stream :

A data stream is a(possibly unchained) sequence of tuples. Each tuple comprised of a set of attributes, similar to a row in a database table

2. Transactional data stream :

It is a log interconnection between entities

- Credit card - purchases by consumers from producer.
- Telecommunications - phone calls by callers to the dialed parties.
- Web - accesses by clients of information at servers

3. Measurement data streams :

- Sensor Networks – a physical natural phenomenon, road traffic.
- IP Network – traffic at router interfaces.
- Earth climate – temperature, humidity level at weather stations.

Characteristics of Data Streams :

- ❖ Large volumes of continuous data, possibly infinite.
- ❖ Steady changing and requires a fast, real-time response.
- ❖ Data stream captures nicely our data processing needs of today.
- ❖ Random access is expensive and a single scan algorithm
- ❖ Store only the summary of the data seen so far.
- ❖ Maximum stream data are at a pretty low level or multidimensional in creation, needs multilevel and multidimensional treatment.

PROGRAM :-

```
import collections
```

```
class DGIM:
```

```
    def __init__(self, window_size):
```

```
        self.window_size = window_size
```

```
        self.buckets = collections.deque(maxlen=window_size)
```

```
    def add_bit(self, bit):
```

```
        self.buckets.append(int(bit))
```

```
    def query(self, k):
```

```
        count = 0
```

```
        for i, bit in enumerate(self.buckets):
```

```
            if bit == 1:
```

```
                count += 1
```

```
                if count == k:
```

```
                    return 2**(i+1)
```

```
        return None
```

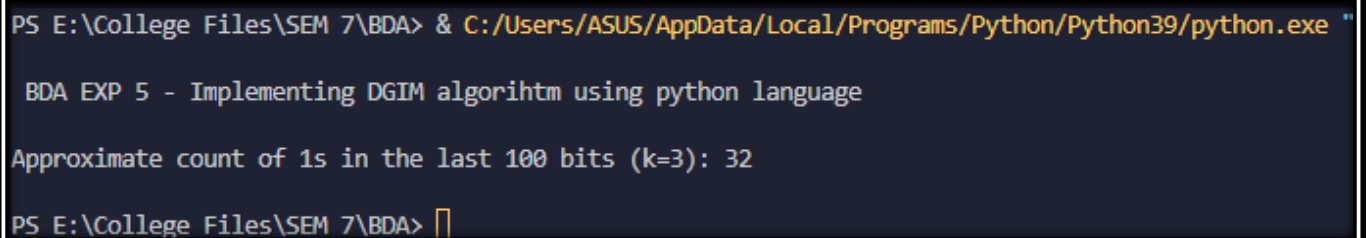
```
    def slide_window(self):
```

```
        self.buckets.popleft()
```

```
window_size = 100
```

```
dgim = DGIM(window_size)
binary_stream = "100110101011010010110101001001010101010110100100101011010"
for bit in binary_stream:
    dgim.add_bit(bit)
k = 3
approximate_count = dgim.query(k)
print("\n BDA EXP 5 - Implementing DGIM algorihtm using python language \n")
print(f"Approximate count of 1s in the last {window_size} bits (k={k}): {approximate_count}\n ")
```

OUTPUT :-



```
PS E:\College Files\SEM 7\BDA> & C:/Users/ASUS/AppData/Local/Programs/Python/Python39/python.exe "
BDA EXP 5 - Implementing DGIM algorihtm using python language
Approximate count of 1s in the last 100 bits (k=3): 32
PS E:\College Files\SEM 7\BDA> █
```

Conclusion:

We have successfully implemented DGIM algorithm using Python Programming