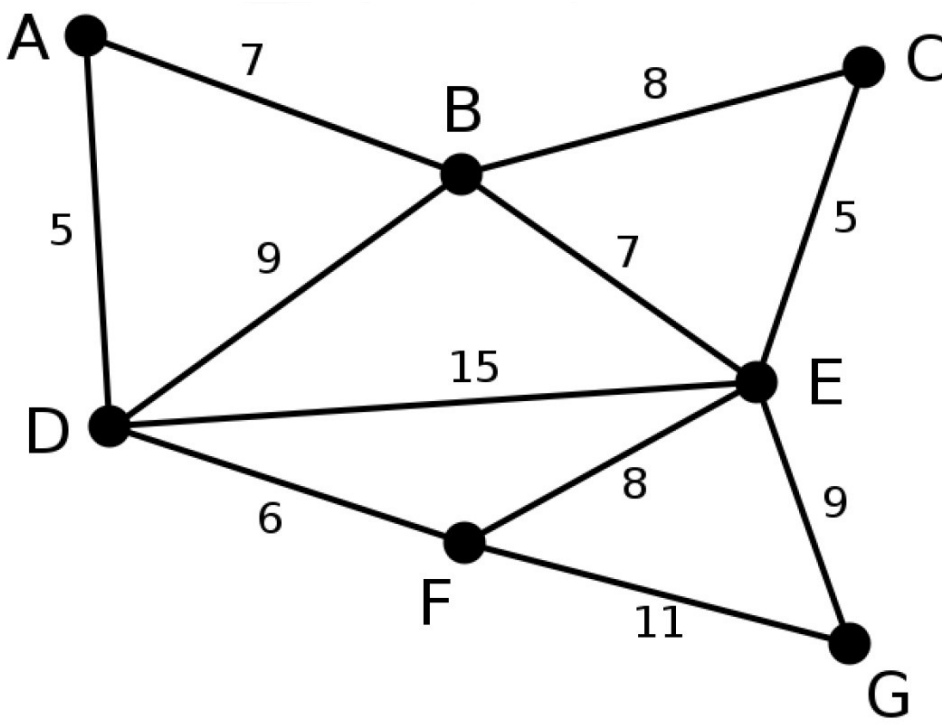
Q1 Bellman-Ford

10 Points

In this question you will explore Bellman-Ford Shortest Path algorithms. Consider the following weighted undirected graph with 7 vertices and 11 edges.



In Problem Set 6, you saw that changing the weight of edge EF from $+8$ to -8 led to Dijkstra's Algorithm producing incorrect outputs. In order to solve this problem, this weeks module discussed Bellman-Ford, technique. Now consider the following questions

Q1.1

5 Points

Let's convert the above graph to a $directed$ graph, where each edge is pointed in one direction -- i.e., xy means the edge goes from x to y. We will also make some of our edge weights negative.

Suppose our edges (and weights) are listed in this order: GE=+9, FG=+11, EF=+8, ED=$-$15, EC=+5, DF=+6, DB=+9, CB=-8, BE=+7, BA=-7, AD=+5.

Using the above order, apply the Bellman-Ford Algorithm to determine the shortest distance from the source vertex A to each of the other six vertices in the graph. Clearly show your steps.

From the question, we can get the directed edges are:
(A,D), (D,F), (D,B), (B,A), (B,E), (C,B), (F,G), (E,C), (E,F), (E,D), (G,E)
then, we just start to iterate the directed edges starting from vertex as follows:

Edges	(A,D)	(D,B)	(D,F)	(B,E)	(C,B)	(F,G)	(E,C)	(E,F)	(E,D)	(G,E)
1st	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞
2st	5	9	6							
3st	5	14	11							
4st	5	14	11	21						
5st	5	14	11	21	26					
6st	5	14	11	21	26	22				
7st	5	14	11	21	26	22	26	11	5	
8st.	5	14	11	21	26	22	26	11	5	21

So, we can have the final shortest distance from source vertex A to others is:

A->D 5
(A,B) = A-D-B = 14
(A,C)=A-D-B-C=26
(A,E)=A-D-B-E=21
(A,F)=A-D-F=11
(A,G)=A-D-F-G=22

No files uploaded

Q1.2
5 Points

Determine a precise Loop Invariant for the Bellman-Ford Algorithm, clearly stating your Initialization, Maintenance, and Termination statements. Prove that your loop invariant holds, clearly and carefully justifying each step in your proof.

Relax edges, progressively decreasing v.d until we get the actual shortest-path weight δ(s,v)

for the The Bellman-Ford algorithm, we have the following pseudocode:(See attached file)

Loop invariant:
We can have a BF[v] or whatever name is okay for this question, which is the length of v-t path and after we have i rounds, the value BF[v] is no larger than the length of shortest path v-t using <= i edges.

Initialization:
We just initialized a single source (G, s), and the distances from the source to all vertices as infinite and distance to the source itself as 0. So, the invariant is trivially true.

Maintenance:
The algorithm makes |V| -1passes over the edges of the graph. Each pass is one iteration of the for loop of lines 3–5 and consists of relaxing each edge of the graph once.

Termination:
After making |V| -1 passes, lines 6–9 check for a negative-weight cycle and return the appropriate boolean value.

Q1.2.txt Download

```
1  Bellman-Ford(G,w,s)
2      Initialize-single-source(G,s)
3      for i = 1 to |G.V|-1
4          for each edge(u,v) ∈ G.E
5              Relax(u,v,w)
6      for each edge(u,v) ∈ G.E
7          if v.d > u.d +w(u,v)
8              return False
9      return True
10
```

Q2 Bellman Ford - Programming

10 Points

Problem 787. Cheapest Flights Within K Stops is a Leetcode problem that can be solving using Bellman Ford's technique - Your work is get to the problem and provide a solution.
Please make a screen grab of your solution (from Leetcode) and upload it here. You code will also be needed

Q2.pdf Download

Q2.java Download

```java
class Solution {
    public int findCheapestPrice(int n, int[][] flights, int src, int dst, int k) {
        int[] price = new int[n];
        int[] priceNext = new int[n];

        Arrays.fill(price, 1 << 20);
        Arrays.fill(priceNext, 1<<20);
        price[src] = 0;
        priceNext[src] = 0;

```

```
11          for(int i = 0; i <= k; i++) {
12              for(int j = 0; j < flights.length; j++) {
13                  int from = flights[j][0];
14                  int to = flights[j][1];
15                  int cost = flights[j][2];
16
17                  if(price[from] + cost < priceNext[to])
    {
18                      priceNext[to] = price[from] +
    cost;
19                  }
20              }
21              System.arraycopy(priceNext, 0, price, 0,
    price.length);
22          }
23
24          if(price[dst] < (1 << 20)){
25              return price[dst];
26          }else{
27              return -1;
28          }
29      }
30  }
```

Q3 Floyd Warshall

10 Points

Problem 1334. Find the City With the Smallest Number of Neighbors at a Threshold Distance is a Leetcode problem that can be solving using Floyd Warshall algorithm. Your work is get to the problem and provide a solution.

Please make a screen grab of your solution (from Leetcode) and upload it here. You code will also be needed

▾ Q3.pdf Download

Q3.java Download

```
1   class Solution {
2       private static int INF= (int) Math.pow(10,4)+1;
3       public int findTheCity(int n, int[][] edges, int
    distanceThreshold) {
4           int[][] graph= new int[n][n];
5           int res=0, smallest= n;
6           for(int[] row: graph)
7               Arrays.fill(row, INF);
8
9           for(int[] edge: edges)
10              graph[edge[0]][edge[1]]= graph[edge[1]]
    [edge[0]]= edge[2];
```

```
11
12          for (int i = 0; i < n; ++i)
13              graph[i][i] = 0;
14
15          for(int k=0;k<n;k++){
16              for(int i=0;i<graph.length;i++){
17                  for(int j=0;j<graph[i].length;j++){
18                      if(graph[i][k]==INF || graph[k]
    [j]==INF) continue;
19                      graph[i][j]= Math.min(graph[i][j],
    graph[i][k]+graph[k][j]);
20                  }
21              }
22          }
23
24          for(int i=0;i<n;i++){
25              int count= 0;
26              for(int j=0;j<graph[i].length;j++)
27                  if(graph[i][j]<=distanceThreshold)
28                      count++;
29              if(count<=smallest){
30                  res= i;
31                  smallest= count;
32              }
33          }
34
35          return res;
36
37      }
38  }
```

Problem Set 9

GRADED

STUDENT
Kejian Tong

TOTAL POINTS
30 / 30 pts

QUESTION 1
Bellman-Ford **10** / 10 pts

1.1 (no title) **5** / 5 pts

1.2 (no title) **5** / 5 pts

QUESTION 2

Bellman Ford - Programming **10** / 10 pts

QUESTION 3

Floyd Warshall **10** / 10 pts