Assignment 1 : Graph Theory

Condensed Representation of Adjacency Matrix of a simple graph and transforming it into an Incidence Matrix

Jatin Goel • ICM2014503

Shubham Vashishtha • IHM2014007

Proposed Compressed Forms

- Adjacency List for sparse graph
- Decimal Representation
- Bit masking Based Representation for dense graph

Adjacency Matrix to Adjacency List

Pseudo Code :

```
For i=1 to V
{
     For j=i+1 to V
     {
          if(A[i][j])
          {
               Adj_List[i].add(j);
               Adj_List[j].add(i);
          }
     }
}
```

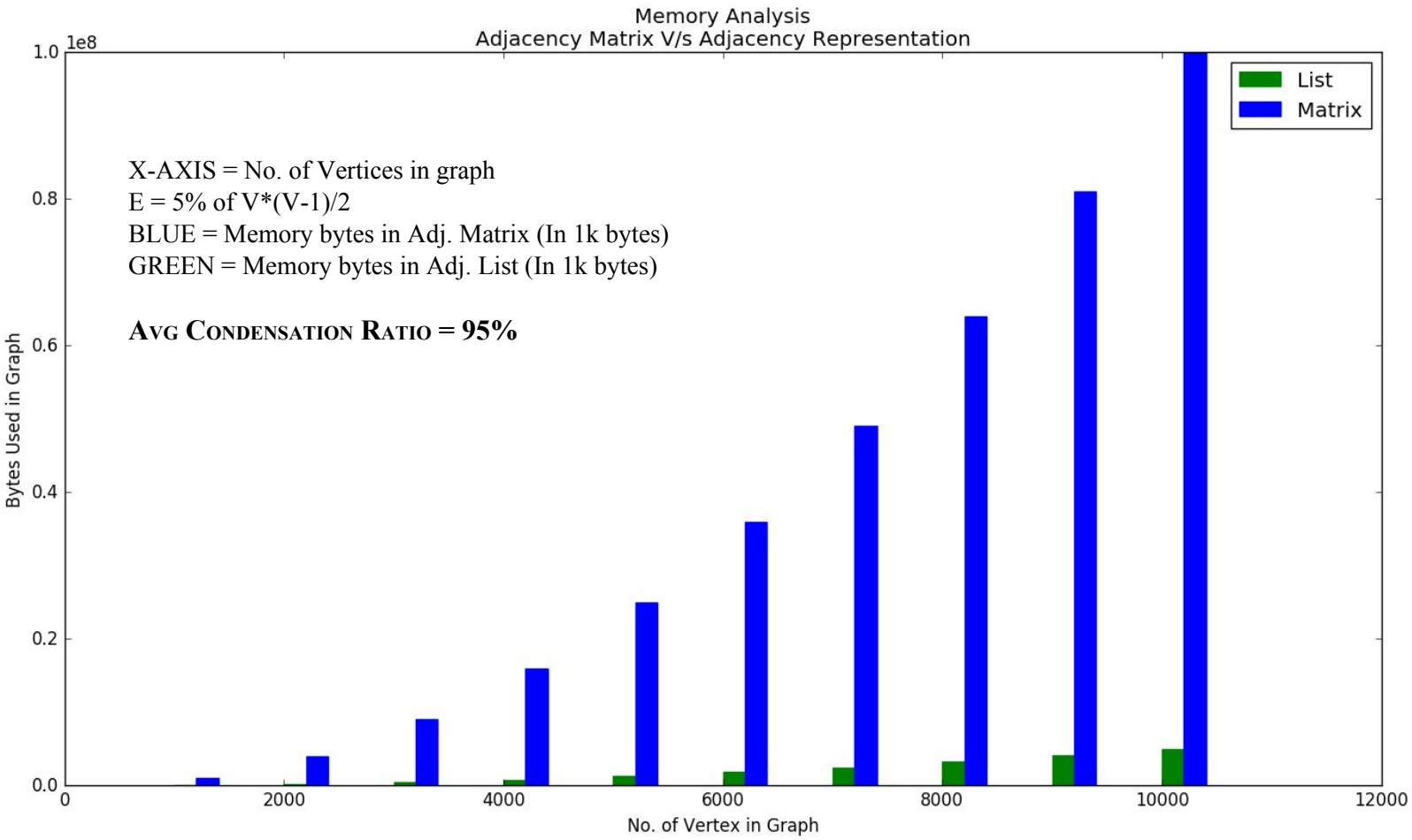
Adjacency Matrix to Adjacency List

Pseudo Code :

```
For i=1 to V
{
     For j=i+1 to V
     {
          if(A[i][j])
          {
               Adj_List[i].add(j);
               Adj_List[j].add(i);
          }
     }
}
```

Adjacency List to Incidence Matrix

```
Input  :  A[V]
V = No. of vertices in Graph
A[i] = Adjacency List of vertex i
Output : I = incidence matrix
I : initially an empty list
For i=1 to V
{
      For j=1 to deg(i)
      {
            if(i < A[i][j])
            {
                  edge : an array of size V
                  initialized by zero
                  edge[i] = 1
                  edge[A[i][j]] = 1
                  Append edge to I
            }
      }
}
return I
```


Memory Analysis
Adjacency Matrix V/s Adjacency Representation
1e8
1.0
0.8
0.6
0.4
0.2
0.0
Bytes Used in Graph
0
2000
4000
6000
8000
10000
12000
No. of Vertex in Graph
List
Matrix
X-AXIS = No. of Vertices in graph
E = 5% of V*(V-1)/2
BLUE = Memory bytes in Adj. Matrix (In 1k bytes)
GREEN = Memory bytes in Adj. List (In 1k bytes)
AVG CONDENSATION RATIO = 95%

Adjacency Matrix to Decimal Rep.

Pseudo Code :

```
For i=1 to V
{
     d = convert_decimal(i_row);
     Dec_Graph[i] = d;
}
```

Adjacency Matrix to Decimal Rep.

Pseudo Code :

```
For i=1 to V
{
      d = convert_decimal(i_row);
      Dec_Graph[i] = d;
}
```

Decimal Rep. to Incidence Matrix

```
Input  :  DEC_G[V]
V = No. of vertices in Graph
Output : I = incidence matrix
I : initially an empty list
bit_pos = V-1
For i=1 to V
{
      For j=i+1 to V
      {
            if(DEC_G[j] & (1<<bit_pos))
            {
                  edge : an array of size V
                  initialized by zero
                  edge[i] = 1
                  edge[j] = 1
                  Append edge to I
            }
      }
      bit_pos = bit_pos - 1
}
return I
```

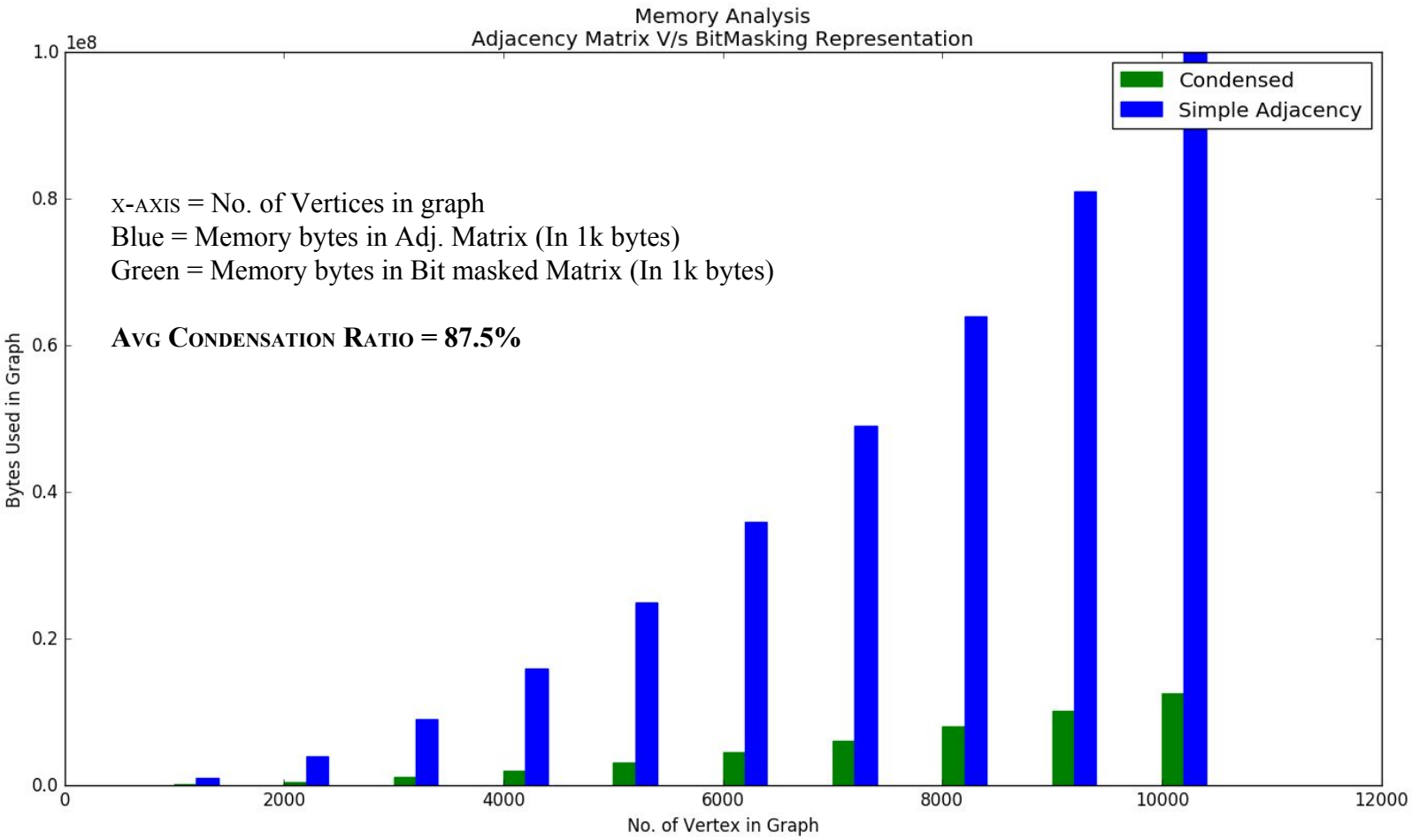
Adjacency Matrix to Bitmasking Rep.

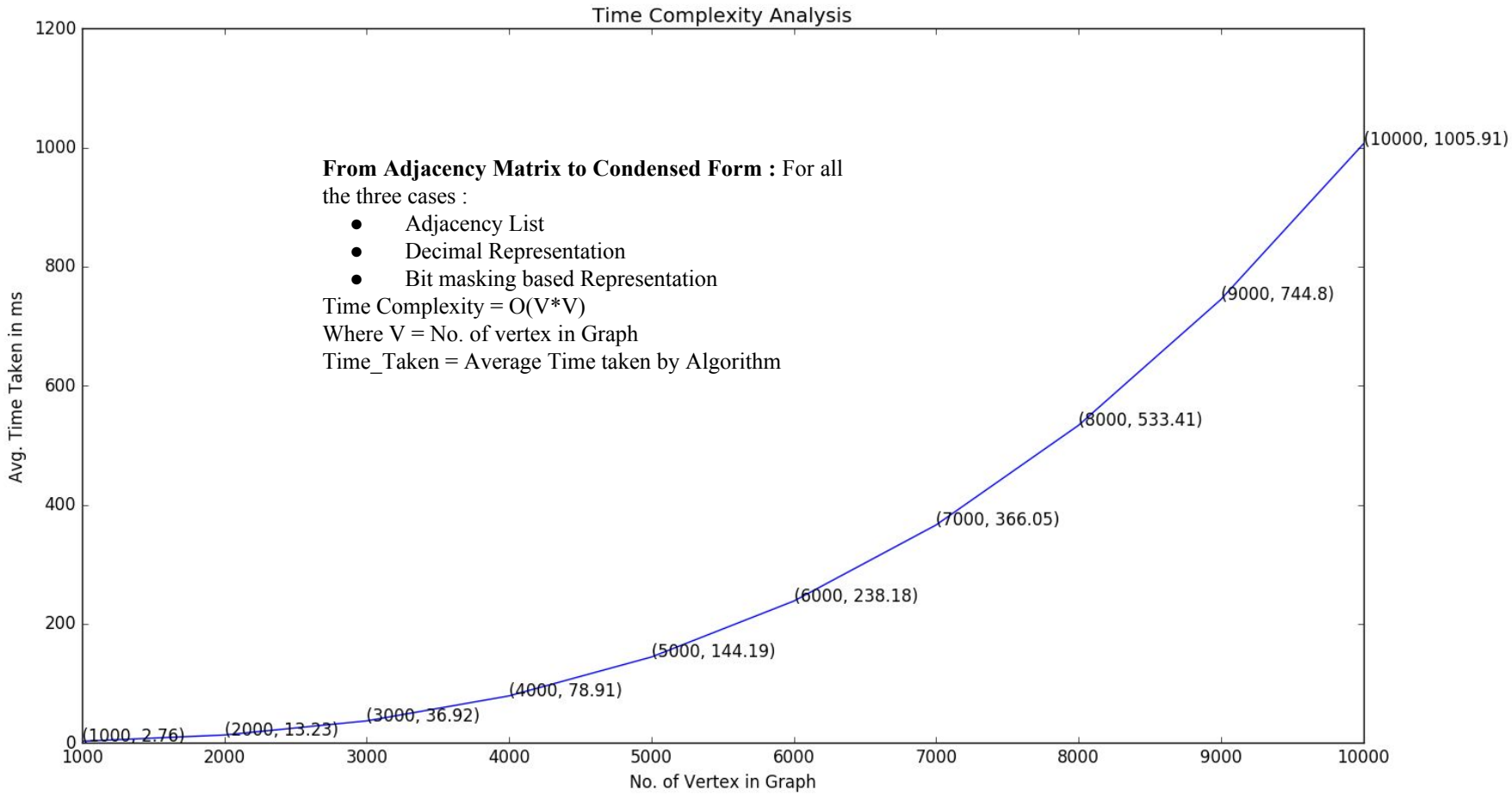
Pseudo Code :

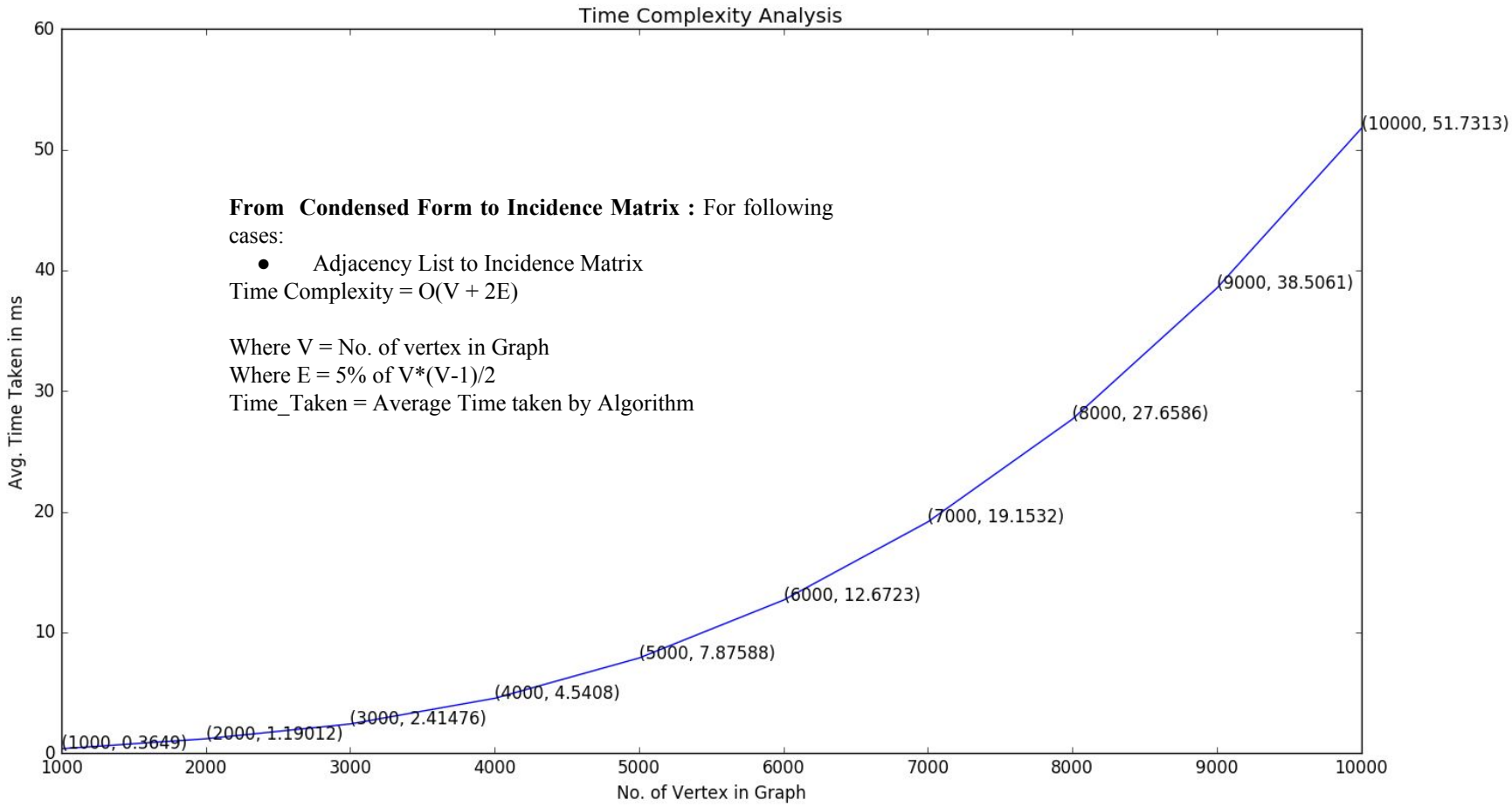
```
Input  :  A[V][V]
V = No. of vertices in Graph
k = Ceil(V/32);
Output : BMG[V][k]
For i=1 to V
{
      For j=i+1 to V
      {
            if(A[i][j])
            {
                  p = (j-1)/32 + 1;
                  q = (j-1)%32 + 1;
                  x = (i-1)/32 + 1;
                  y = (i-1)%32 + 1;
                  Set(BMG[i][p],q,true);
                  Set(BMG[j][x],y,true);

            }
      }
}
```

Bit Mask Rep. to Incidence Matrix

```
Input : BMG[V][k]
V = No. of vertices in Graph
k = Ceil(V/32);
Output : I[V][E]
I : an empty list
For i=1 to V {
      For j=i+1 to V {
            p = (j-1)/32 + 1;
            q = (j-1)%32 + 1;
            x = (i-1)/32 + 1;
            y = (i-1)%32 + 1;
            if(isSet(BMG[i][p], q)  {
                  edge : an array of size V
                  initialized by zero
                  edge[i] = 1
                  edge[j] = 1
                  Append edge to I
            }
      }
}
return (I)
```


Memory Analysis
Adjacency Matrix V/s BitMasking Representation
1e8
1.0
0.8
0.6
0.4
0.2
0.0
Bytes Used in Graph
0
2000
4000
6000
8000
10000
12000
No. of Vertex in Graph
Condensed
Simple Adjacency
x-axis = No. of Vertices in graph
Blue = Memory bytes in Adj. Matrix (In 1k bytes)
Green = Memory bytes in Bit masked Matrix (In 1k bytes)
Avg Condensation Ratio = 87.5%


Time Complexity Analysis
Avg. Time Taken in ms
No. of Vertex in Graph
0
200
400
600
800
1000
1200
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
(1000, 2.76)
(2000, 13.23)
(3000, 36.92)
(4000, 78.91)
(5000, 144.19)
(6000, 238.18)
(7000, 366.05)
(8000, 533.41)
(9000, 744.8)
(10000, 1005.91)
From Adjacency Matrix to Condensed Form : For all the three cases :
● Adjacency List
● Decimal Representation
● Bit masking based Representation
Time Complexity = O(V*V)
Where V = No. of vertex in Graph
Time_Taken = Average Time taken by Algorithm


Time Complexity Analysis
Avg. Time Taken in ms
No. of Vertex in Graph
0
10
20
30
40
50
60
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000
(1000, 0.3649)
(2000, 1.19012)
(3000, 2.41476)
(4000, 4.5408)
(5000, 7.87588)
(6000, 12.6723)
(7000, 19.1532)
(8000, 27.6586)
(9000, 38.5061)
(10000, 51.7313)
From Condensed Form to Incidence Matrix : For following cases:
● Adjacency List to Incidence Matrix
Time Complexity = O(V + 2E)
Where V = No. of vertex in Graph
Where E = 5% of V*(V-1)/2
Time_Taken = Average Time taken by Algorithm

Thank You