Fundamentals of Bellman-Ford algorithm

- Able to find the shortest distance from the source with a graph that has negative weights
- Considers all edges V-1 times to find shortest distance
- No shortest distance if there’s a negative cycle because it’ll loop forever
- Loop V-1 times as it’s the max amount of jumps without cycles in graphs to find shortest distance
- Last loop of loop V is to determine if there’s a negative cycle. If there is a difference between the array built in V-1’s iteration and V’s iteration, then there’s a negative cycle
- Time Complexity: O(VE), where V is number of vertices and E is number of edges
- Space Complexity: O(V), using the one column approach(always use this!)

2021 Sem 1 Question 17,18 and 19

Graph and Shortest Distance

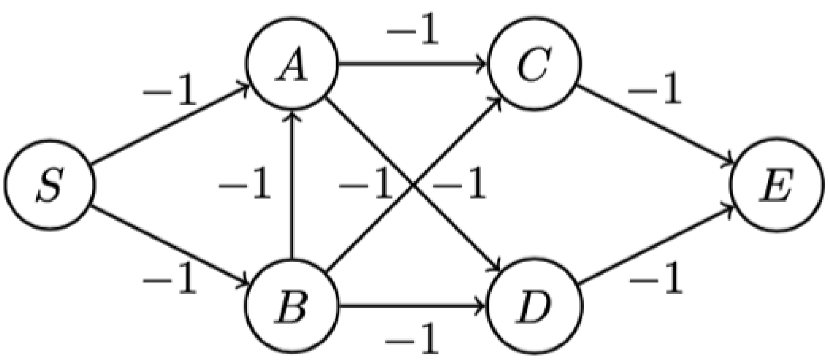
Question 17

Consider the following version of the **Bellman-Ford algorithm**:

Algorithm 59 Bellman-Ford

```
function BELLMAN_FORD(G = (V, E), s)
    dist[1..n] = ∞
    pred[1..n] = null
    dist[s] = 0
    for k = 1 to n − 1 do
        for each edge e in E do
            RELAX(e)
    return dist[1..n], pred[1..n]
```

and the following **directed graph**



Let S be the source node for the execution of the Bellman-Ford algorithm.

If the edges are relaxed in the following order (S,B), (A,C), (B,C), (S,A), (B,A), (C,E), (D,E), (A,D), (B,D).

What is the value of dist[C] **after the first iteration of the outer loop** is done?

Question 18

What is the value of dist[D] **after the first iteration of outer loop** of Bellman-Ford in the scenario above?

Question 19

What is the value of dist[E] **after the first iteration of outer loop** of Bellman-Ford in the scenario above?

After first iteration of the outer loop

Node	Shortest distance from source(S)
S	0
A	-2
B	-1
C	-2
D	-3
E	-3

2022 Sem 1 Question 10

Question 10

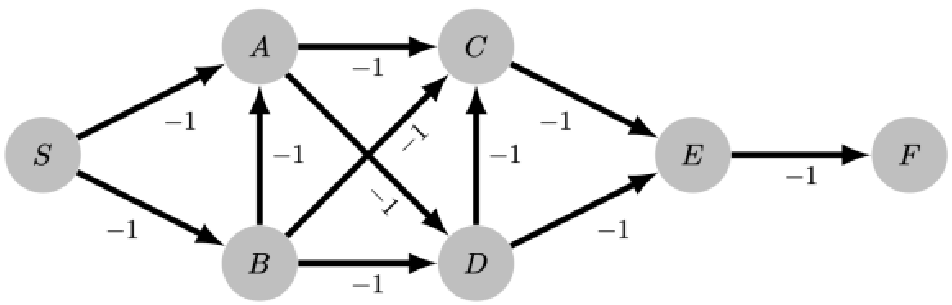
Consider the following version of the **Bellman-Ford algorithm**

3
Marks

Algorithm 54 Bellman-Ford

function BELLMAN_FORD($G = (V, E)$, s)
    $dist[1..n] = \infty$
    $pred[1..n] =$ **null**
    $dist[s] = 0$
    for $k = 1$ **to** $n - 1$ **do**
        for each edge e **in** E **do**
            RELAX(e)
    return $dist[1..n]$, $pred[1..n]$

and the following directed graph



Let S be the source node for the execution of the Bellman-Ford algorithm.

If the edges are relaxed in the following order (S,A), (S,B), (B,A), (B,D), (D,E), (A,D), (A,C), (D,C), (E,F), (B,C), (C,E), what is the **value of dist[E]+dist[F] after the first iteration of the outer loop is finished?**

After first iteration of the outer loop

Node	Shortest distance from source(S)
S	0
A	-2
B	-1
C	-4
D	-3
E	-5
F	-4

Answer is -9

Fundamentals of Floyd-Warshall algorithm

- Find **All-Pairs** shortest distance
- Builds on the transitive closure property: If A -> B and B -> C, then A -> C
- Able to detect negative cycles easily in a graph
- Time Complexity: $O(V^3)$, where V is the number of vertices
- Space Complexity: $O(V^2)$, where V is the number of vertices
- Faster than Dijkstra and Bellman-Ford for All-Pairs shortest distance because those algorithms need to be ran multiple times. Finding All-Pairs shortest distance for Dijkstra is $O(V^3 \log V)$ and for Bellman-Ford is $O(V^4)$.

2021 Sem 1 Question 20

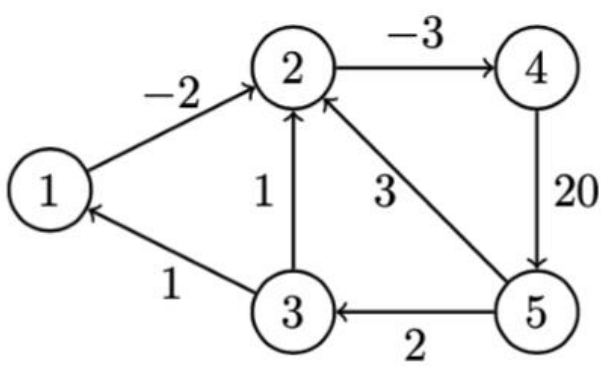
Question 20

Consider the **Floyd-Warshall algorithm**

Algorithm 63 Floyd-Warshall

```
function FLOYD_WARSHALL(G = (V, E))
    dist[1..n][1..n] = ∞
    dist[v][v] = 0 for all vertices v
    dist[u][v] = w(u, v) for all edges e = (u, v) in E
    for each vertex k = 1 to n do
        for each vertex u = 1 to n do
            for each vertex v = 1 to n do
                dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v])
    return dist[1..n][1..n]
```

and the following **directed graph**



After the outer loop of the algorithm **finished two iterations**, what is the sum of all values in the array *dist* **that are not equal to infinity?** Just type the numerical answer without punctuation or spaces.

Initial matrix

	1	2	3	4	5
1	0	-2	INF	INF	INF
2	INF	0	INF	-3	INF
3	1	1	0	INF	INF
4	INF	INF	INF	0	20
5	INF	3	2	INF	0

Processing when k = 1

This means that Node 1 is the intermediate node

1 -> 1 -> 1 = 0 + 0 = 0

1 -> 1 -> 2 = 0 + -2 = -2

3 -> 1 -> 1 = 1 + 0 = 1

3 -> 1 -> 2 = 1 + -2 = -1

After first outer loop

	1	2	3	4	5
1	0	-2	INF	INF	INF
2	INF	0	INF	-3	INF
3	1	-1	0	INF	INF
4	INF	INF	INF	0	20
5	INF	3	2	INF	0

Processing when k = 2

This means that Node 2 is the intermediate node

1 -> 2 -> 2 = -2 + 0 = -2

1 -> 2 -> 4 = -2 + -3 = -5

2 -> 2 -> 2 = 0 + 0 = 0

2 -> 2 -> 4 = 0 + -3 = -3

3 -> 2 -> 2 = -1 + 0 = -1

3 -> 2 -> 4 = -1 + -3 = -4

5 -> 2 -> 2 = 3 + 0 = 3

5 -> 2 -> 4 = 3 + -3 = 0

After second outer loop

	1	2	3	4	5
1	0	-2	INF	-5	INF
2	INF	0	INF	-3	INF
3	1	-1	0	-4	INF
4	INF	INF	INF	0	20
5	INF	3	2	0	0

Answer = 0 + -2 + -5 + 0 + -3 + 1 + -1 + 0 + -4 + 0 + 20 + 3 + 2 + 0 + 0

= 11

2022 Sem 1 Question 11

Question 11

Consider a run of the **Floyd-Warshall algorithm** on a **Directed Weighted Graph** *G*.

Algorithm 56 Floyd-Warshall

```
function FLOYD_WARSHALL(G = (V, E))
    dist[1..n][1..n] = ∞
    dist[v][v] = 0 for all vertices v
    dist[u][v] = w(u, v) for all edges e = (u, v) in E
    for each vertex k = 1 to n do
        for each vertex u = 1 to n do
            for each vertex v = 1 to n do
                dist[u][v] = min(dist[u][v], dist[u][k] + dist[k][v])
    return dist[1..n][1..n]
```

The run produced the following matrix:

	A	B	C	D	E	F	G	H
A	0	14	None	None	20	-9	20	-8
B	13	0	None	None	49	20	None	44
C	16	49	0	20	12	-6	-3	40
D	25	49	None	0	30	47	None	None
E	None	39	None	None	0	40	45	12
F	None	11	31	18	25	0	49	39
G	32	None	None	None	None	47	0	34
H	-10	7	22	None	None	15	21	0

Select the correct observation(s) that can be made from the matrix above.

Note: You **do not need to validate the correctness of the matrix** but just make observations based on the given matrix.

Select one or more:

a.
There is a negative cycle in the graph.

b.
There is a cycle including vertices *A* and *G*.

c.
There is an edge from vertex *B* to vertex *E* with a distance of 49.

d.
There is a path from vertex *C* to vertex *F* with a distance of -6.

Point A is false because the values at the diagonals in the matrix is not negative

Point B is correct because Matrix[A][G] is not None, and Matrix[G][A] is not None

Point C is false because it’s not necessarily an edge, because there can be many different paths from B going through many other vertices as intermediate nodes many times to then get to E with a distance of 49

Point D is correct

Answer is B and D