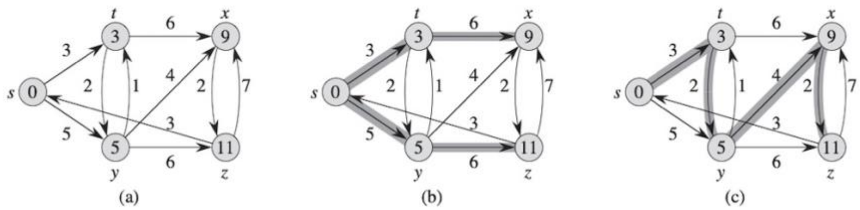
Hand_on_15

⇨ Implement and test on examples from the book. Then upload your source code to GitHub. Do this for the following algorithms:

1. Dijkstra's algorithm:

⇒ Run Dijkstra’s algorithm on the directed graph of Figure 24.2, first using vertex S as the source and then using vertex z as the source. In the style of Figure 24.6, show the d and π values and the vertices in set S after each iteration of the while loop.



(fig 24.2)

24.3 Dijkstra’s algorithm 659

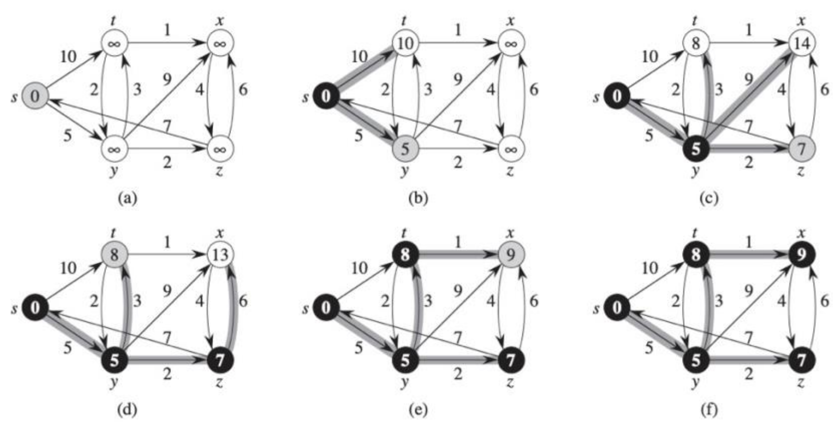


Figure 24.6 The execution of Dijkstra’s algorithm. The source *s* is the leftmost vertex. The

(fig 24.6)

⇨ Solution:

Here,

Here’s the step-by-step output of Dijkstra’s algorithm executed on the graph from Figure 24.2, modeled in the style of Figure 24.6. The output includes each iteration of the algorithm showing:

- The set S of vertices for which the shortest path is known (visited).
- The d values (shortest known distances from the source).
- The π values (predecessor of each vertex on the shortest path).

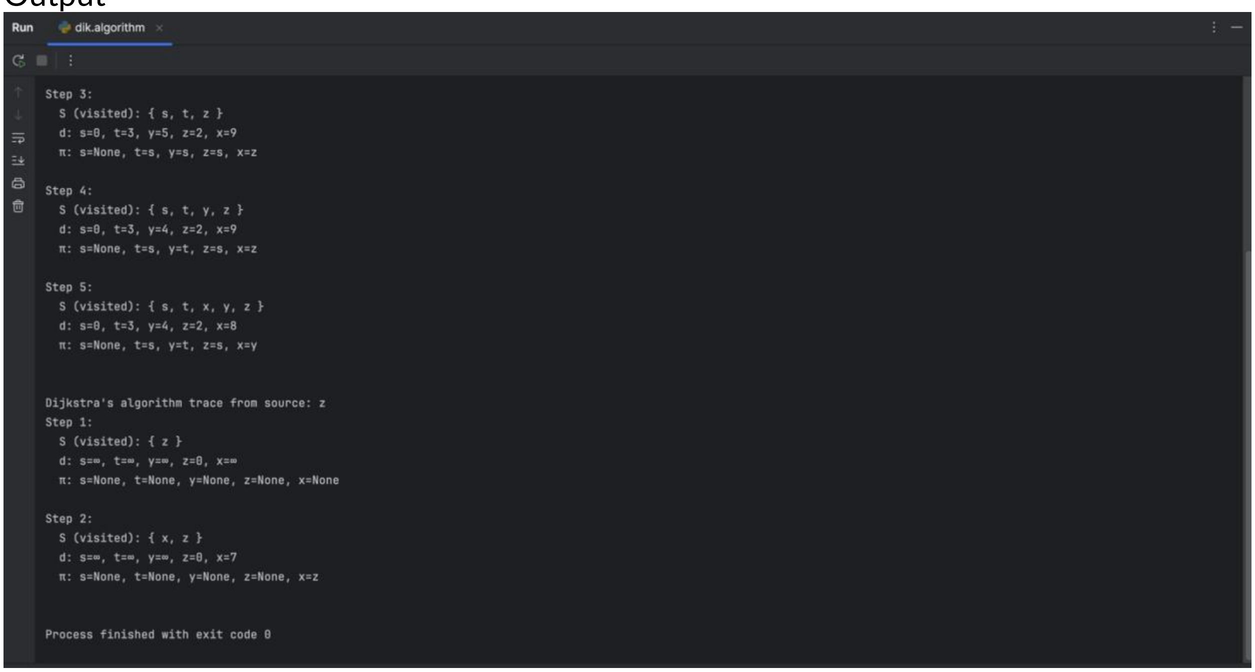
⇨ Source s:

Step	S (visited)	d-values	π-values
1	{s}	s=0, t=∞, y=∞, z=∞, x=∞	s=None
2	{s, z}	s=0, t=3, y=5, z=2, x=∞	t=s, y=s, z=s
3	{s, z, t}	s=0, t=3, y=5, z=2, x=9	x=z
4	{s, z, t, y}	s=0, t=3, y=4, z=2, x=9	y=t
5	All visited	s=0, t=3, y=4, z=2, x=8	x=y

⇨ Source: z

Step	S (visited)	d-values	π-values
1	{z}	z=0, x=∞, others=∞	z=None
2	{z, x}	z=0, x=7, others=∞	x=z

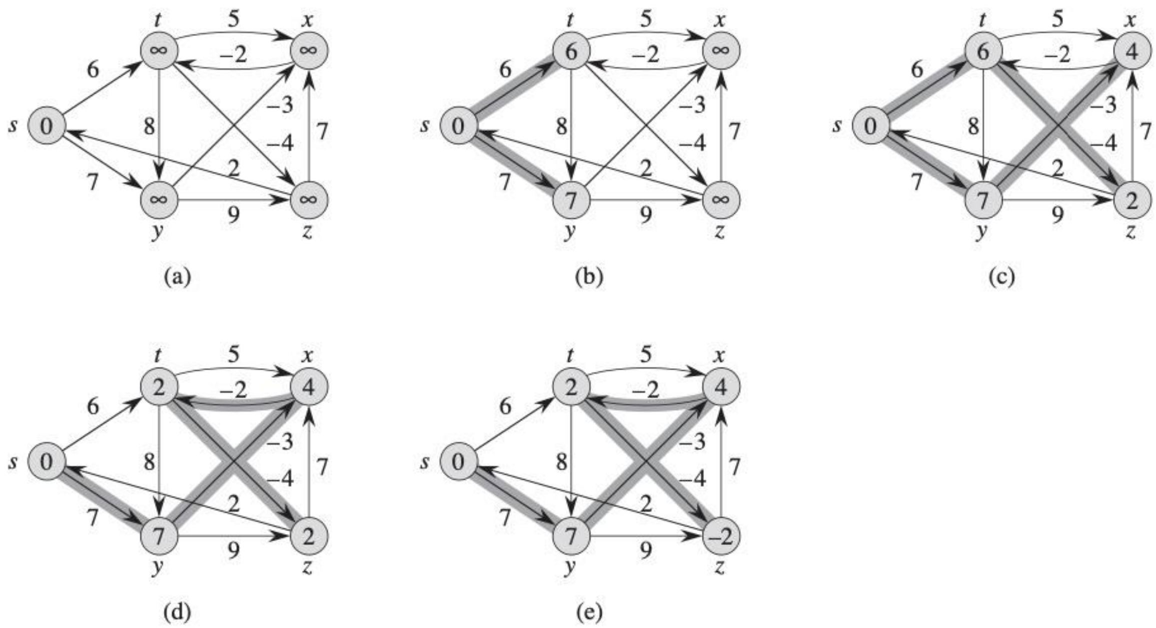
⇨ Output



```
Run   dik.algorithm

Step 3:
  S (visited): { s, t, z }
  d: s=0, t=3, y=5, z=2, x=9
  π: s=None, t=s, y=s, z=s, x=z

Step 4:
  S (visited): { s, t, y, z }
  d: s=0, t=3, y=4, z=2, x=9
  π: s=None, t=s, y=t, z=s, x=z

Step 5:
  S (visited): { s, t, x, y, z }
  d: s=0, t=3, y=4, z=2, x=8
  π: s=None, t=s, y=t, z=s, x=y


Dijkstra's algorithm trace from source: z
Step 1:
  S (visited): { z }
  d: s=∞, t=∞, y=∞, z=0, x=∞
  π: s=None, t=None, y=None, z=None, x=None

Step 2:
  S (visited): { x, z }
  d: s=∞, t=∞, y=∞, z=0, x=7
  π: s=None, t=None, y=None, z=None, x=z


Process finished with exit code 0
```

2. Bellman-Ford algorithm:

⇨ Run the Bellman-Ford algorithm on the directed graph of Figure 24.4, using vertex z as the source. In each pass, relax edges in the same order as in the figure, and show the d and π values after each pass. Now, change the weight of edge (z, x) to 4 and run the algorithm again, using s as the source.



⇨ Solution:
Here,
implementation of the Bellman-Ford algorithm is solved below; it includes two runs:

- Using vertex z as the source (with original weights).
- Changing weight of edge (z, x) to 4, and using vertex s as the source.

```
Bellman-Ford trace from source: z
Pass 1:
  d: s=2, t=∞, y=∞, x=7, z=0
  π: s=z, t=None, y=None, x=z, z=None

Pass 2:
  d: s=2, t=6, y=9, x=7, z=0
  π: s=z, t=y, y=s, x=z, z=None

Pass 3:
  d: s=2, t=6, y=9, x=7, z=0
  π: s=z, t=y, y=s, x=z, z=None

Pass 4:
  d: s=2, t=6, y=9, x=7, z=0
  π: s=z, t=y, y=s, x=z, z=None


Bellman-Ford trace from source: s
Pass 1:
  d: s=0, t=4, y=7, x=5, z=2
  π: s=None, t=y, y=s, x=y, z=t

Pass 2:
  d: s=0, t=4, y=7, x=4, z=0
  π: s=None, t=y, y=s, x=z, z=t

Pass 3:
  d: s=0, t=4, y=7, x=4, z=0
  π: s=None, t=y, y=s, x=z, z=t

Pass 4:
  d: s=0, t=4, y=7, x=4, z=0
  π: s=None, t=y, y=s, x=z, z=t
```

PythonProject > bellman_algo.py   67:1 LF UTF-8 4 spaces Python 3.13 (PythonProject)

---

3. Run the Floyd-Warshall algorithm on the weighted, directed graph of Figure 25.2. Show the matrix D^k that results for each iteration of the outer loop.

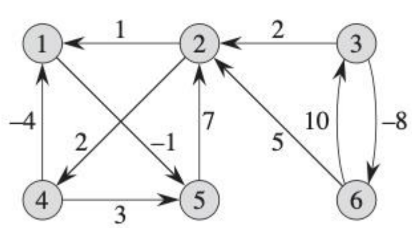


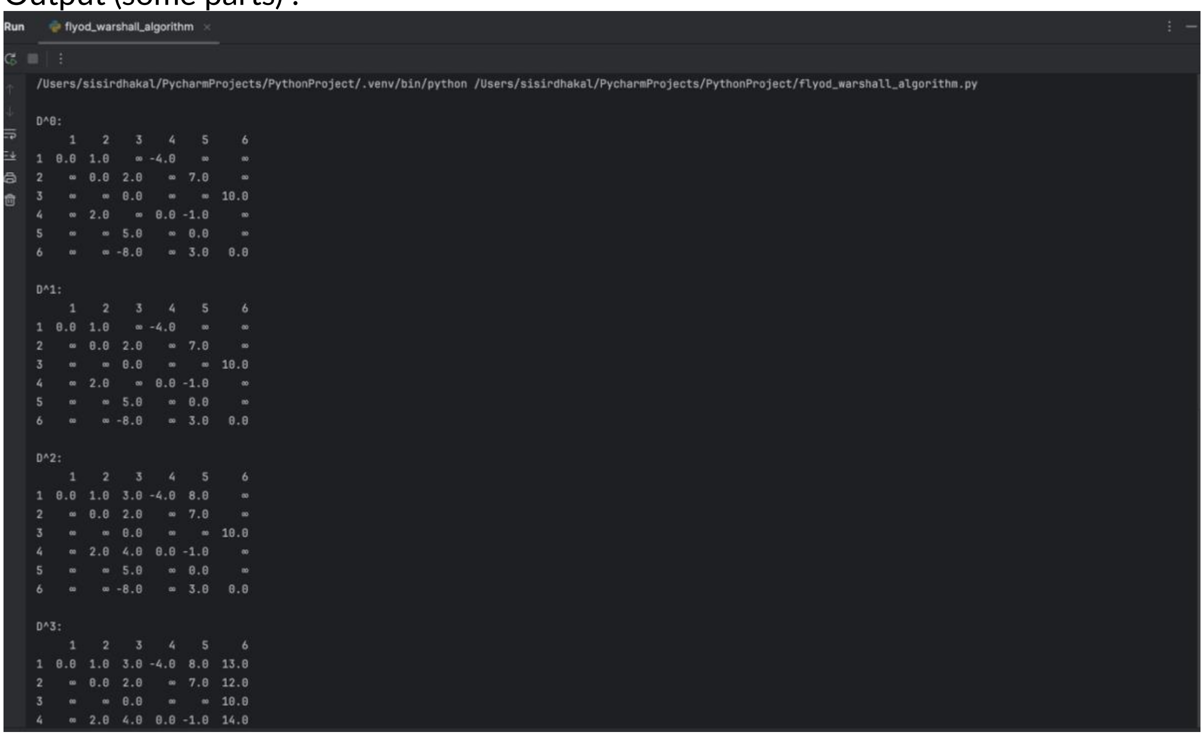
FIG: 25.2

⇨ Solution:
Here,
Let's implement the Floyd-Warshall algorithm in Python and generate the intermediate distance matrices D^K for each iteration of the outer loop:

- Edges with weights (directed):

$$(1 \rightarrow 2, weight = 1)$$
$$(1 \rightarrow 4, weight = -4)$$
$$(2 \rightarrow 3, weight = 2)$$
$$(2 \rightarrow 5, weight = 7)$$
$$(3 \rightarrow 6, weight = 10)$$
$$(4 \rightarrow 2, weight = 2)$$
$$(4 \rightarrow 5, weight = -1)$$
$$(5 \rightarrow 3, weight = 5)$$
$$(6 \rightarrow 3, weight = -8)$$
$$(6 \rightarrow 5, weight = 3)$$

- Output (some parts) :




```
/Users/sisirdhakal/PycharmProjects/PythonProject/.venv/bin/python /Users/sisirdhakal/PycharmProjects/PythonProject/flyod_warshall_algorithm.py

D^0:
     1    2    3    4    5     6
1  0.0  1.0    ∞ -4.0    ∞     ∞
2    ∞  0.0  2.0    ∞  7.0     ∞
3    ∞    ∞  0.0    ∞    ∞  10.0
4    ∞  2.0    ∞  0.0 -1.0     ∞
5    ∞    ∞  5.0    ∞  0.0     ∞
6    ∞    ∞ -8.0    ∞  3.0   0.0

D^1:
     1    2    3    4    5     6
1  0.0  1.0    ∞ -4.0    ∞     ∞
2    ∞  0.0  2.0    ∞  7.0     ∞
3    ∞    ∞  0.0    ∞    ∞  10.0
4    ∞  2.0    ∞  0.0 -1.0     ∞
5    ∞    ∞  5.0    ∞  0.0     ∞
6    ∞    ∞ -8.0    ∞  3.0   0.0

D^2:
     1    2    3    4    5     6
1  0.0  1.0  3.0 -4.0  8.0     ∞
2    ∞  0.0  2.0    ∞  7.0     ∞
3    ∞    ∞  0.0    ∞    ∞  10.0
4    ∞  2.0  4.0  0.0 -1.0     ∞
5    ∞    ∞  5.0    ∞  0.0     ∞
6    ∞    ∞ -8.0    ∞  3.0   0.0

D^3:
     1    2    3    4    5     6
1  0.0  1.0  3.0 -4.0  8.0  13.0
2    ∞  0.0  2.0    ∞  7.0  12.0
3    ∞    ∞  0.0    ∞    ∞  10.0
4    ∞  2.0  4.0  0.0 -1.0  14.0
```

Thank you.....!!!