INDIAN INSTITUTE OF TECHNOLOGY ROPAR
Lab related to Fibonacci Heaps (Total 10 marks)

You can implement this in C/C++ (C++ preferred though)

Write a program for implementing Johnson's algorithm for All Pair Shortest Path where the algorithm's time complexity is analysed (considering the execution times on large graphs) considering different data structures based heaps implementation.

Bellman Ford Part: Consider simple implementation only and all edges to be considered strictly in lexicographic ordering (Mandatory component but no specific individual marks). This would be needed only in cases when input graph is directed graph. *(Presence of negative edge in an undirected connected graph implies that the shortest paths are not defined)*

Dijkstra's part: One need to implement it using different methods for heaps as follows:

Arrays (0.5 marks), binary heaps (1 mark), binomial heaps (1.5 marks), and Fibonacci heaps (5 marks)

Report (Max. 7 pages PDF document) analysing these four different heap data structures, using graphs/tables/plots. Analysis is done considering random + large graphs (with no negative cycle). Also, report must include your observations/discussion on results and the conclusions. (2 marks)

If one were to claim that he/she implemented and calling Fibonacci (or say, any specific) heap based method, then that has to be what is intended/claimed, and it should not be that actual execution is of binary (or say, some other) heap. Any means of manipulation will attract negative penalty. Name of the functions and the variables should be appropriate so that code can be easily understood. And reminding again of severe penalty for any plagiarism.

Refs (Related to measuring execution time of a program):
https://www.geeksforgeeks.org/how-to-measure-time-taken-by-a-program-in-c/
https://www.techiedelight.com/find-execution-time-c-program/
https://www.geeksforgeeks.org/measure-execution-time-with-high-precision-in-c-c/
https://stackoverflow.com/questions/5248915/execution-time-of-c-program

Your Code would be also considered for indentation, description of your idea and functions, efficiency (provided it is correct).

Input Format

First line: Number of Test Cases T and then follows their description
For each test case,
First line/row indicate N D (single space separated)

where
N is number of vertices in directed graph where vertices are labelled 1 to N (not 0 indexing)
D specifies whether Directed Graph or undirected graph (D=0 means undirected, else it is directed)

This is then followed by **N** lines each containing **N** integers (basically **N**x**N** Adjacency or say weight matrix representation of the graph of that test case).

The NxN matrix values / integers denote the edge-weights, there are no self-loops in the graph, and the weight=999999 indicates that there is no edge between those two vertices considered.
//Diagonal entries will always be zero and one can safely assume no edge from vertex to itself.

Output Format

For each test case,

If there is any negative weight cycle in graph, print just -1 as the shortest path is not well defined for atleast some source vertices.

Otherwise in output for that test case:
there are as many rows as many number of vertices in that graph under consideration (unless there is negative
Consider if the test case has N vertices, then N rows each containing N single space separated values s.t. the kth value in ith row tells the distance of kth vertex from the (source) vertex i. (Kind of NxN shortest path matrix)
Print 999999 if the vertex is not reachable from the source vertex considered.

Lastly,
There must be also one last row at end of outputs for T testcases, where last row has T entries indicative of time taken for different test cases that were considered.

Constraints (if any):
$0 \le T \le 2000$
$1 \le N \le 10$ million
$-999 \le$ Adj matrix values ≤ 999 (Ofcourse, no edge values are not indicated as **Inf** or $\propto$ but as 999999)

Note:
Whenever choices with equal considerations, prefer to consider vertices in lexicographic ordering.
Codes will be evaluated on Linux platform And execution command would be something like

```
      ./a.out 1
Or   ./a.out 2
Or   ./a.out 3
Or   ./a.out 4              OR    ./a.out
```

Where 1,2,3, and 4 indicates which specific data structure were to be used.
(1: Arrays based, 2: binary heaps, 3: binomial heaps and 4: Fibonacci heaps. Default choice: 4)

Deadline: 21st Dec. 2020 11:59:59 PM for program submission
22nd Dec. 2020 10 am (morning) for the report submission

Links for submission will be provided later by 19th Dec. Thanks

All the best !

Sample Input	Sample Output
6	
4 1	
0 999999 999999 2	0 -5 -2 2
6 0 3 999999	6 0 3 8
4 999999 0 5	4 -2 0 5
999999 -7 -3 0	-1 -7 -4 0
4 1	**0 -3 0 2**
0 -3 999999 2	**4 0 3 6**
5 0 3 999999	**1 -2 0 3**
1 999999 0 999999	**-1 -4 -1 0**
-1 999999 4 0	0 -5 -1 0
4 1	999999 0 4 5
0 -5 2 3	999999 999999 0 1
999999 0 4 999999	999999 999999 999999 0
999999 999999 0 1	**-1**
999999 999999 999999 0	-1
4 1	**0 2 4**
0 -4 999999 999999	**2 0 6**
3 0 999999 999999	**4 6 0**
999999 999999 0 2	0.001 **0.002** 0.002 **0.000001** 0.000001 0.0005
999999 999999 -1 0	
4 0	
0 1 2 3	
1 0 2 -1	
2 2 0 4	
3 -1 4 0	
3 0	
0 2 4	
2 0 999999	
4 999999 0	

//Last row here in output - that just shows some hypothetical values of execution time taken...