GRAPHS

CS062

DATA STRUCTURES AND ADVANCED PROGRAMMING

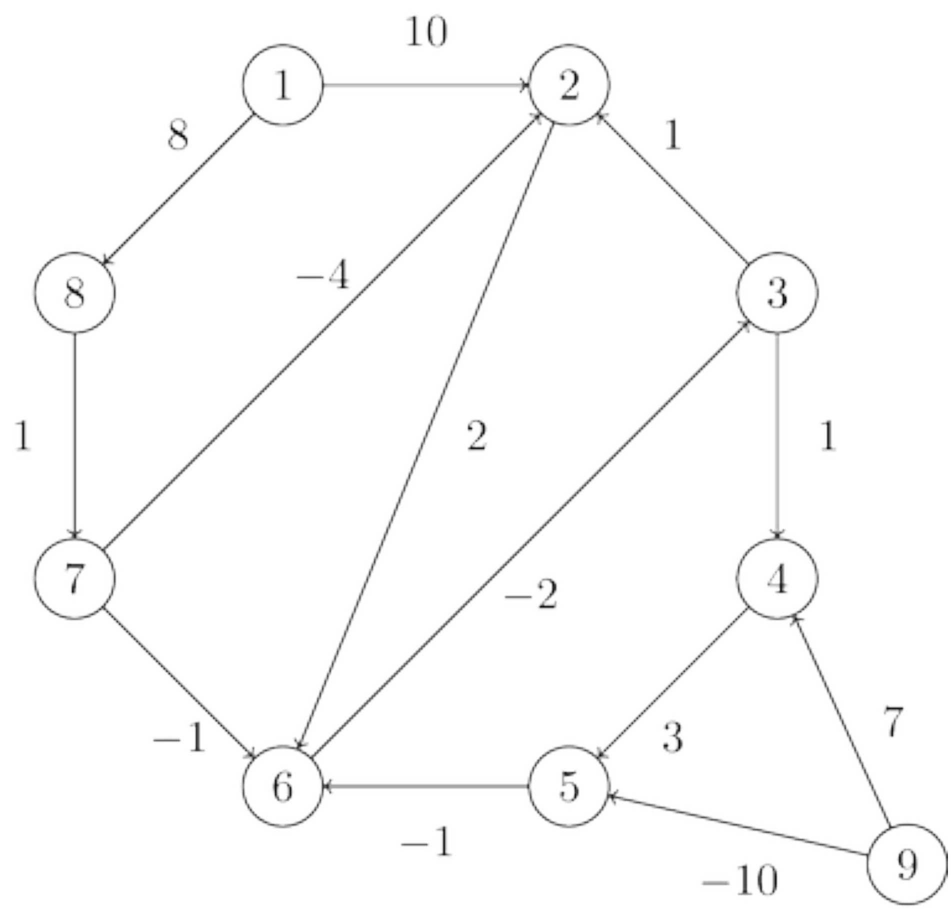
28: Minimum Spanning Trees

Tom Yeh
he/him/his

- Evaluations
- Bellman-Ford Algorithm
- Minimum Spanning Tree

Practice Time

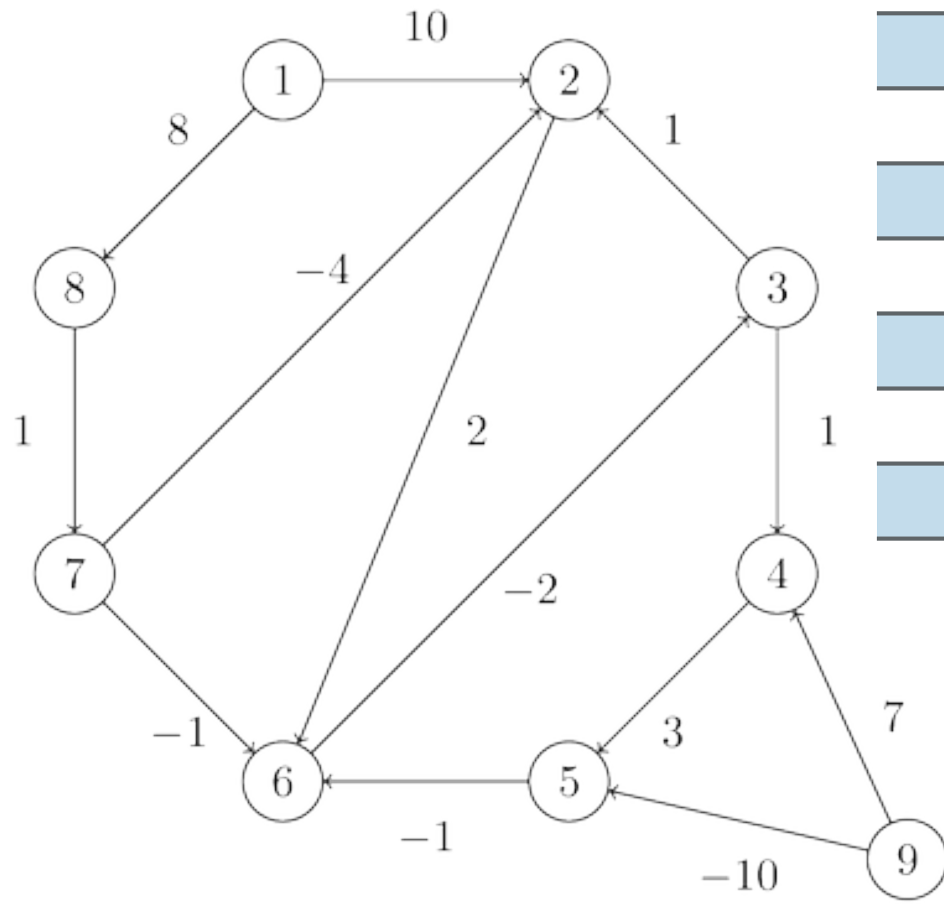
1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7



http://rosalind.info/problems/bf/

Practice Time

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

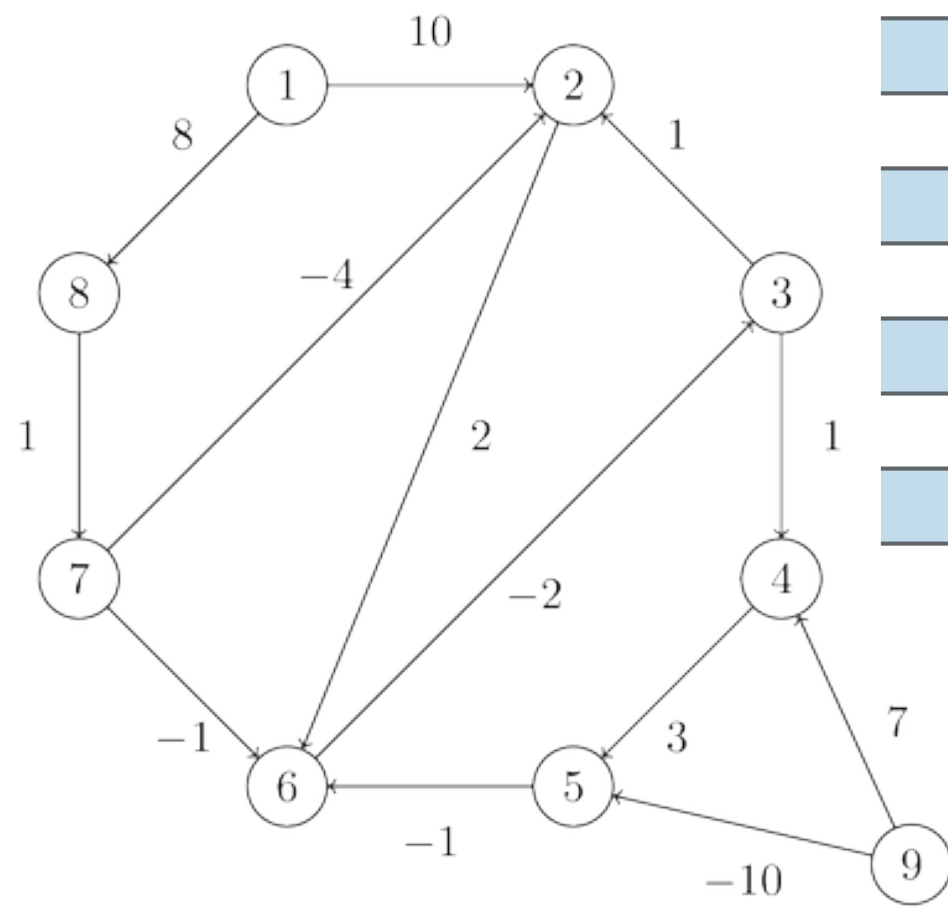


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	Inf	null
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	Inf	null
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

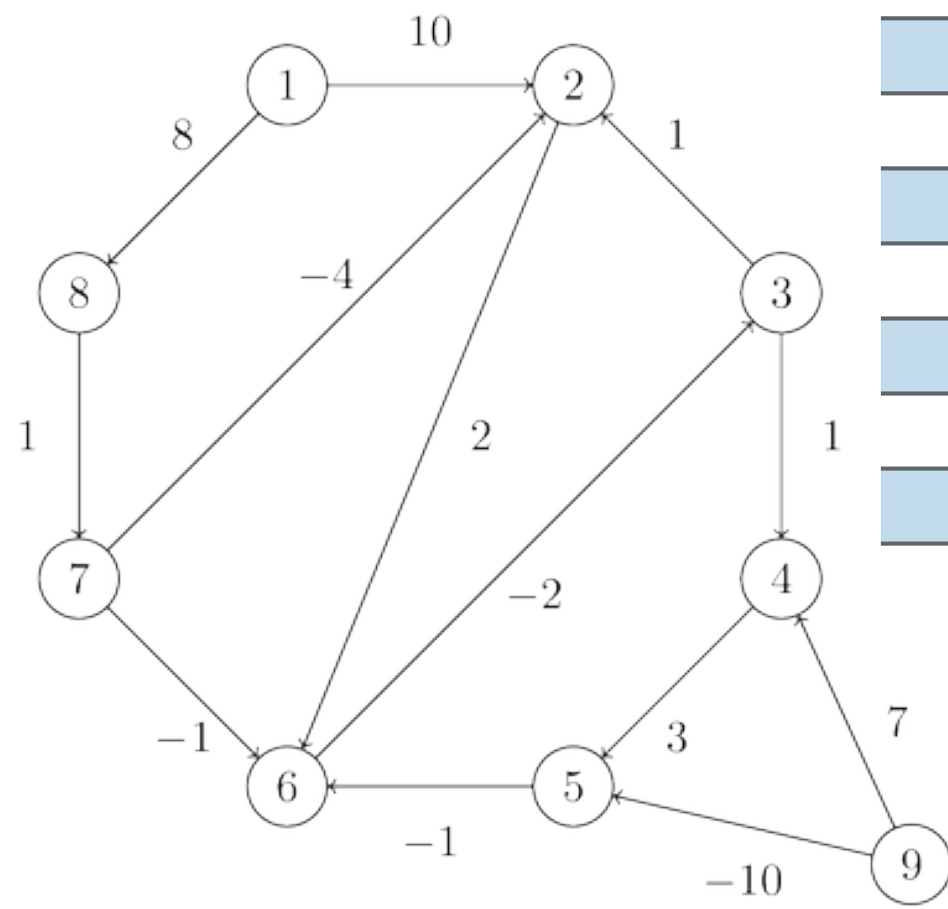


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	Inf	null
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

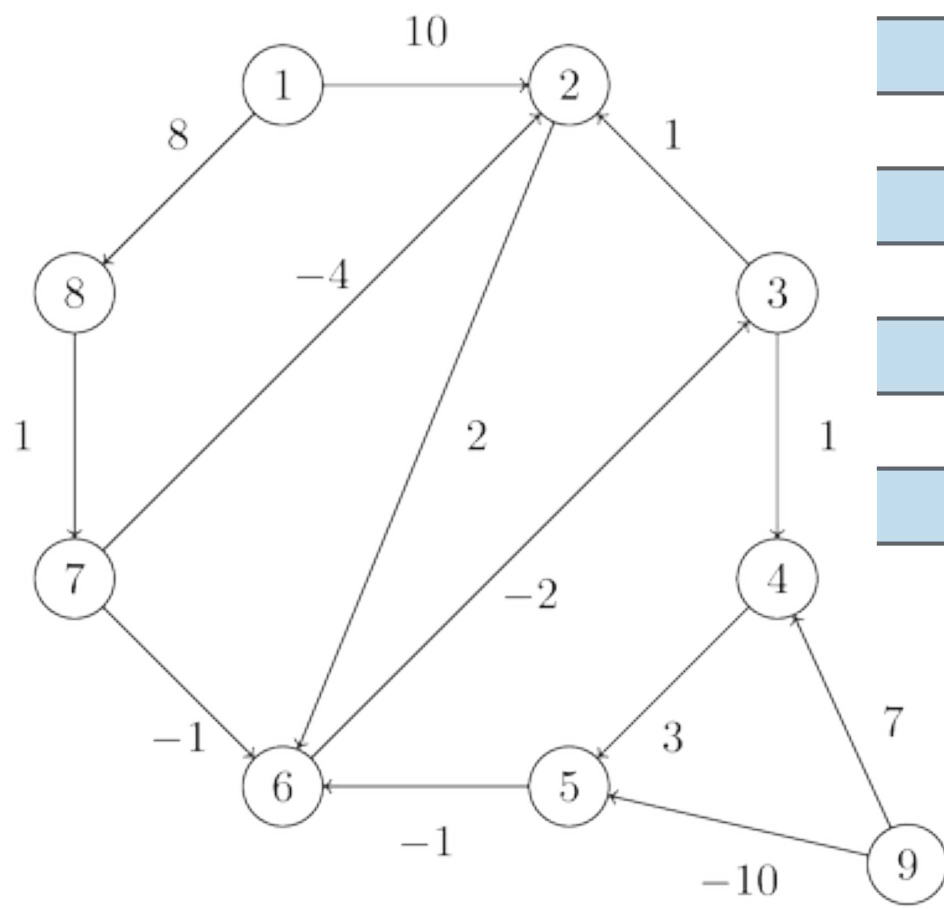


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	Inf	null
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

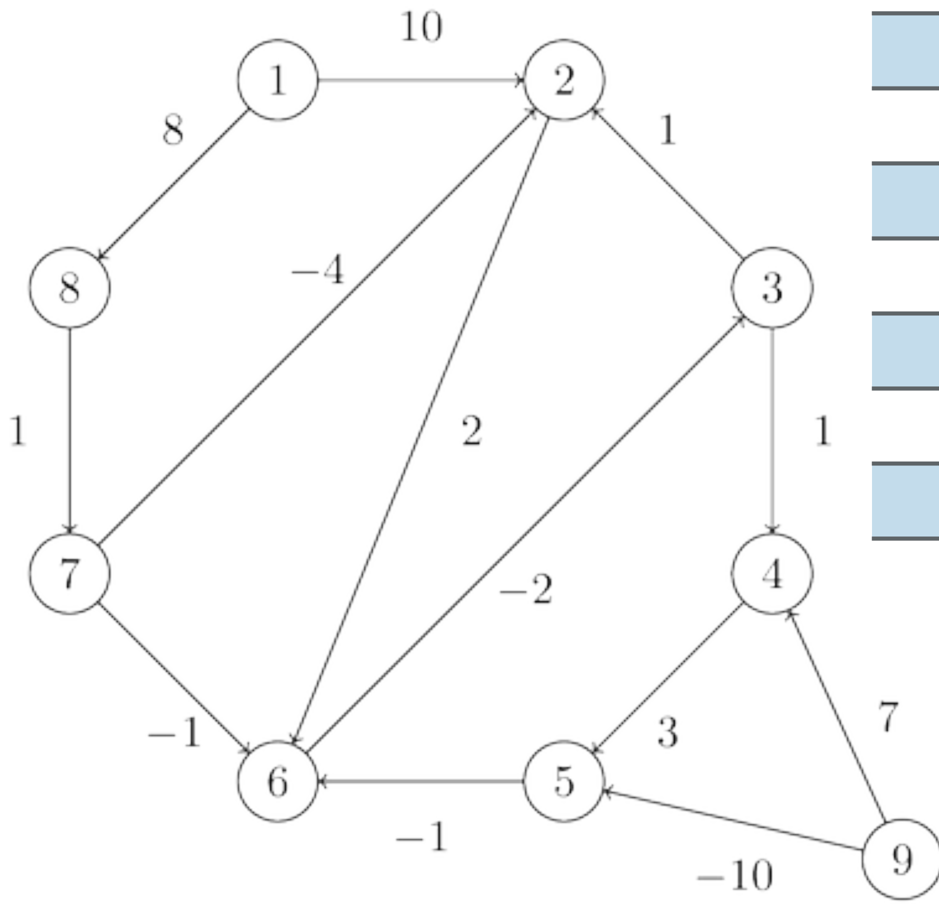


v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	Inf	null
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

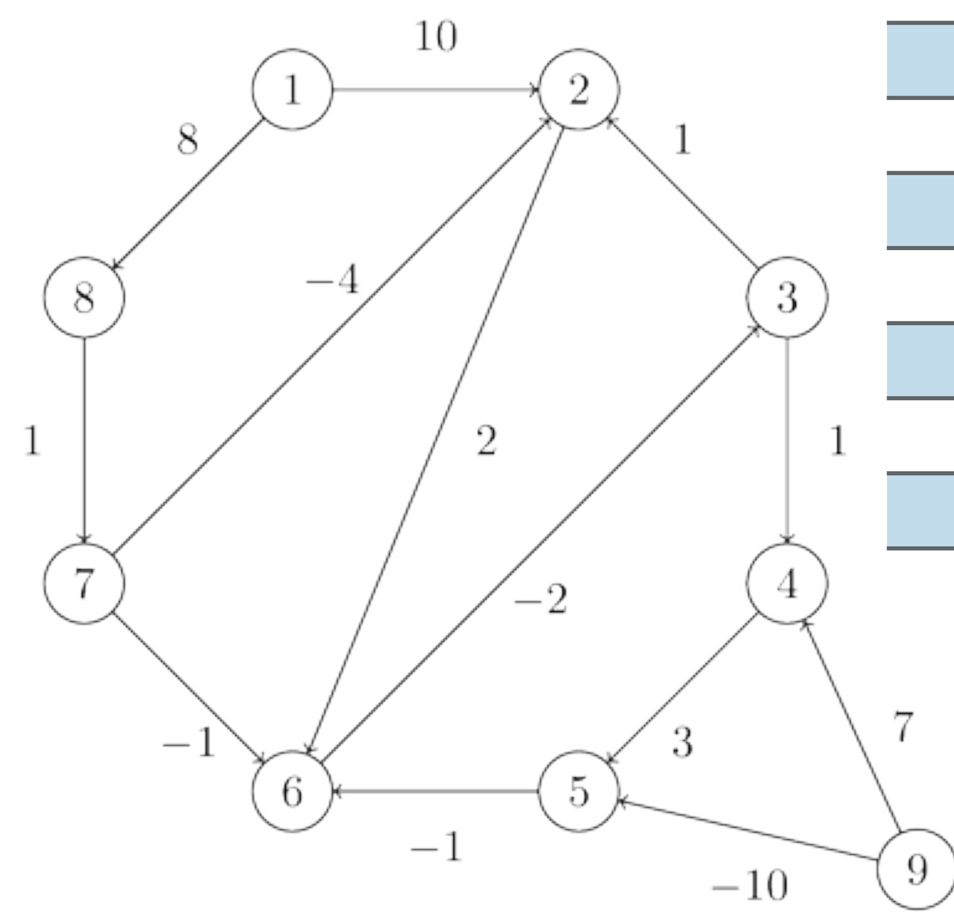


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	Inf	null
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

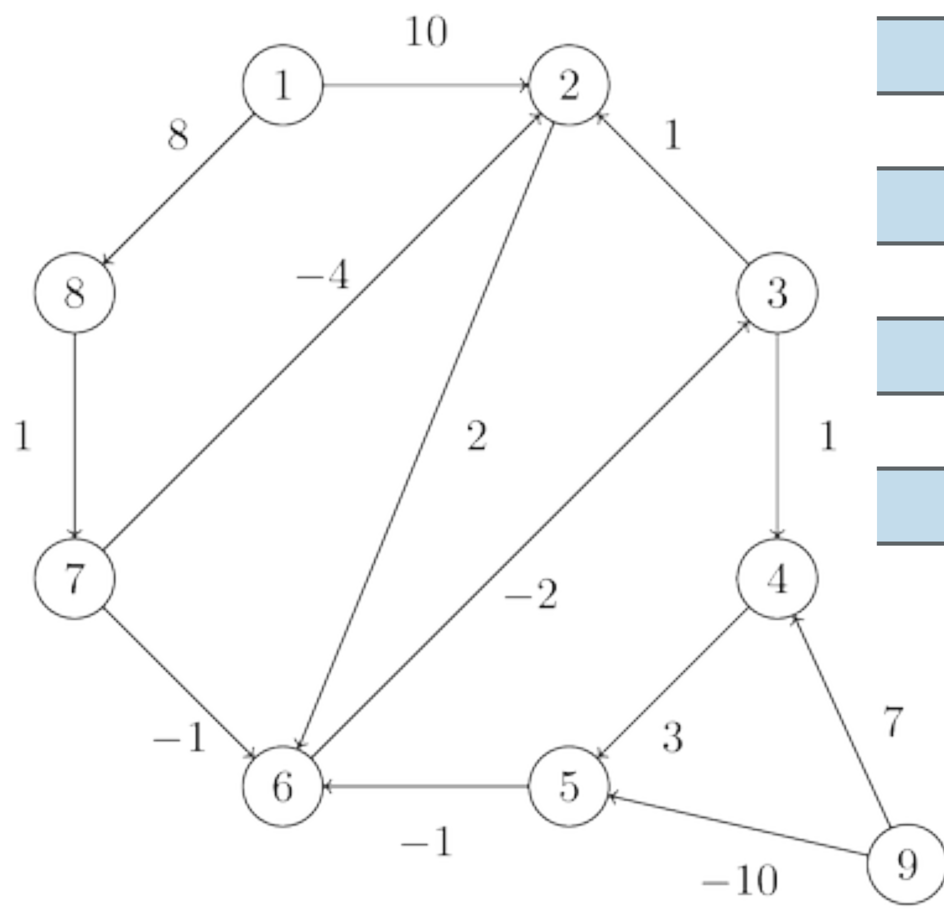


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	Inf	null
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

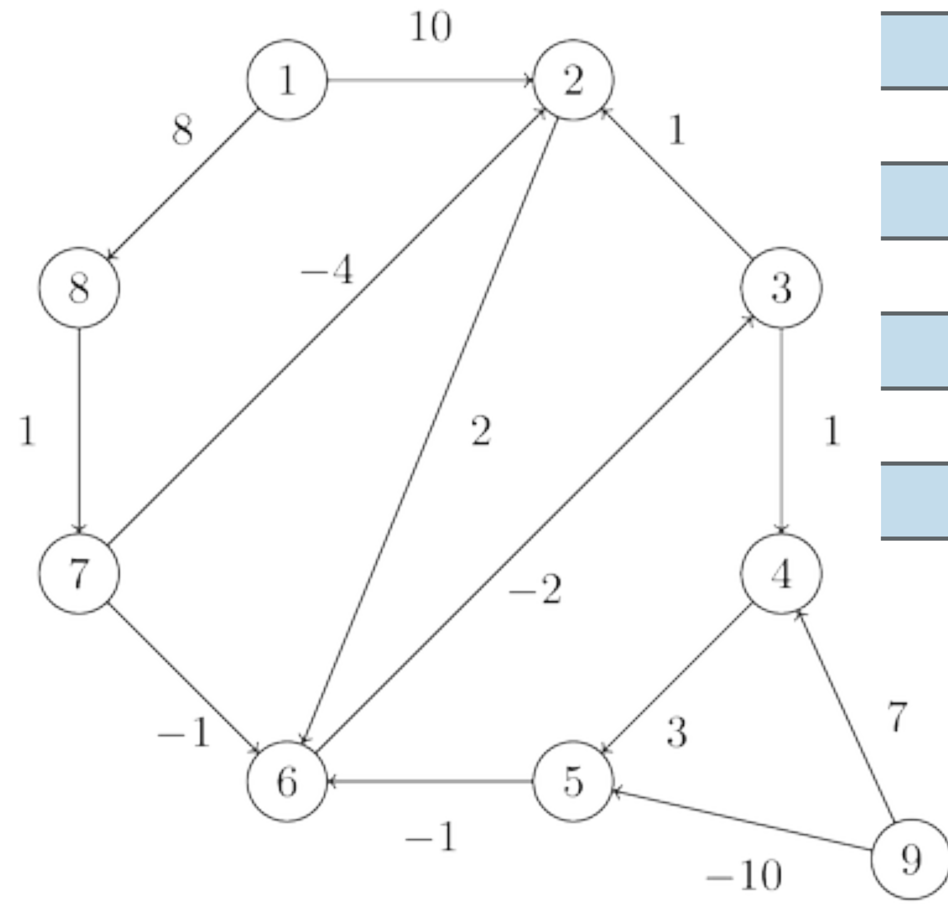


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	Inf	null
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

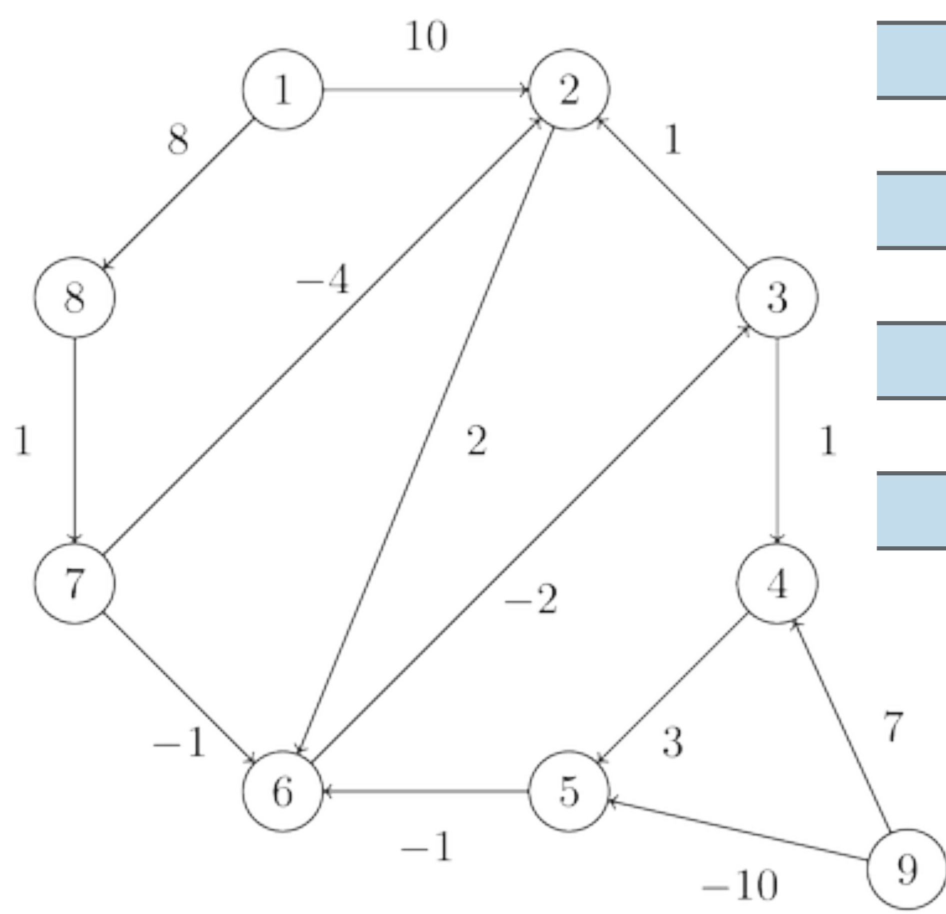


v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	Inf	null
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

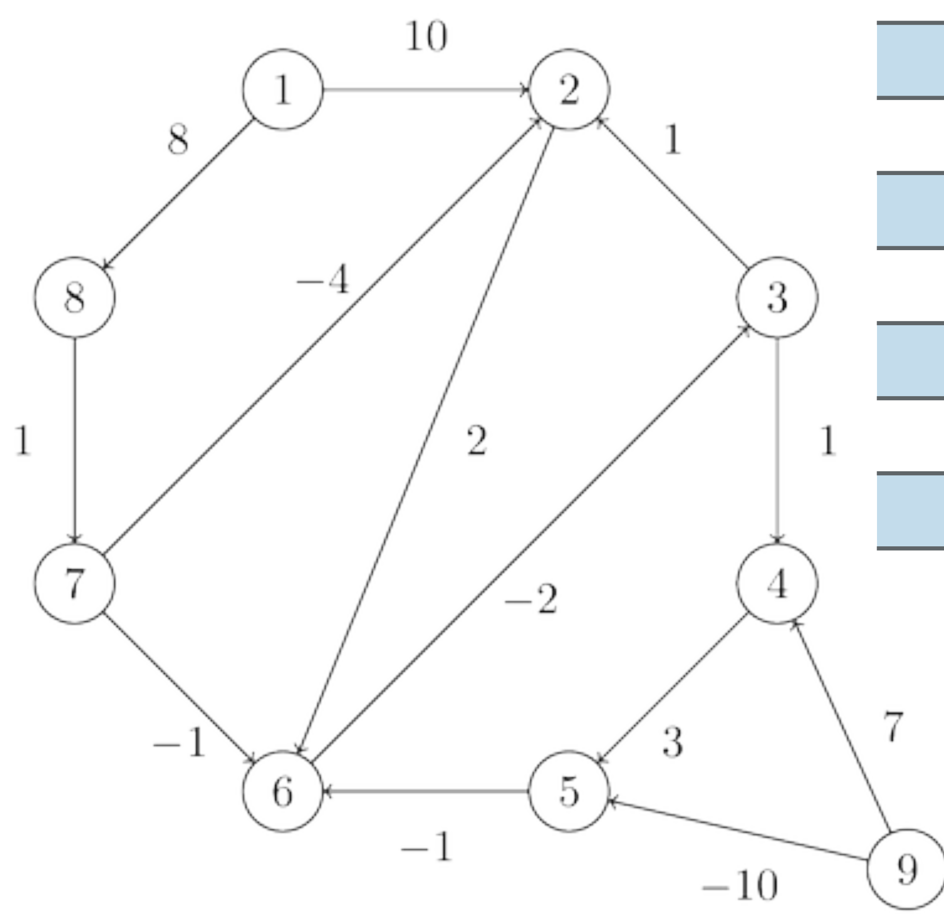


v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	8	1–>8
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

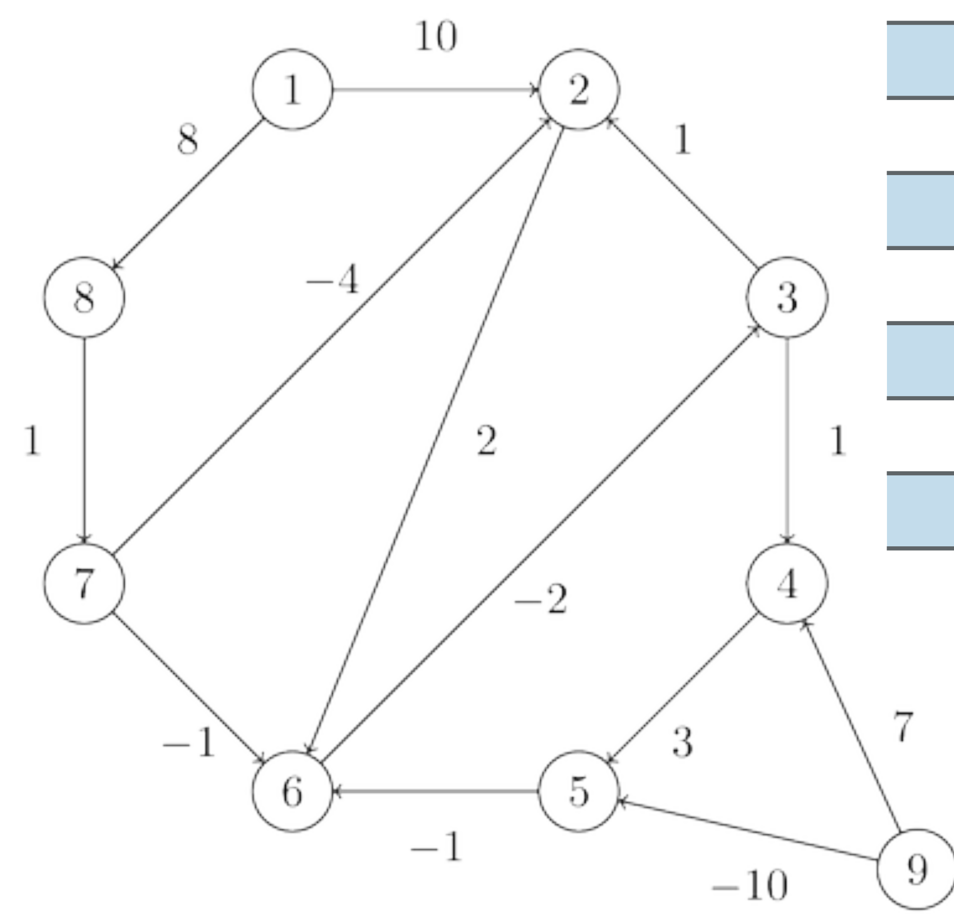


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	Inf	null
7	Inf	null
8	8	1–>8
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

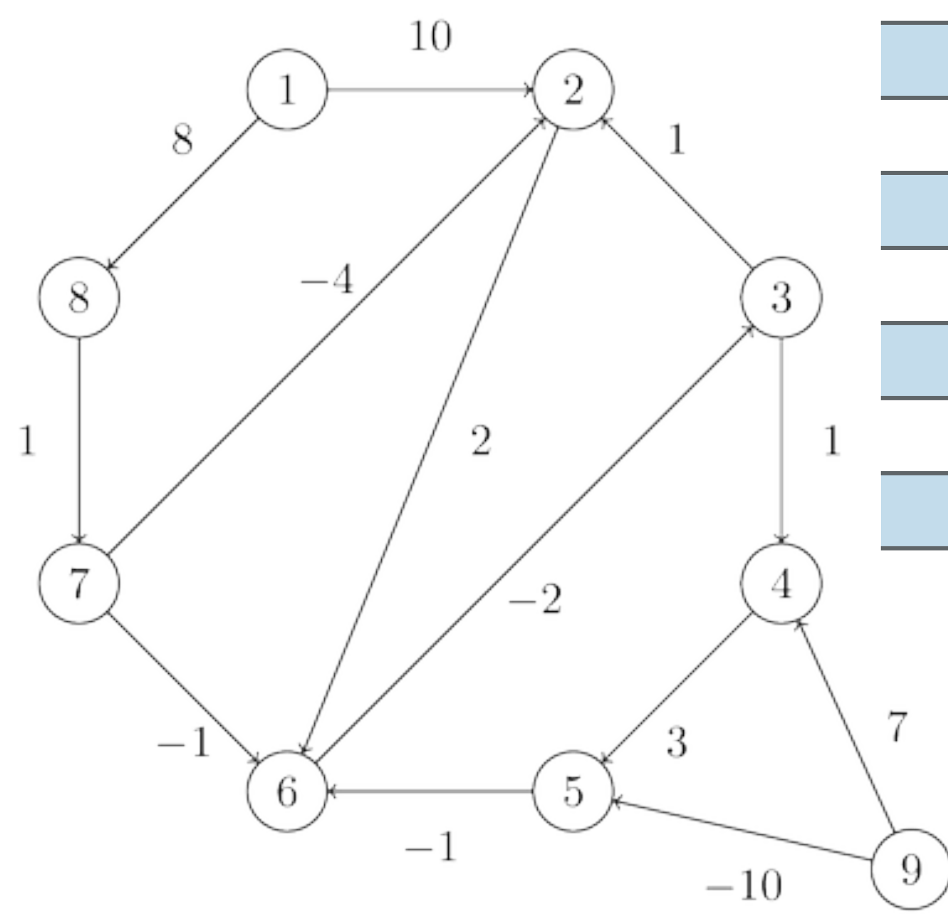


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	Inf	null
4	Inf	null
5	Inf	null
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

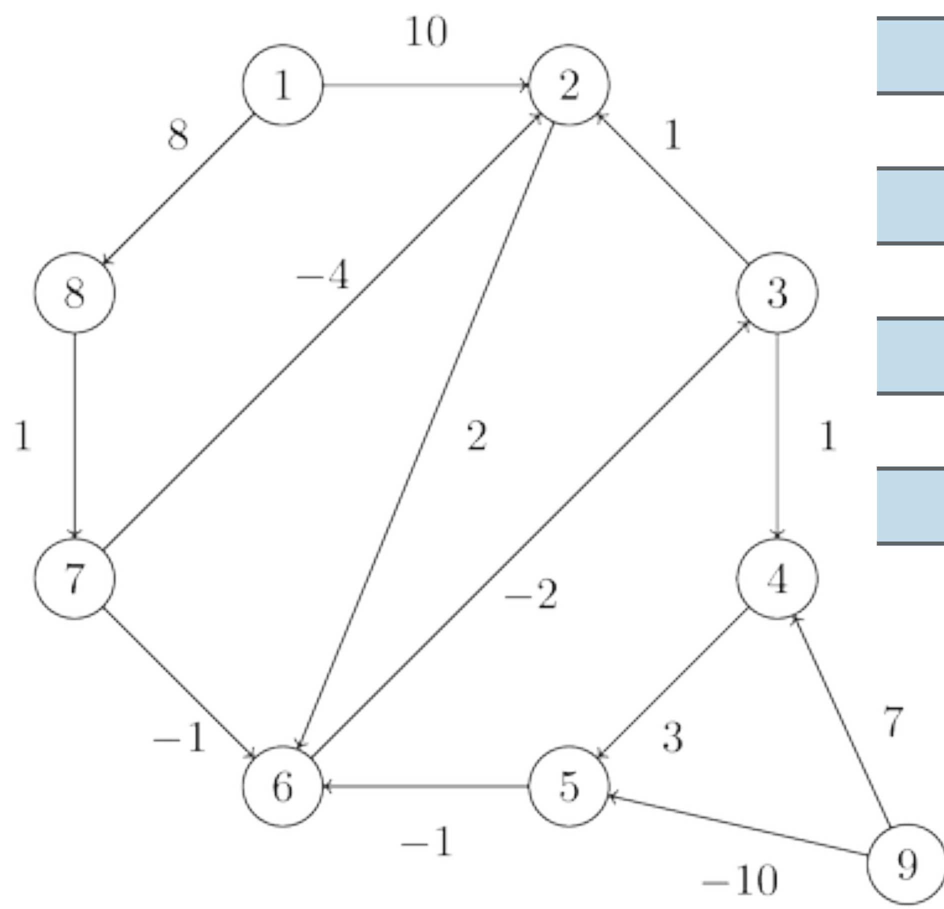


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	Inf	null
5	Inf	null
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

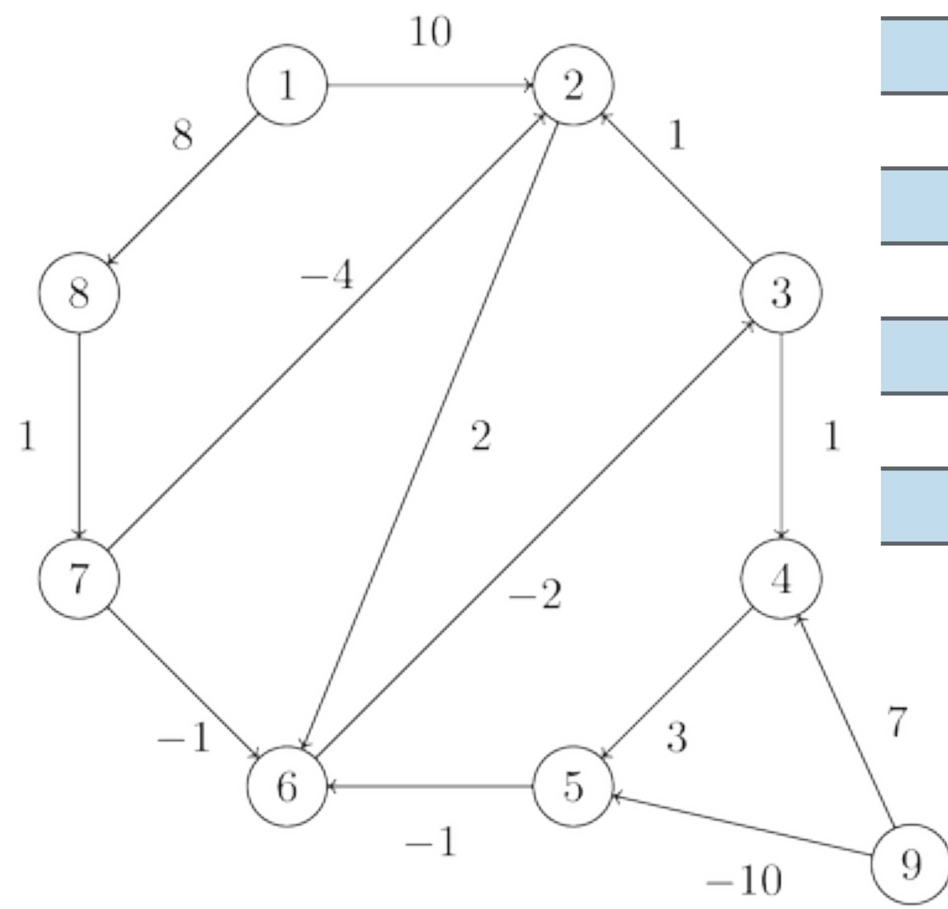


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	Inf	null
5	Inf	null
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

Practice Time - Pass 0

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

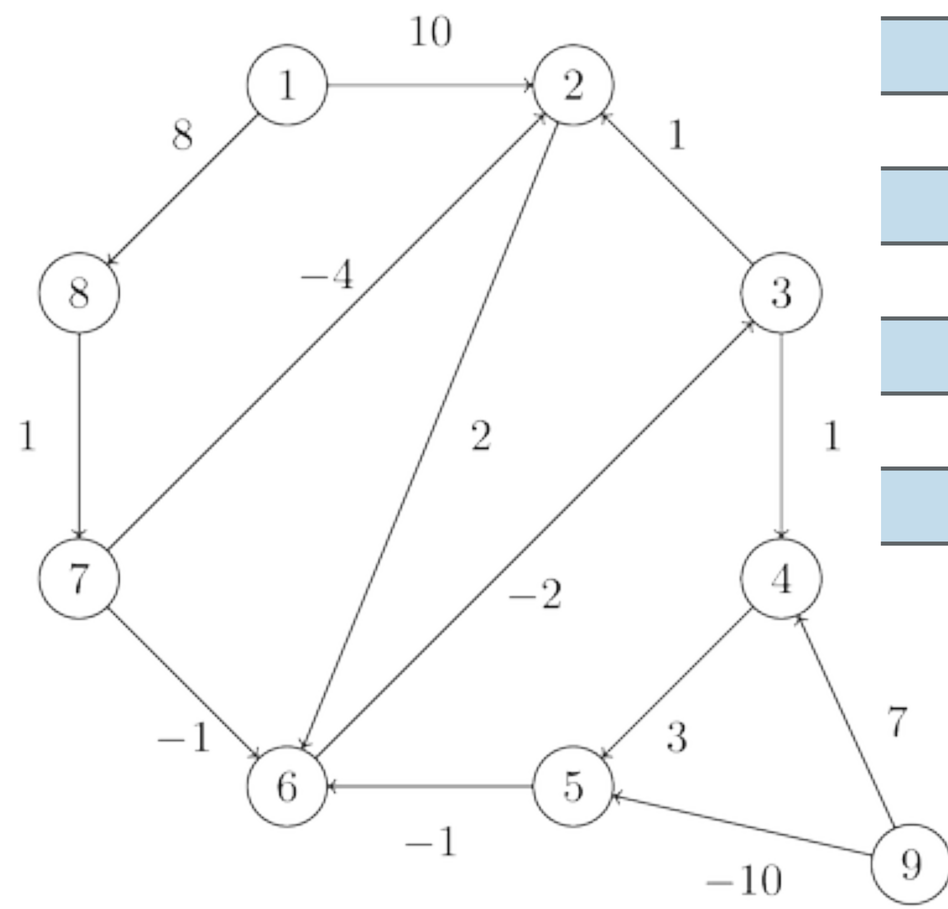


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	Inf	null
5	Inf	null
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

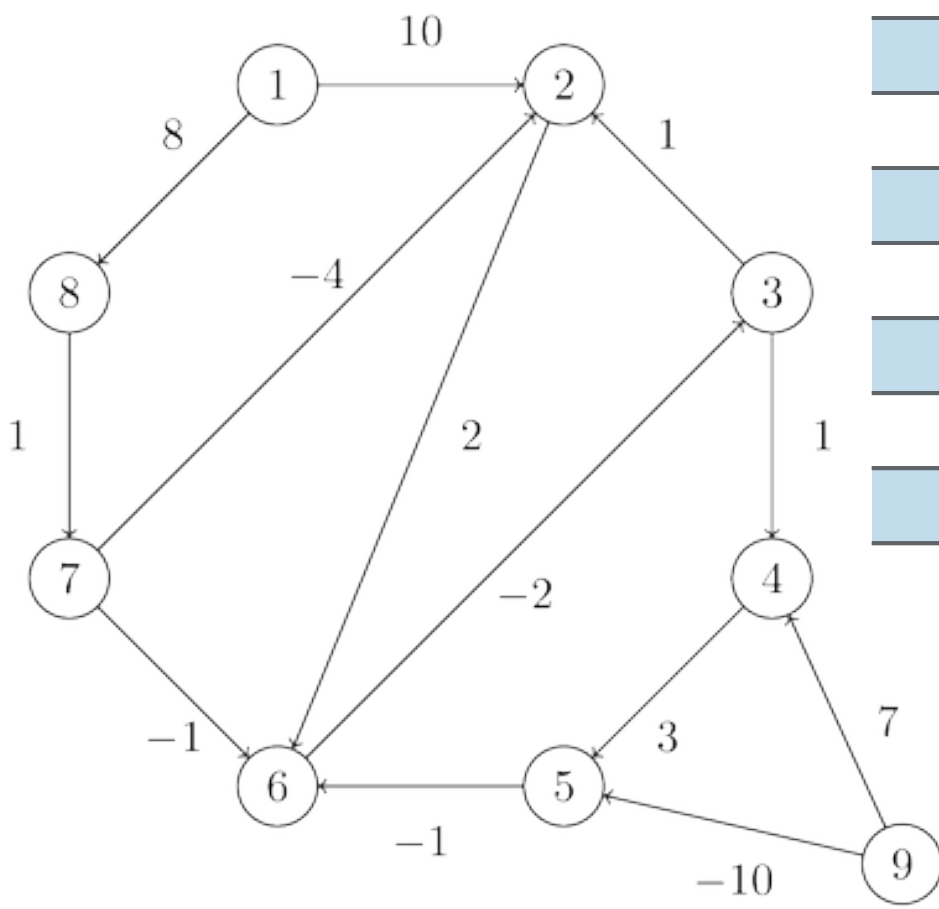


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	Inf	null
5	Inf	null
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

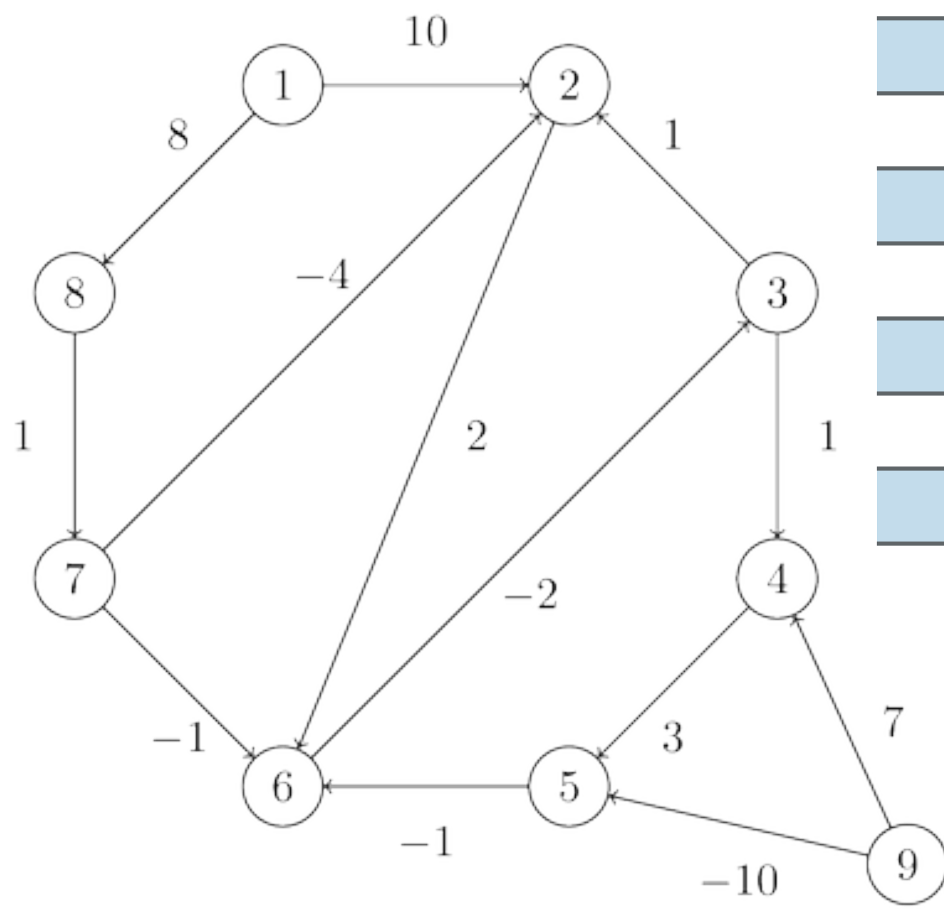


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	Inf	null
5	Inf	null
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

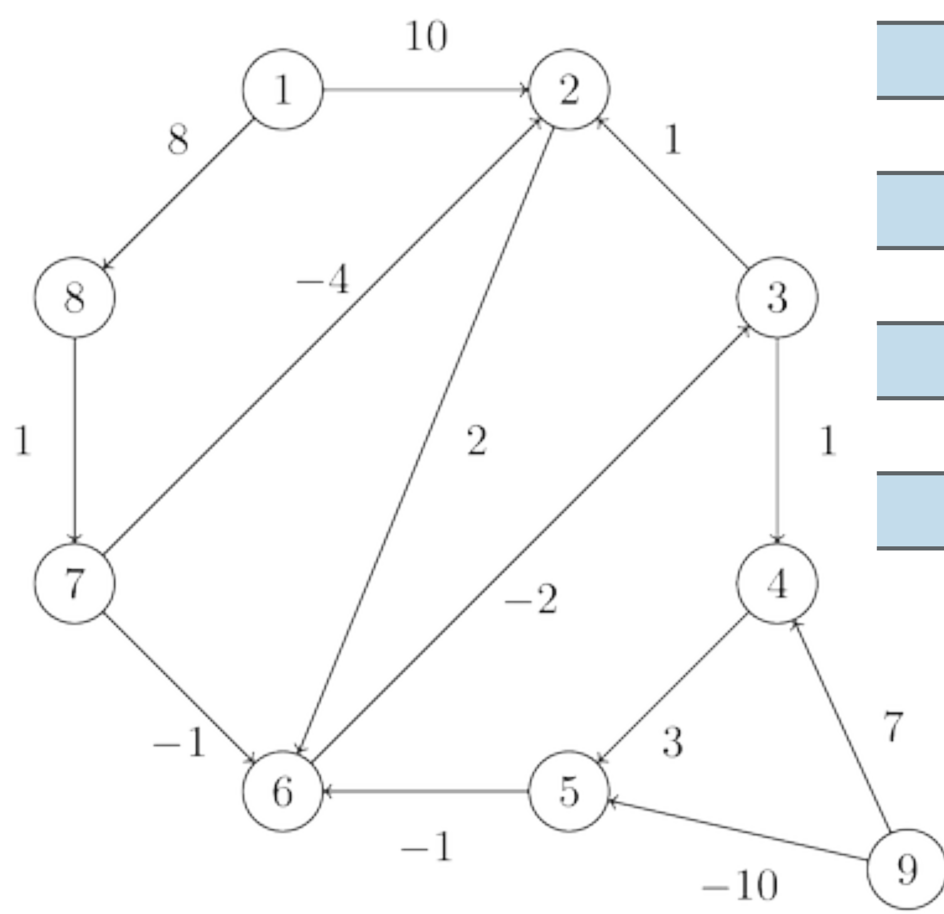


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	11	3–>4
5	Inf	null
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

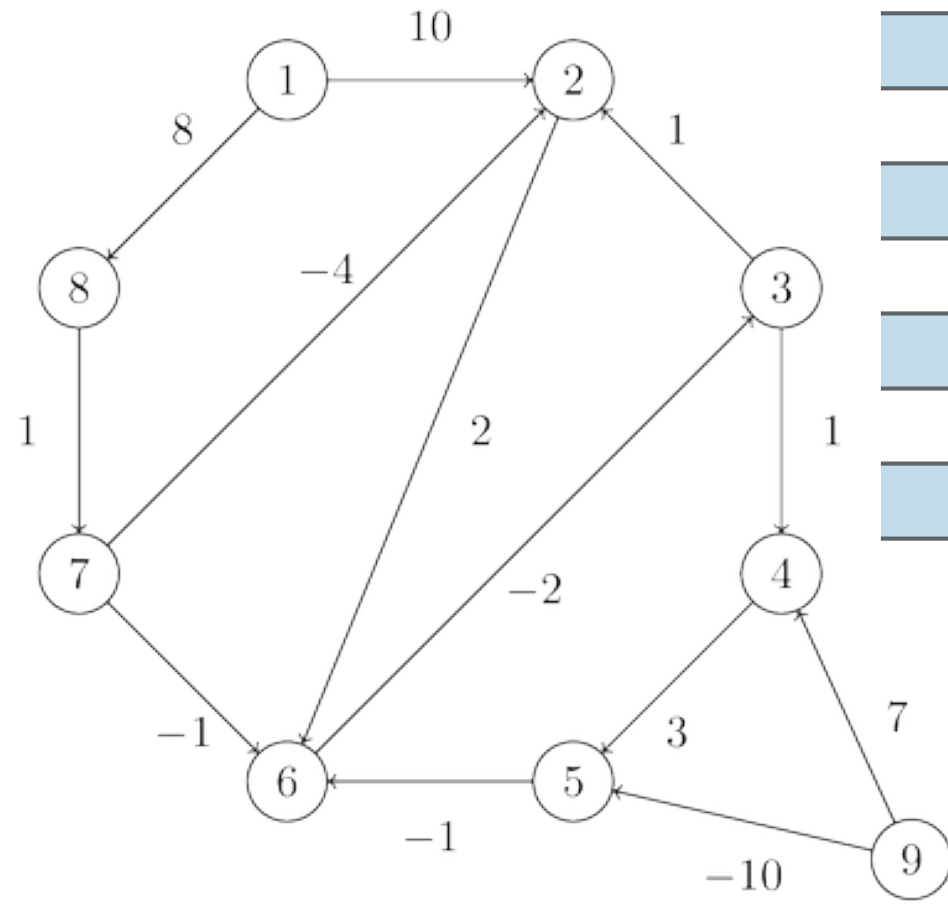


v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	11	3–>4
5	14	4–>5
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

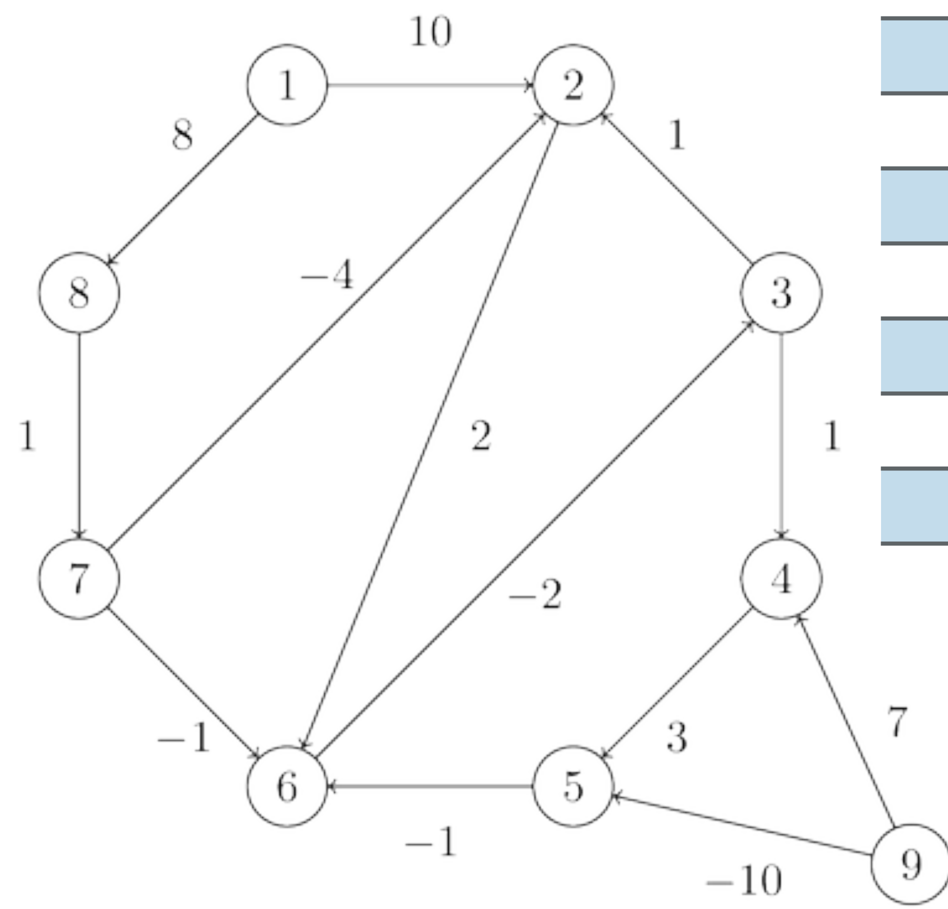


v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	11	3–>4
5	14	4–>5
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

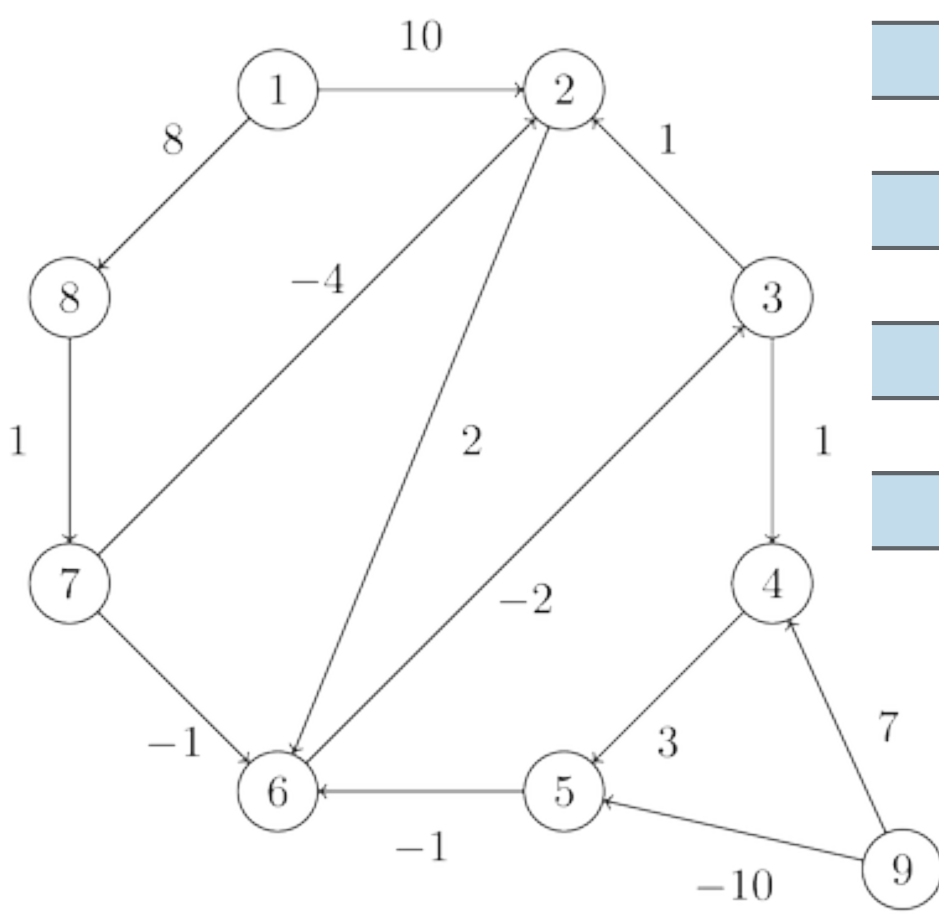


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	11	3–>4
5	14	4–>5
6	12	2–>6
7	Inf	null
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

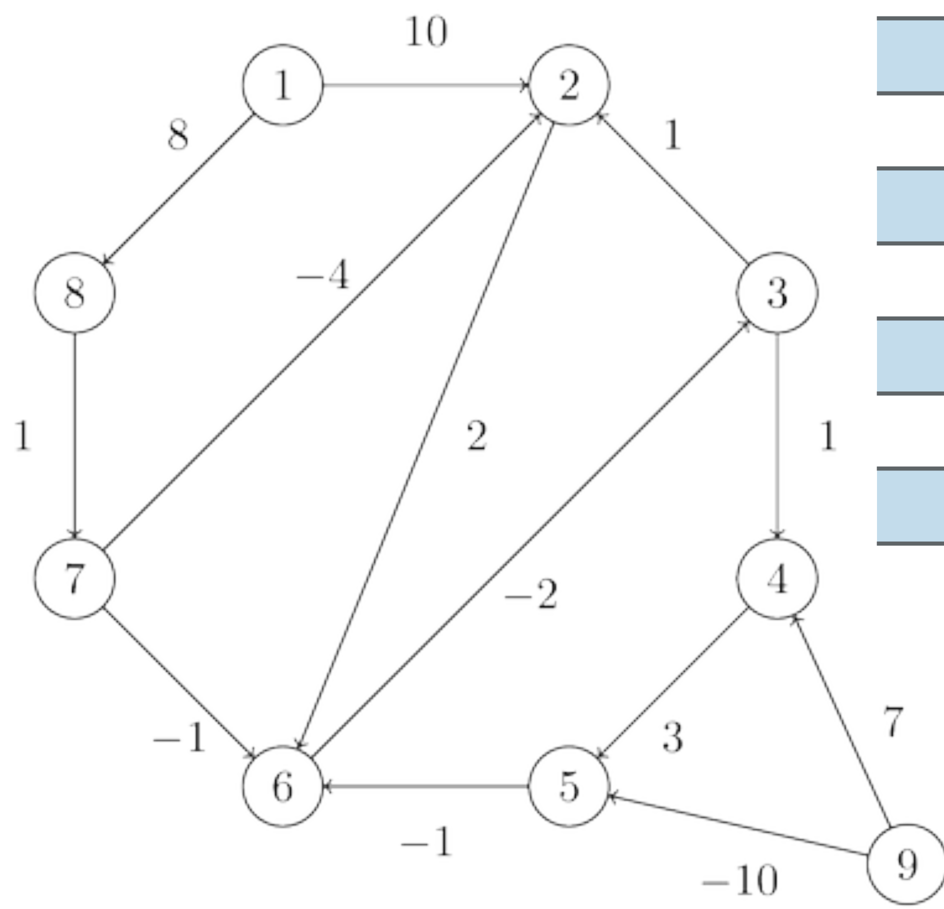


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	11	3–>4
5	14	4–>5
6	12	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

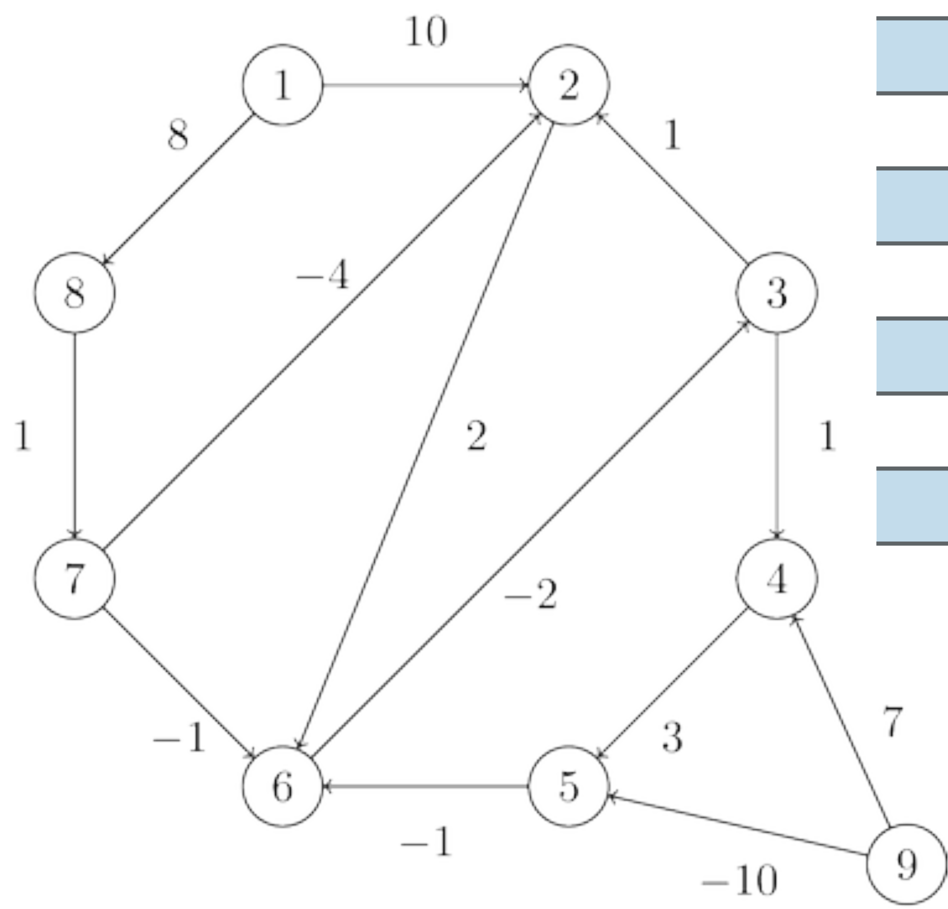


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	10	1–> 2
3	10	6–>3
4	11	3–>4
5	14	4–>5
6	12	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

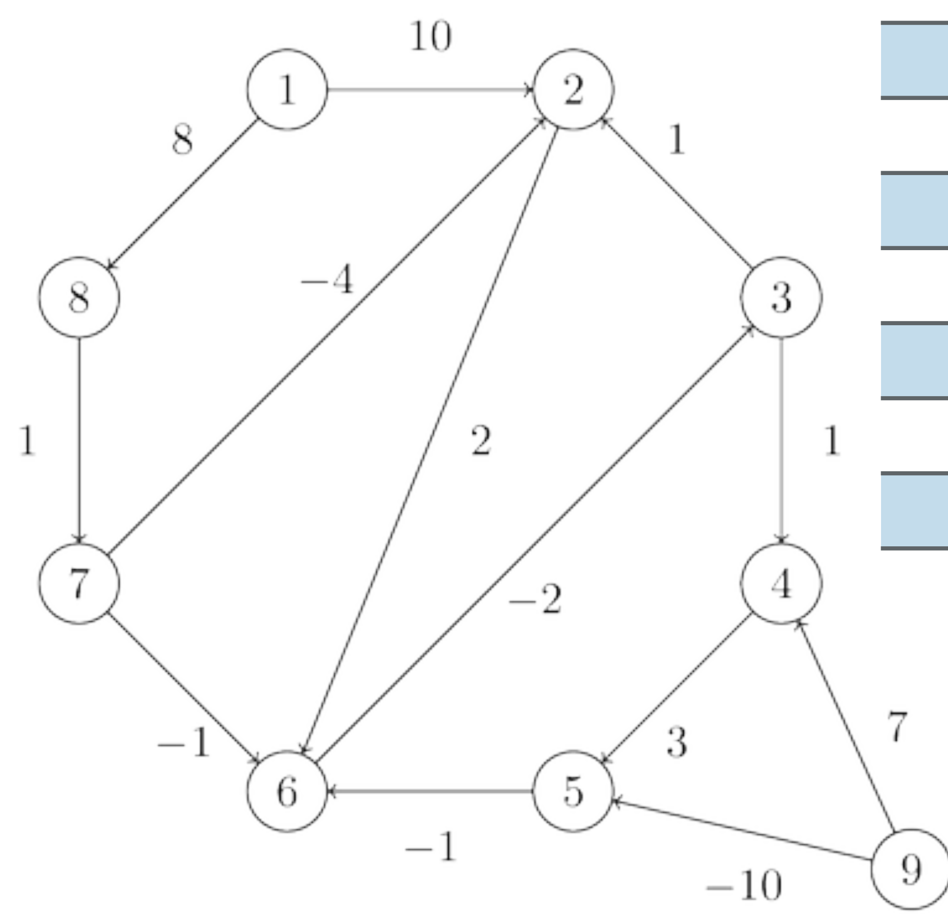


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	10	6–>3
4	11	3–>4
5	14	4–>5
6	12	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

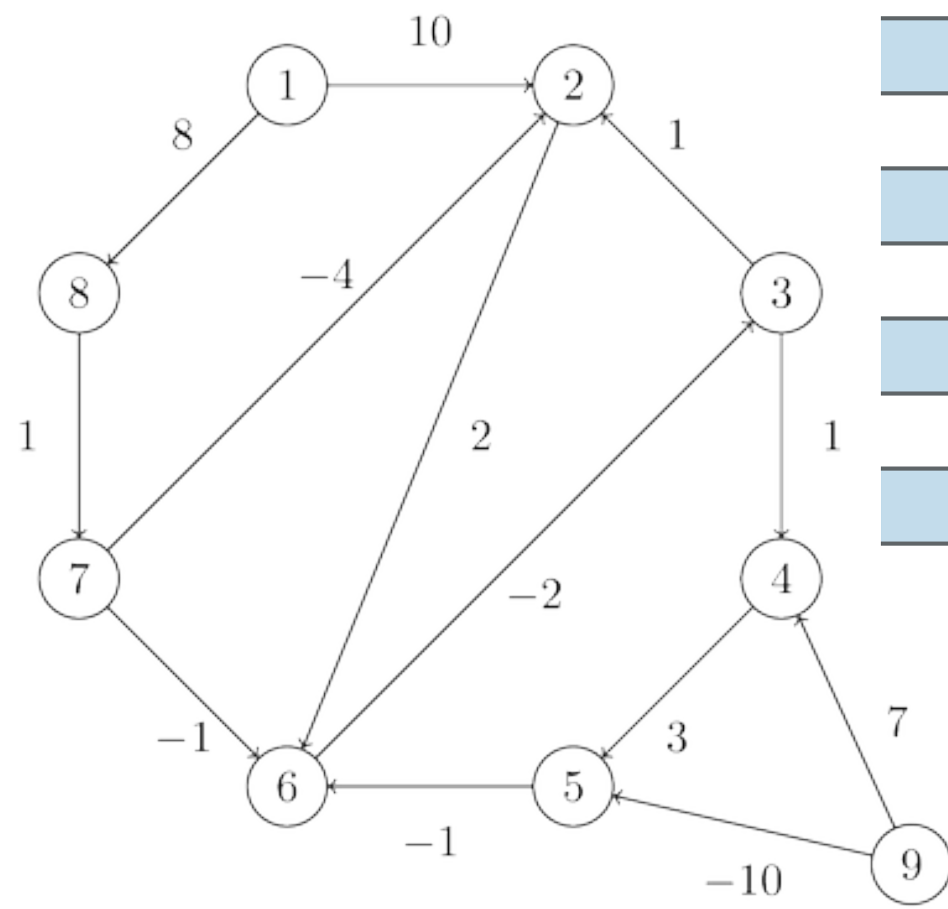


v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	10	6–>3
4	11	3–>4
5	14	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

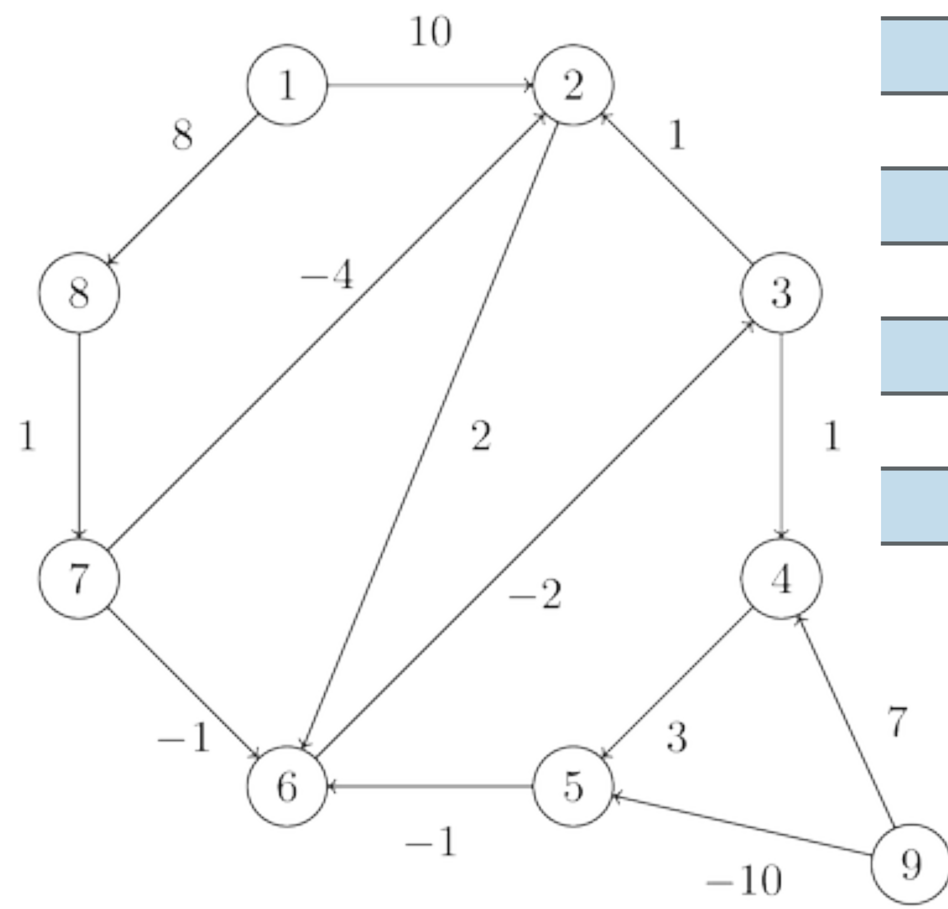


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	11	3–>4
5	14	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

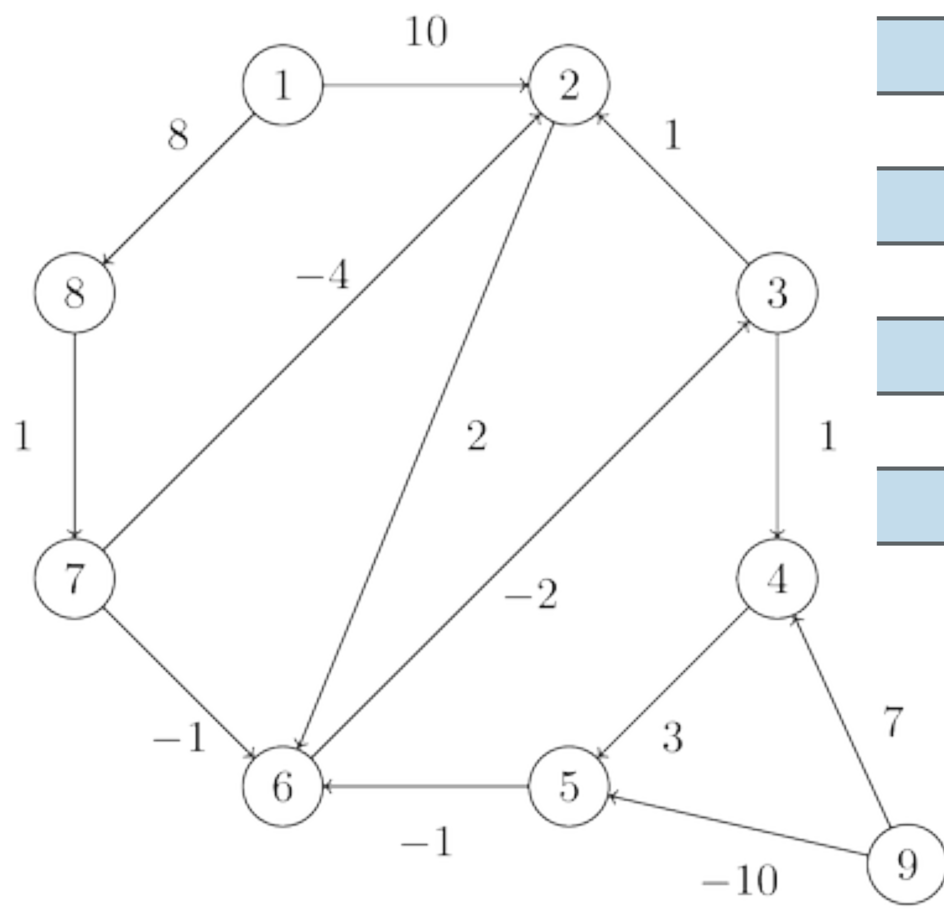


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	11	3–>4
5	14	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 1

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

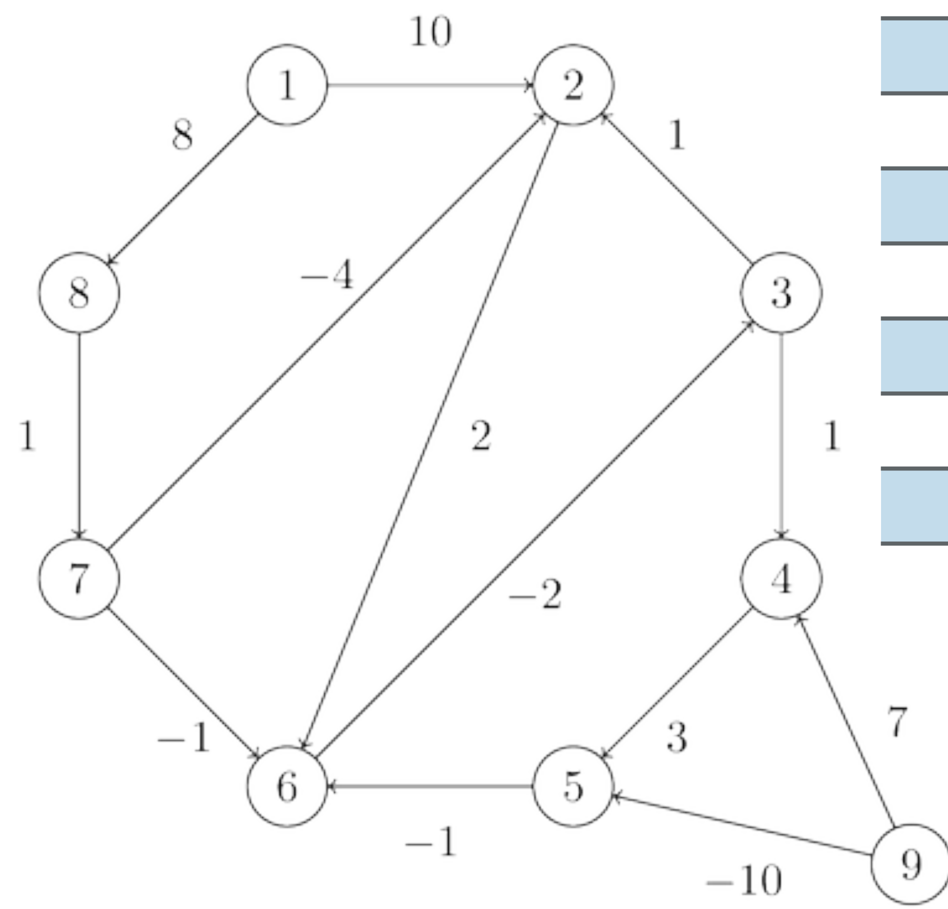


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	11	3–>4
5	14	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

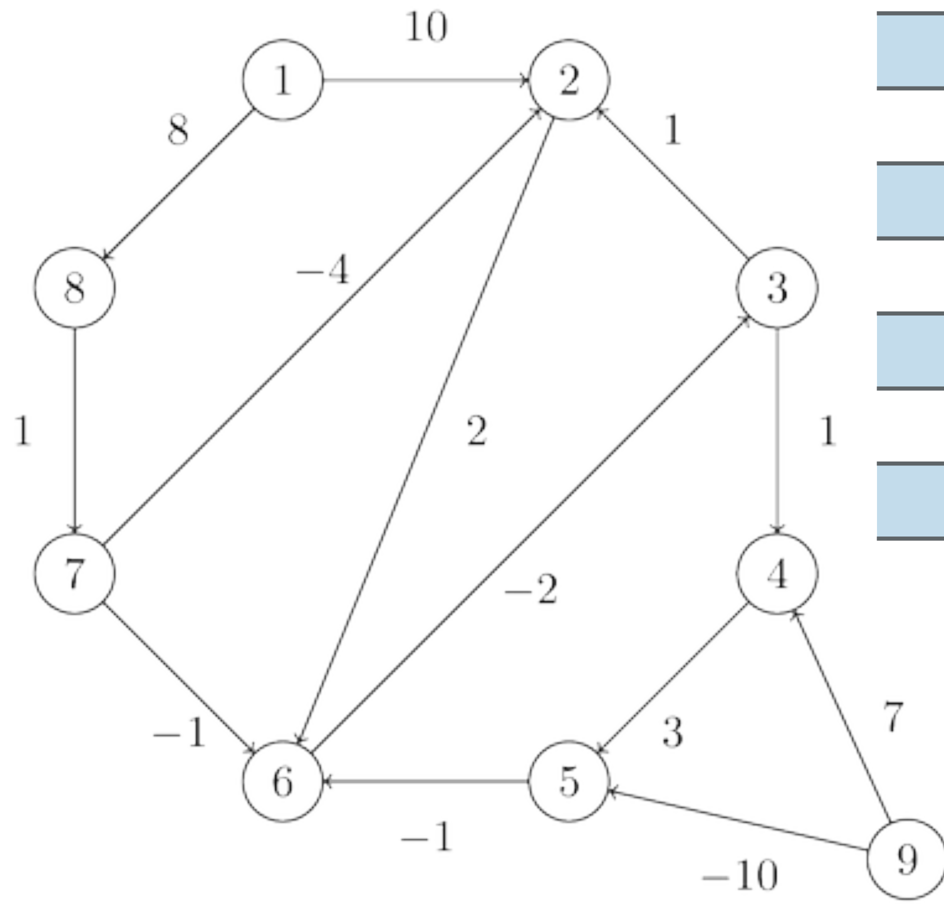


v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	11	3–>4
5	14	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

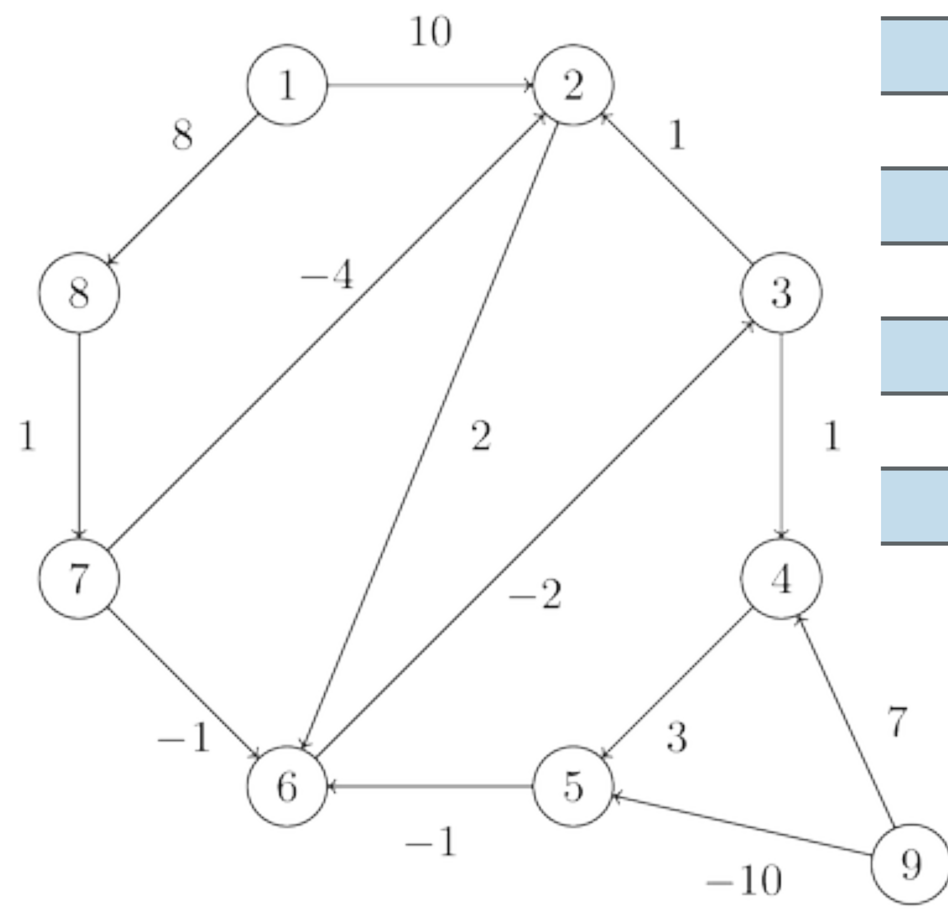


v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	11	3–>4
5	14	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

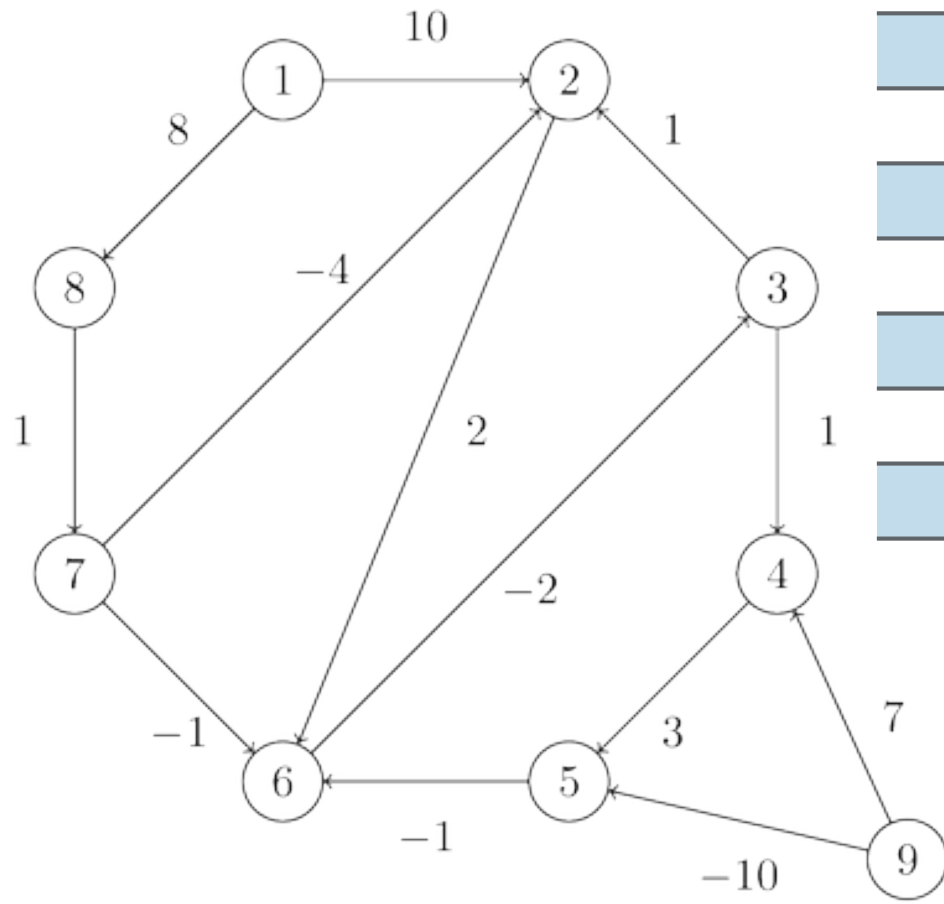


v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	14	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

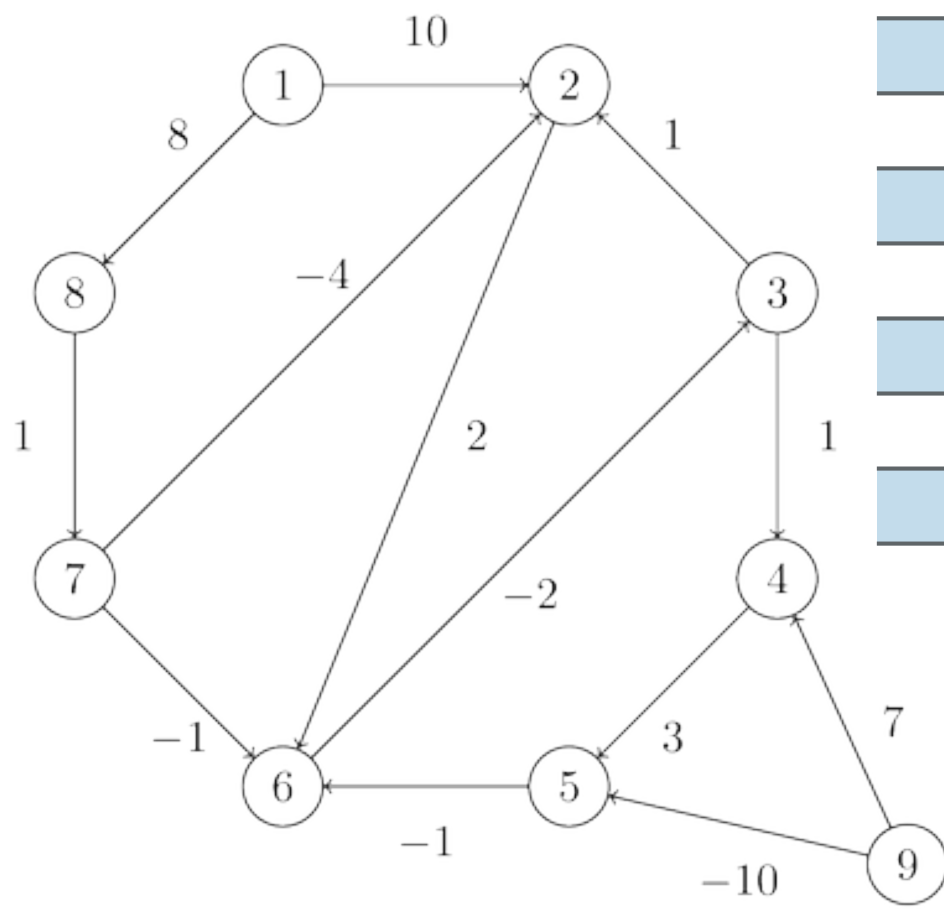


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

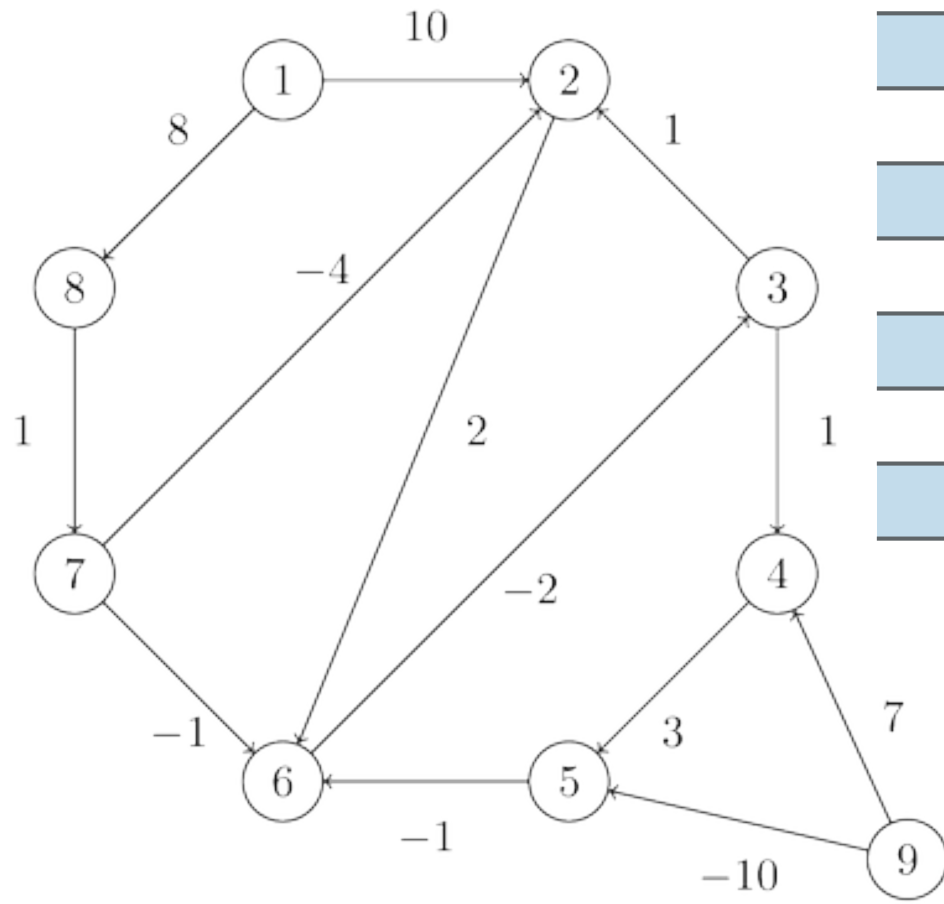


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

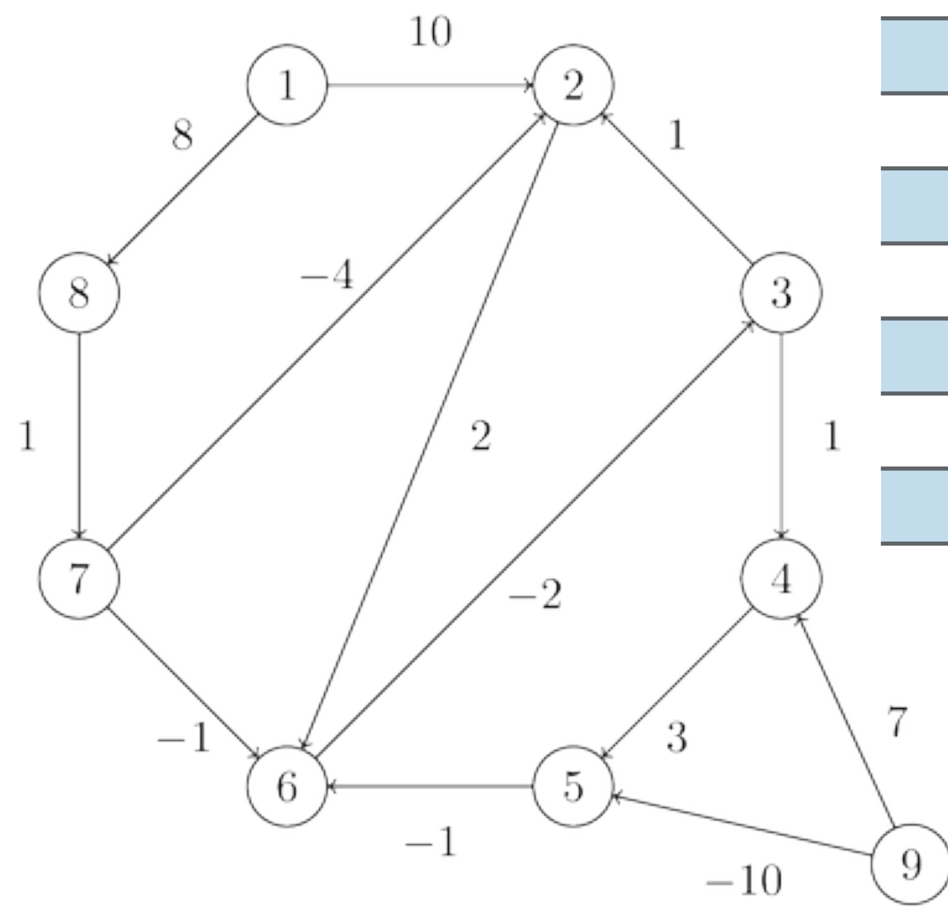


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

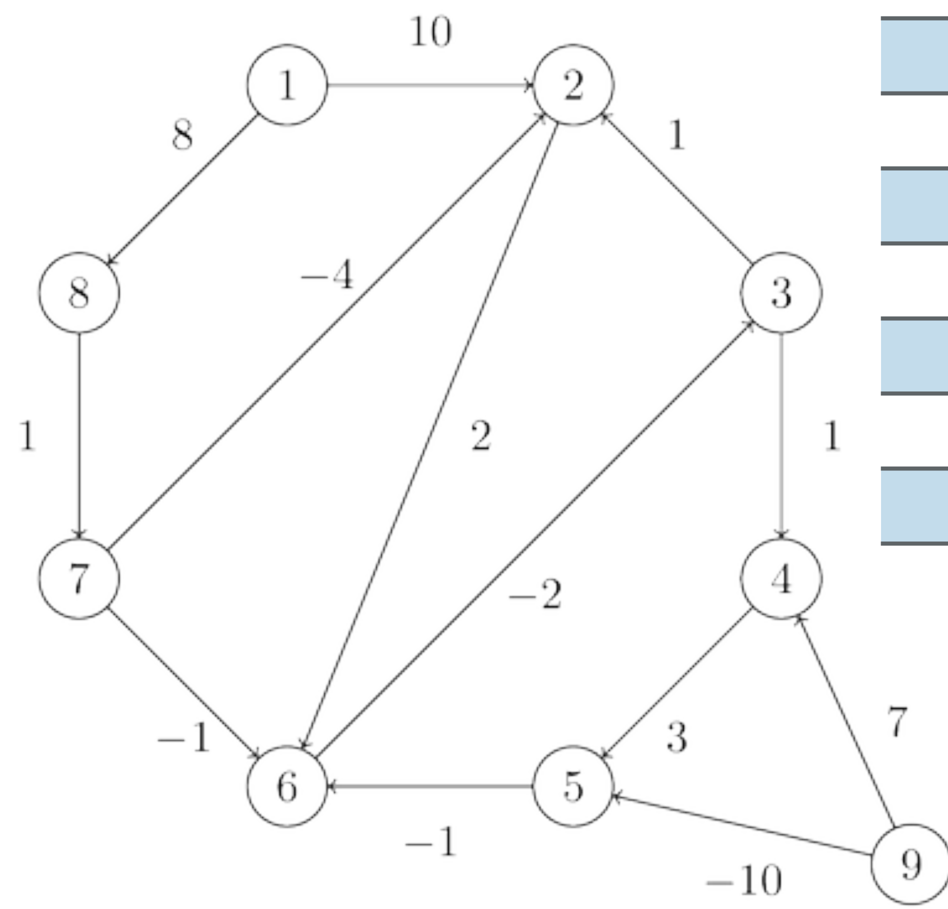


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

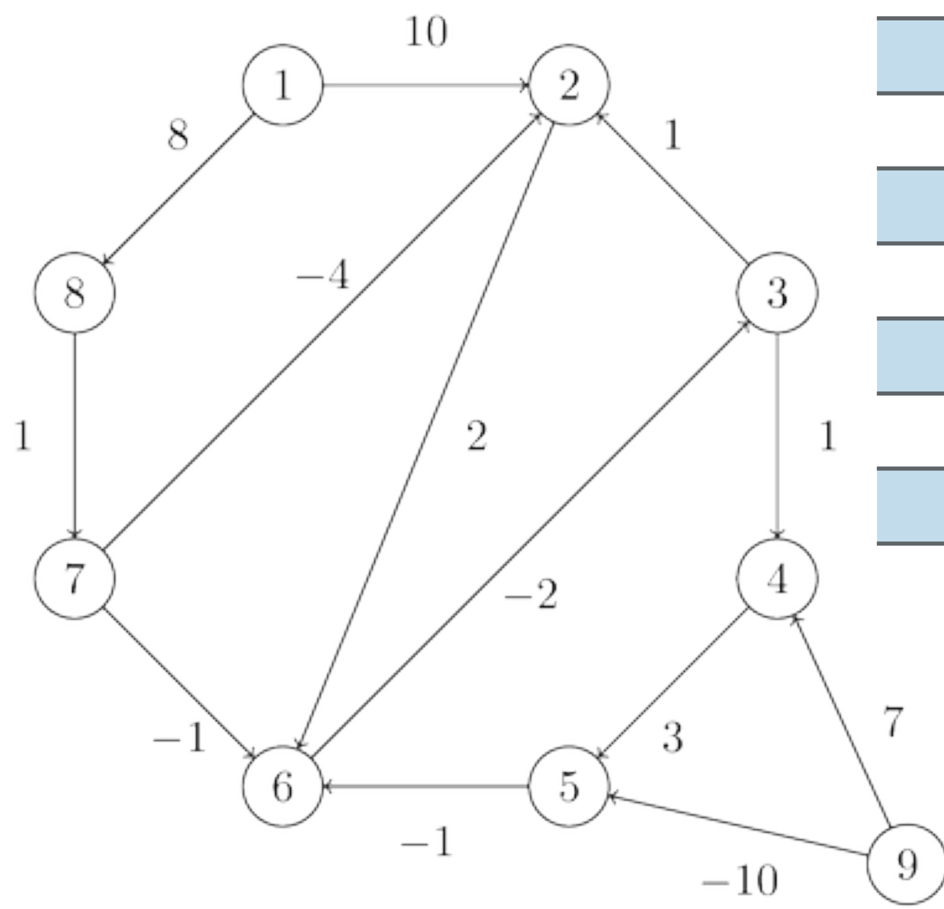


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

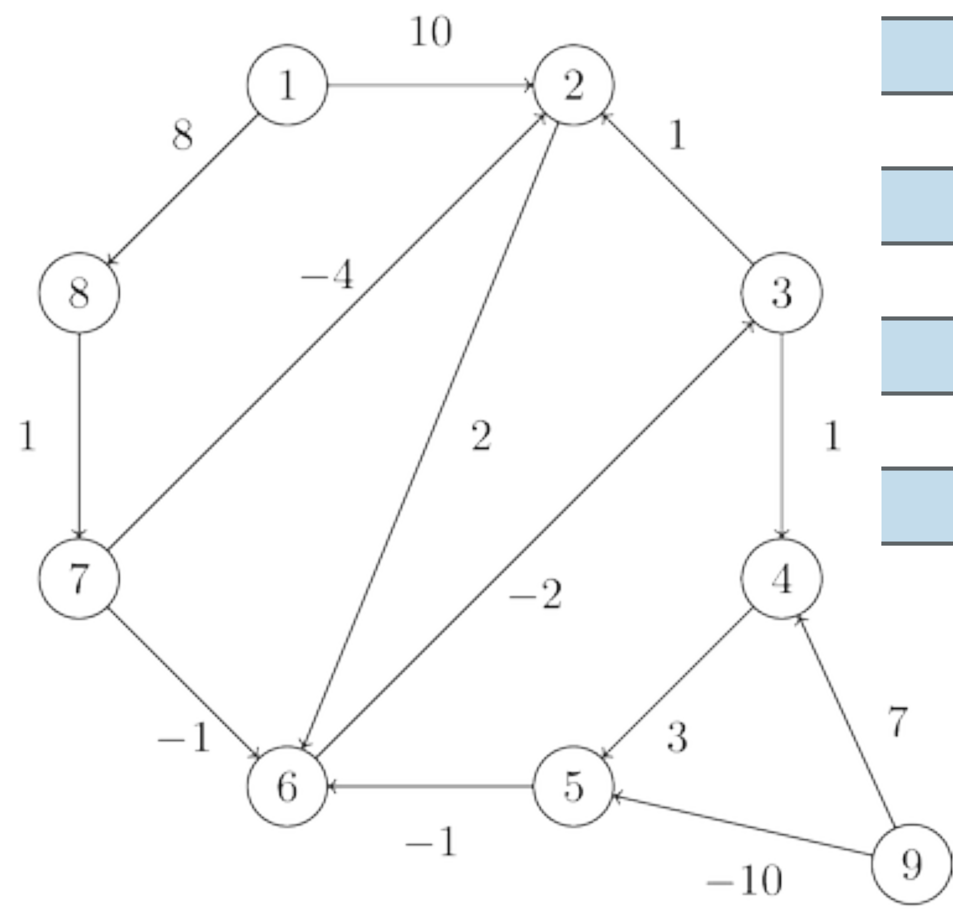


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

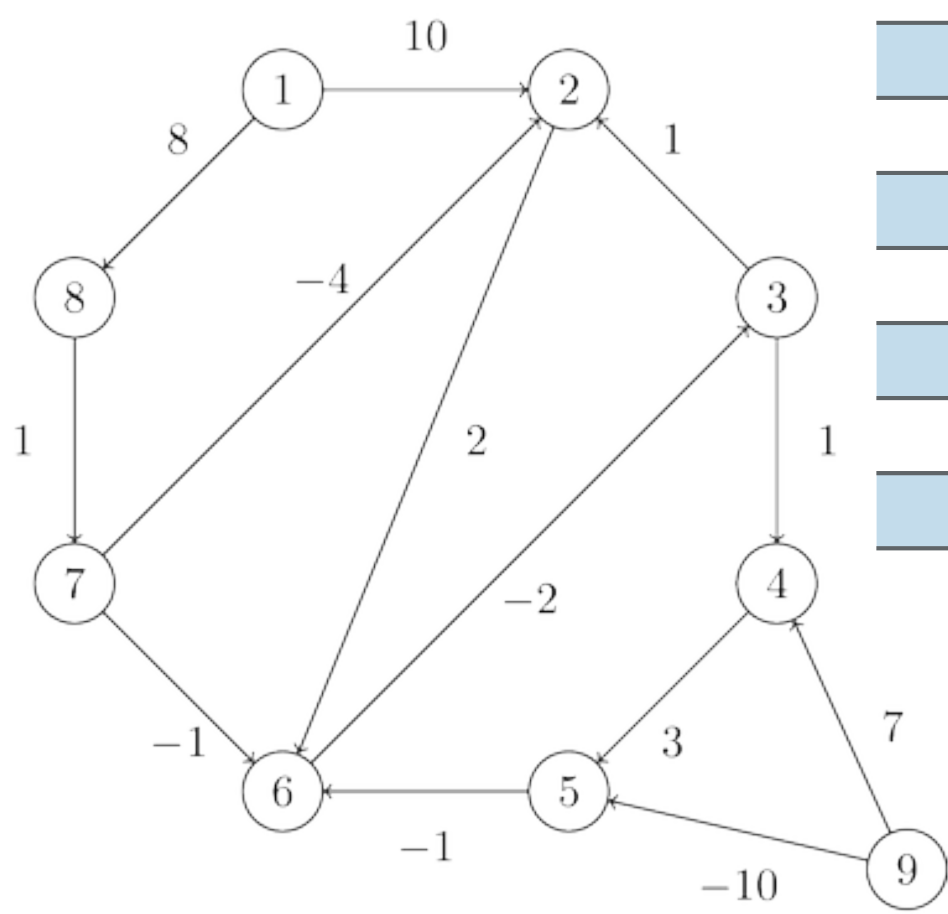


v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

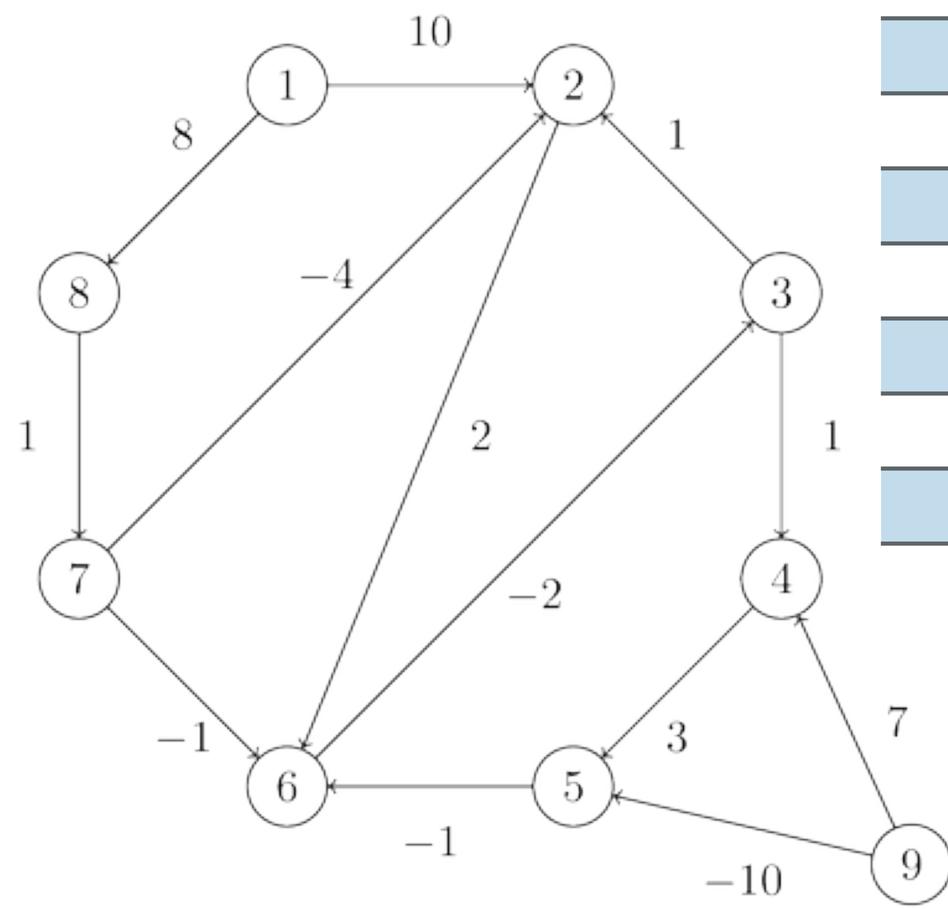


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

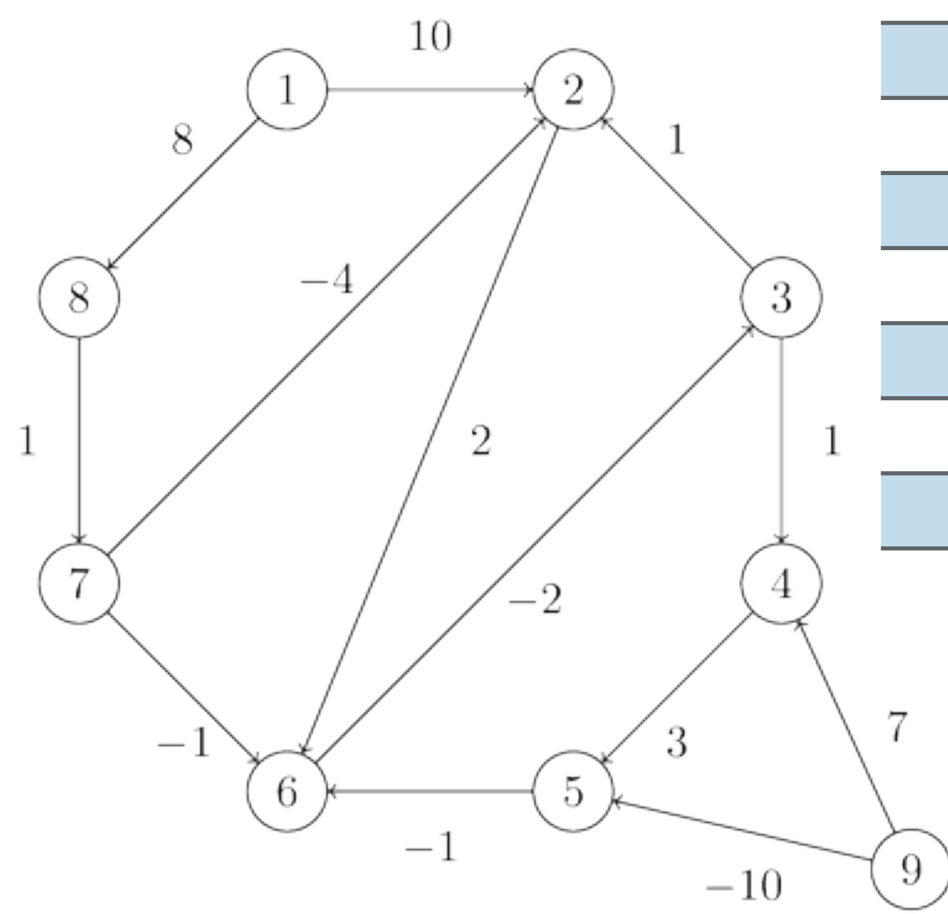


v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

http://rosalind.info/problems/bf/

Practice Time - Pass 2

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

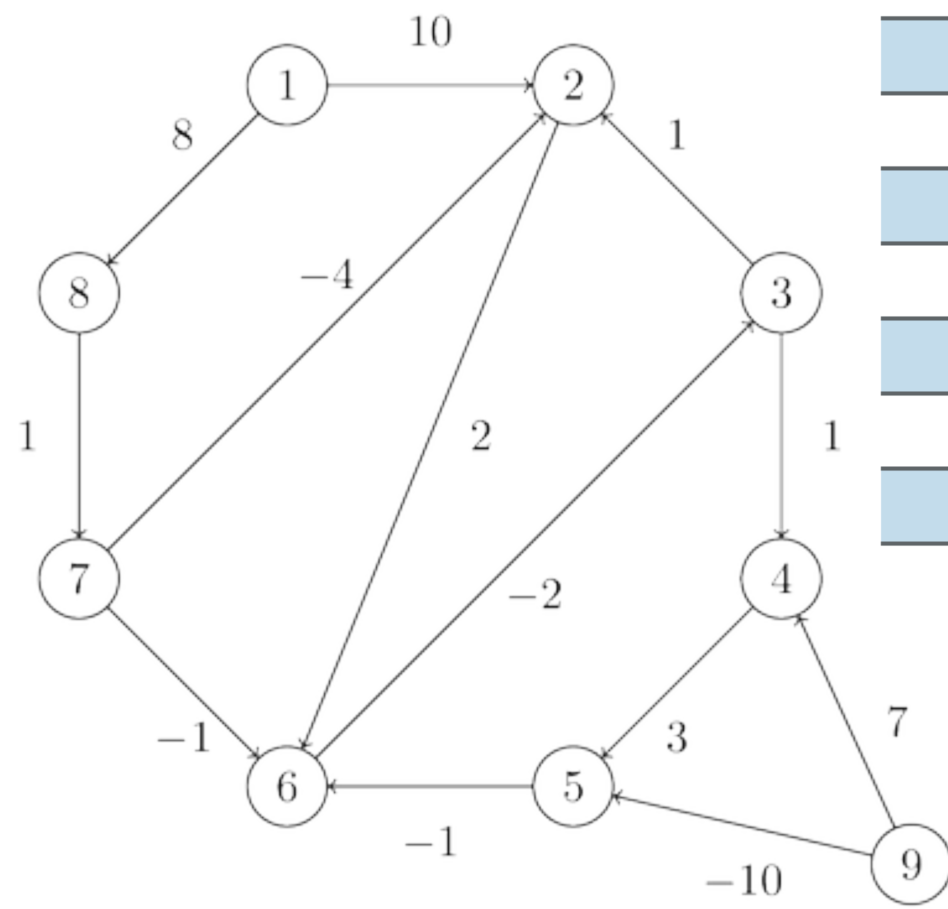


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Practice Time - No changes in passes 3-8

1 2 10
3 2 1
3 4 1
4 5 3
5 6 -1
7 6 -1
8 7 1
1 8 8
7 2 -4
2 6 2
6 3 -2
9 5 -10
9 4 7

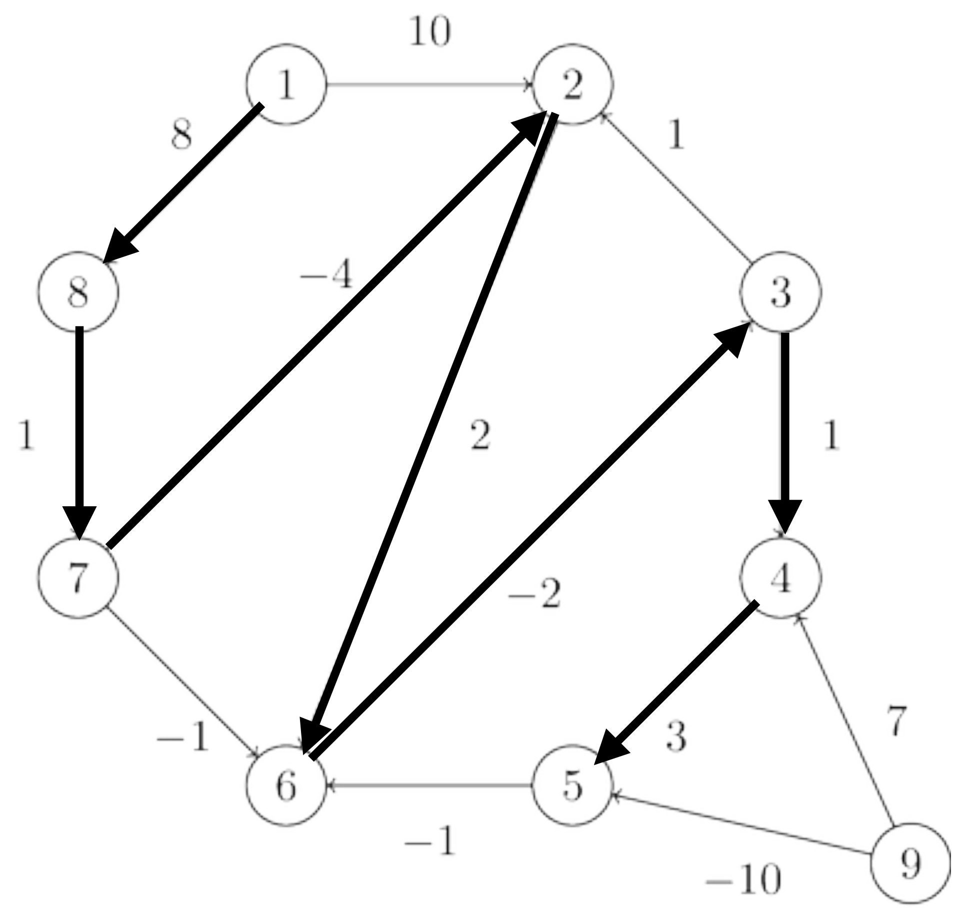


http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7–>2
3	5	6–>3
4	6	3–>4
5	9	4–>5
6	7	2–>6
7	9	8–>7
8	8	1–>8
9	Inf	null

Answer

1	2	10
3	2	1
3	4	1
4	5	3
5	6	-1
7	6	-1
8	7	1
1	8	8
7	2	-4
2	6	2
6	3	-2
9	5	-10
9	4	7



http://rosalind.info/problems/bf/

v	distTo[]	edgeTo[]
1	0	-
2	5	7->2
3	5	6->3
4	6	3->4
5	9	4->5
6	7	2->6
7	9	8->7
8	8	1->8
9	Inf	null

Lecture 27: Shortest Paths

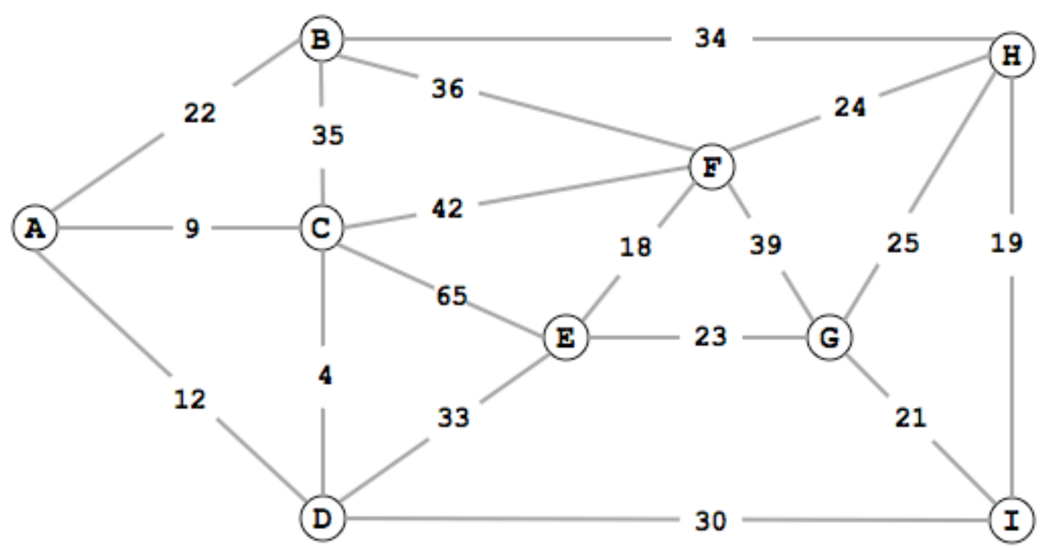
- Introduction to Shortest Paths
- API
- Properties
- Dijkstra's Algorithm
- Belman-Ford Algorithm

Readings:

- Textbook: Chapter 4.4 (Pages 638-676)
- Website:
  - https://algs4.cs.princeton.edu/44sp/

Practice Problems:

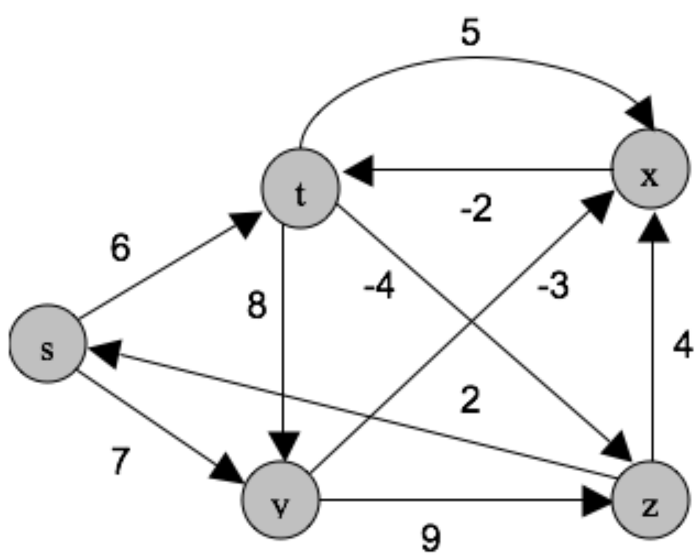
Run Dijkstra's algorithm on the graph on the right with A being the starting vertex.



Practice Problems:

Run the Bellman-Ford algorithm on this directed graph using vertex z as the source. In each pass show the values d and pi. In the graph, V = (s, t, v, x, z} and the weighted, directed edges are E = {(s, t, 6), (s, v, 7), (t, v, 8), (t, z, -4), (t, x, 5), (v, x, -3), (v, z, 9), (x, t, -2), (z, s, 2), (z, x, 4)}.

https://www.chegg.com/homework-help/questions-and-answers/run-bellman-ford-algorithm-following-directed-graph-using-vertex-z-source-pass-show-values-q17182493



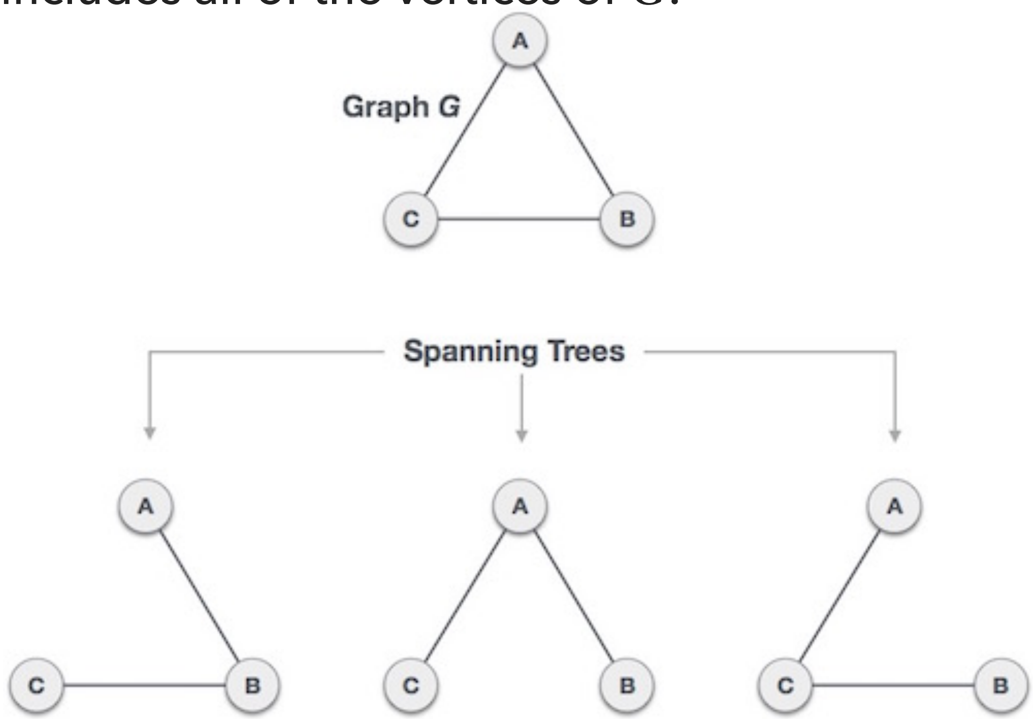
Lecture 28: Minimum Spanning Trees

- Introduction
- Kruskal's Algorithm
- Prim's Algorithm

Some slides adopted from Algorithms 4th Edition or COS226

Spanning Trees

- Given an edge weighted graph G (not digraph!), a spanning tree of G is a subgraph T that is:
  - A tree: connected and acyclic.
  - Spanning: includes all of the vertices of G.



https://www.tutorialspoint.com/data_structures_algorithms/spanning_tree.htm

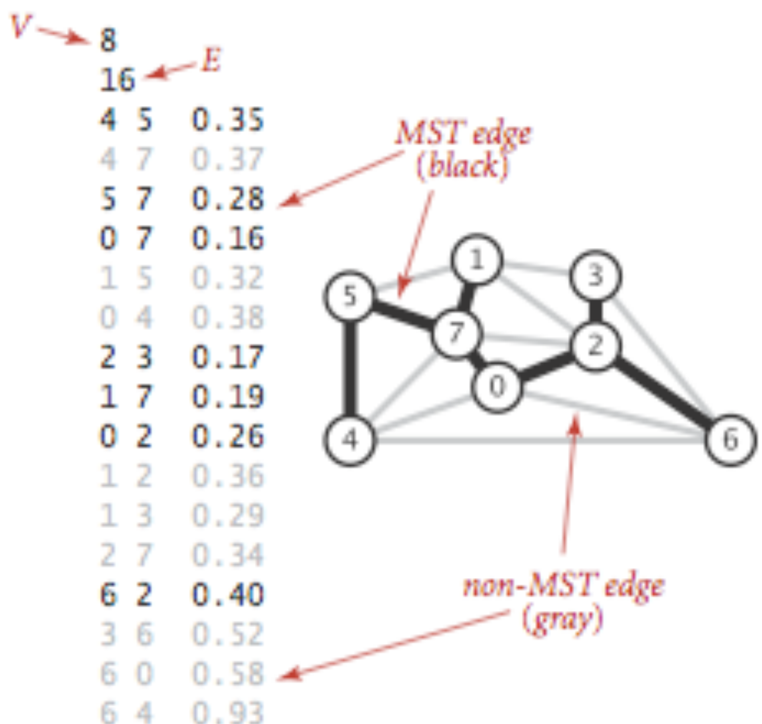
Properties

- A connected graph G can have more than one spanning tree.
- All possible spanning trees of G have the same number of vertices and edges.
- A spanning tree has $|V| - 1$ edges.
- A spanning tree by definition cannot have any cycle.
- Adding one edge to the spanning tree would create a cycle (i.e. spanning trees are maximally acyclic).
- Removing one edge from the spanning tree would make the graph disconnected (i.e. spanning trees are minimally connected).

https://www.tutorialspoint.com/data_structures_algorithms/spanning_tree.htm

Minimum spanning tree problem

- Given a connected edge-weighted undirected graph find a spanning tree of minimum weight.



An edge-weighted graph and its MST

Minimum spanning applications

- Network design
- Cluster analysis
- Cancer imaging
- Cosmology
- Weather data interpretation
- Many others
  - https://www.ics.uci.edu/~eppstein/gina/mst.html
  - https://personal.utdallas.edu/~besp/teaching/mst-applications.pdf

Lecture 28: Minimum Spanning Trees

- Introduction
- Kruskal's Algorithm
- Prim's Algorithm

Kruskal's algorithm

- Sort edges in ascending order of weight.
- Starting from the one with the smallest weight, add it to the MST T unless doing so would create a cycle.

- Uses a data structure called Union-Find (Chapter 1.5 in book).
- Running time of $|E|\log|V|$ in worst case.

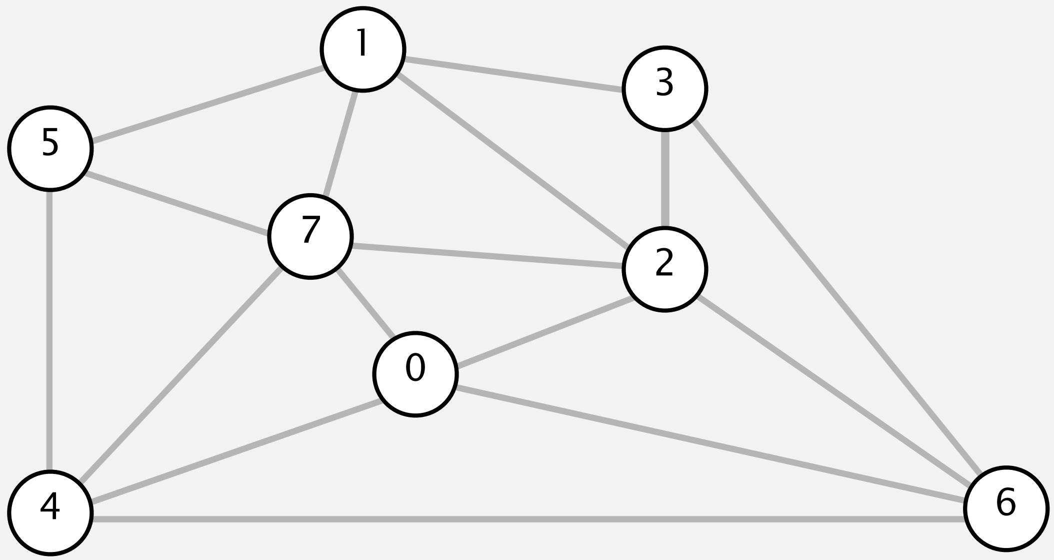
http://algs4.cs.princeton.edu

Kruskal's Algorithm Demo

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



an edge-weighted graph

graph edges sorted by weight

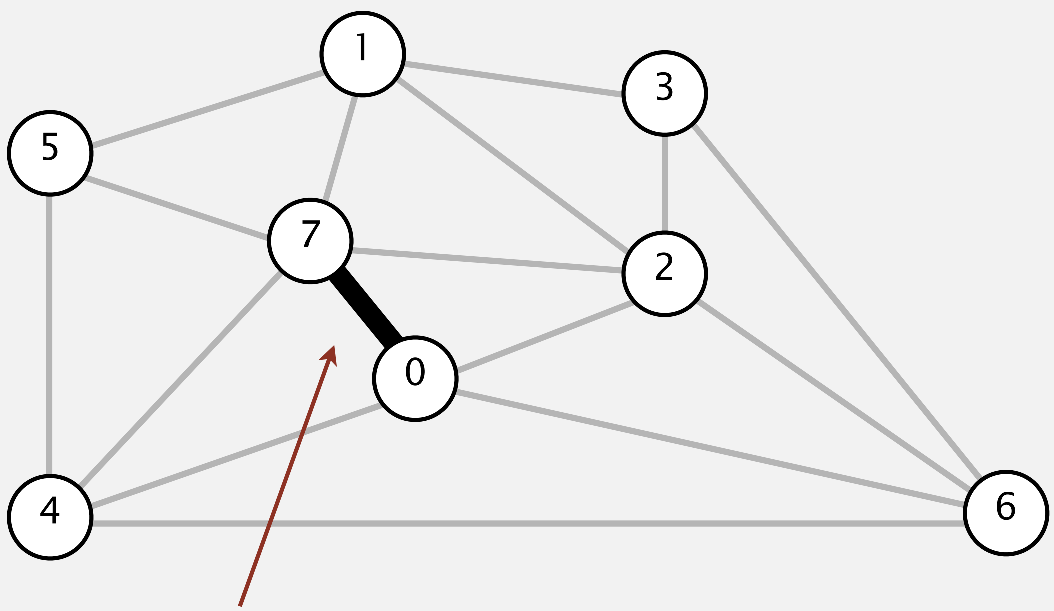
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.

in MST → 0-7 0.16

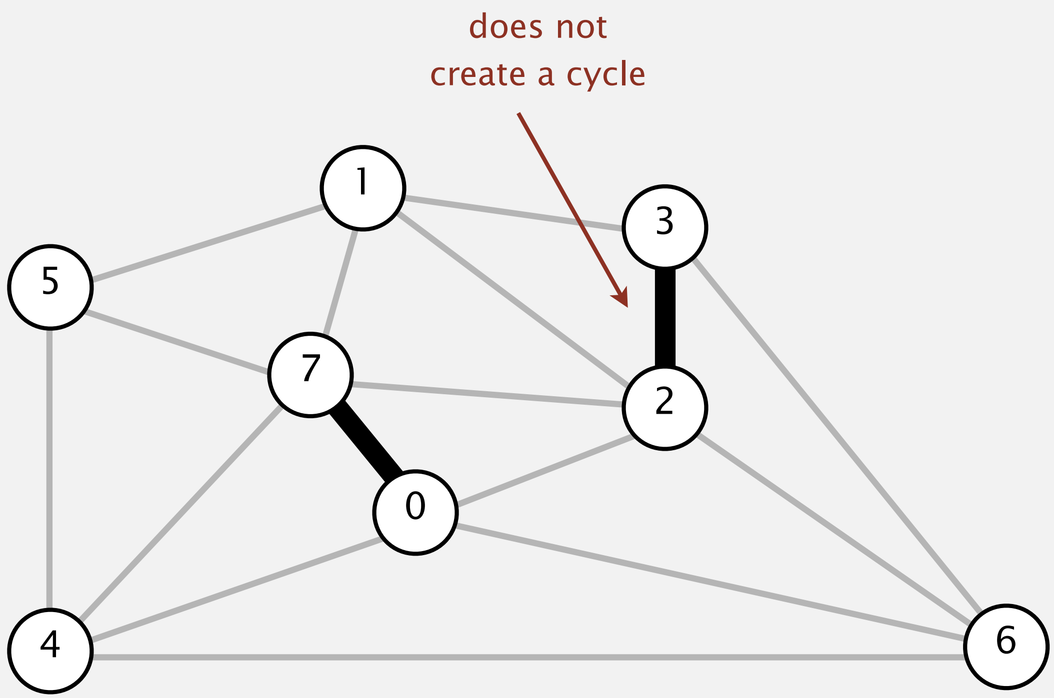


does not create a cycle

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.

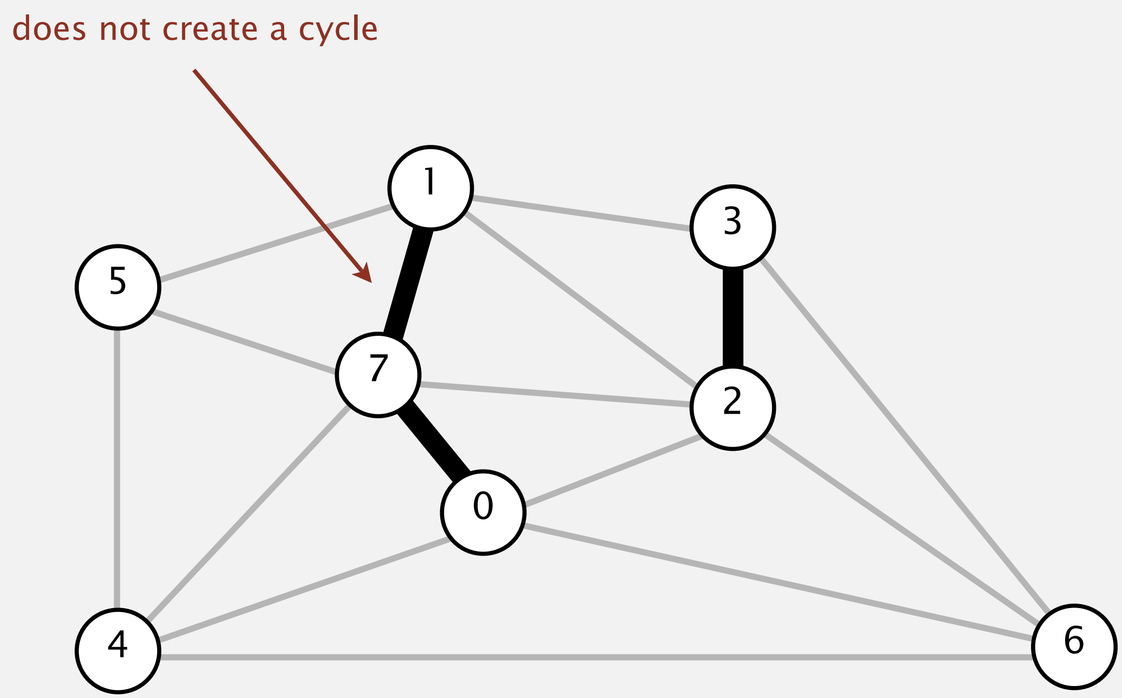


	0-7	0.16
in MST →	2-3	0.17

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



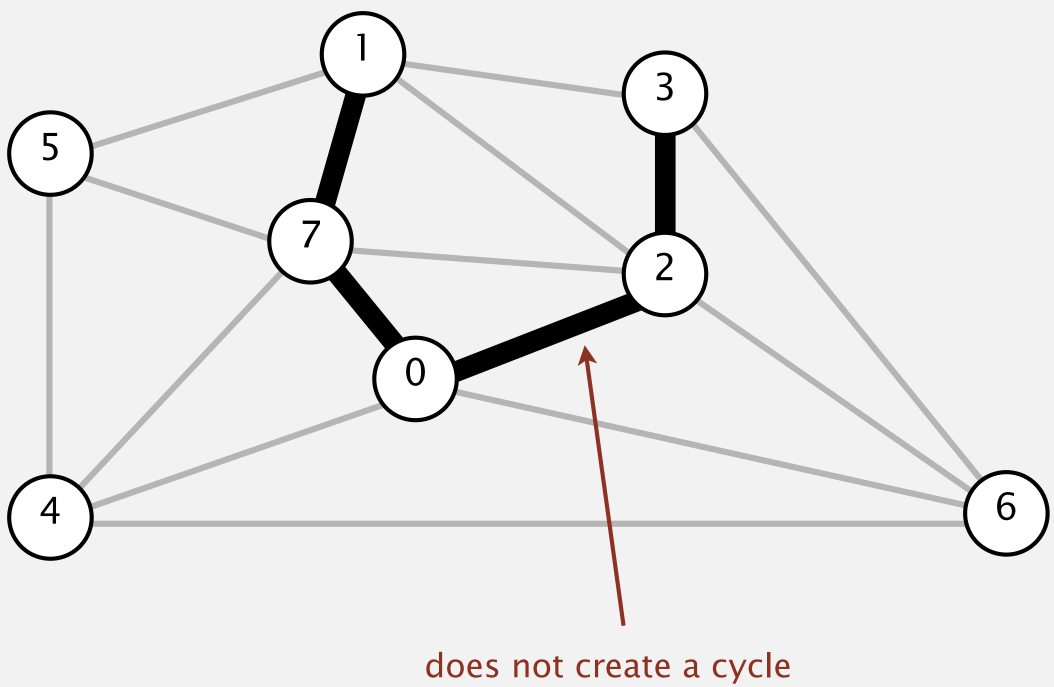
in MST →

0-7	0.16
2-3	0.17
1-7	0.19

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



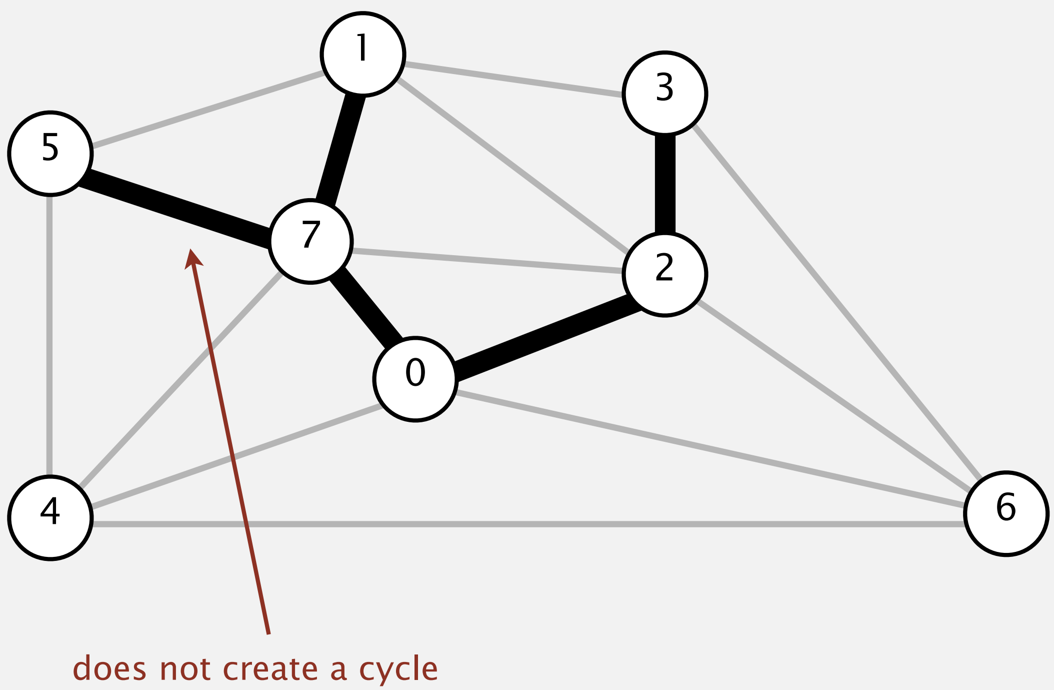
in MST →

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



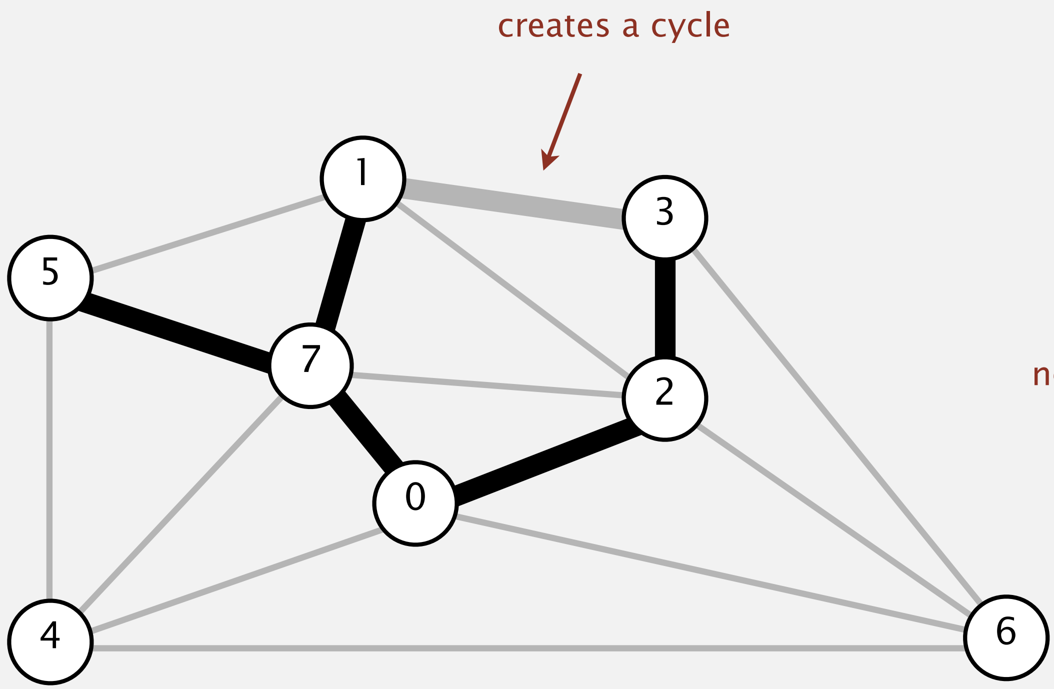
in MST →

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



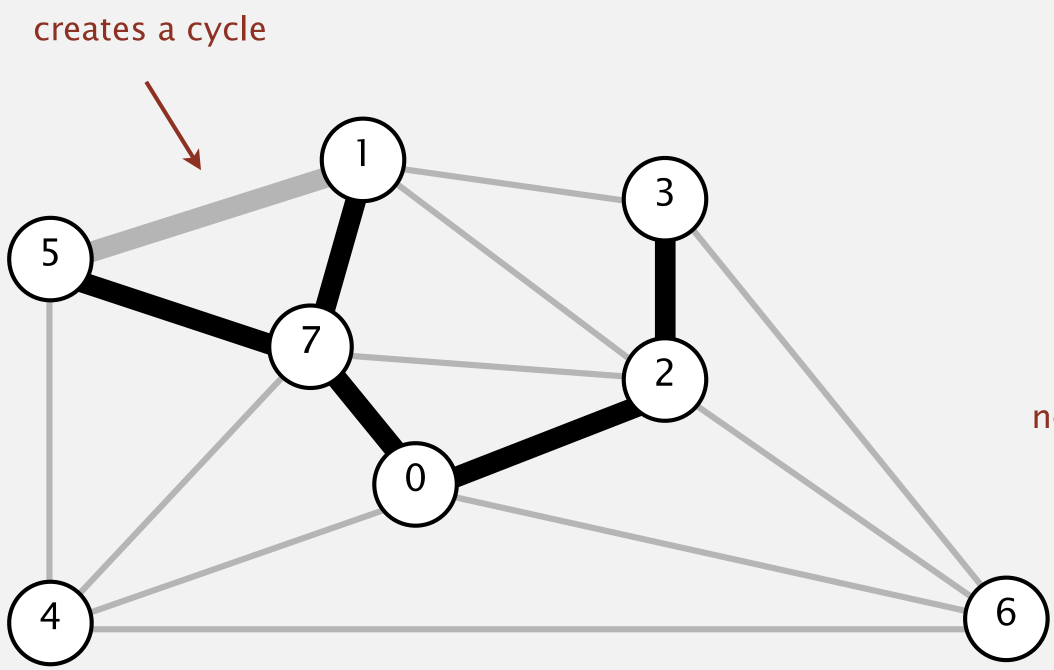
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29

not in MST →

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



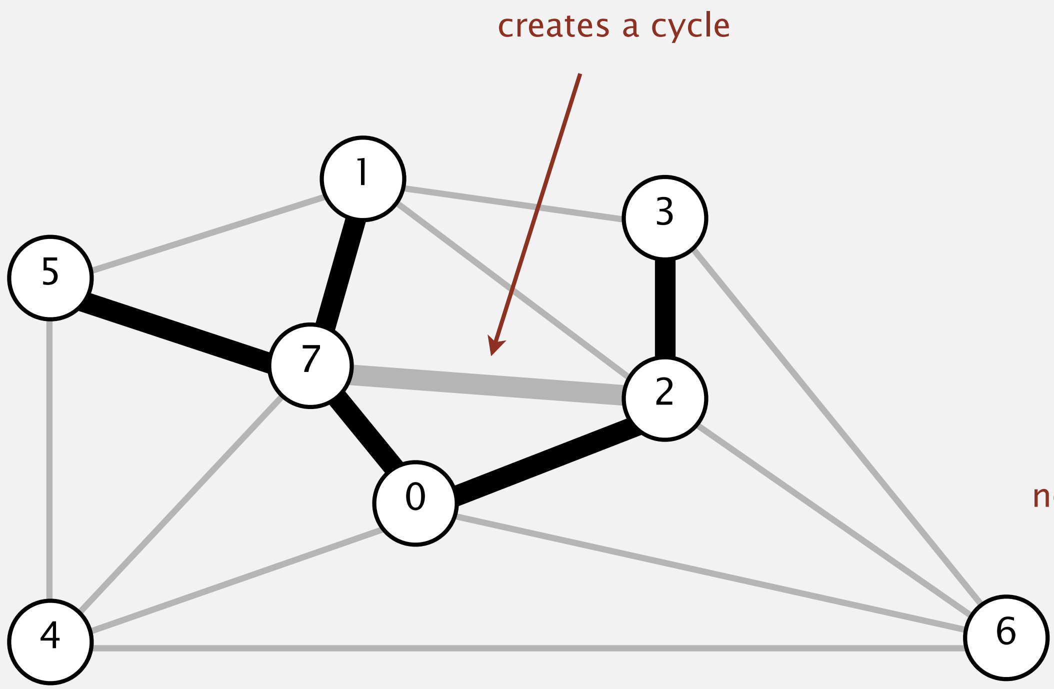
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32

not in MST →

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



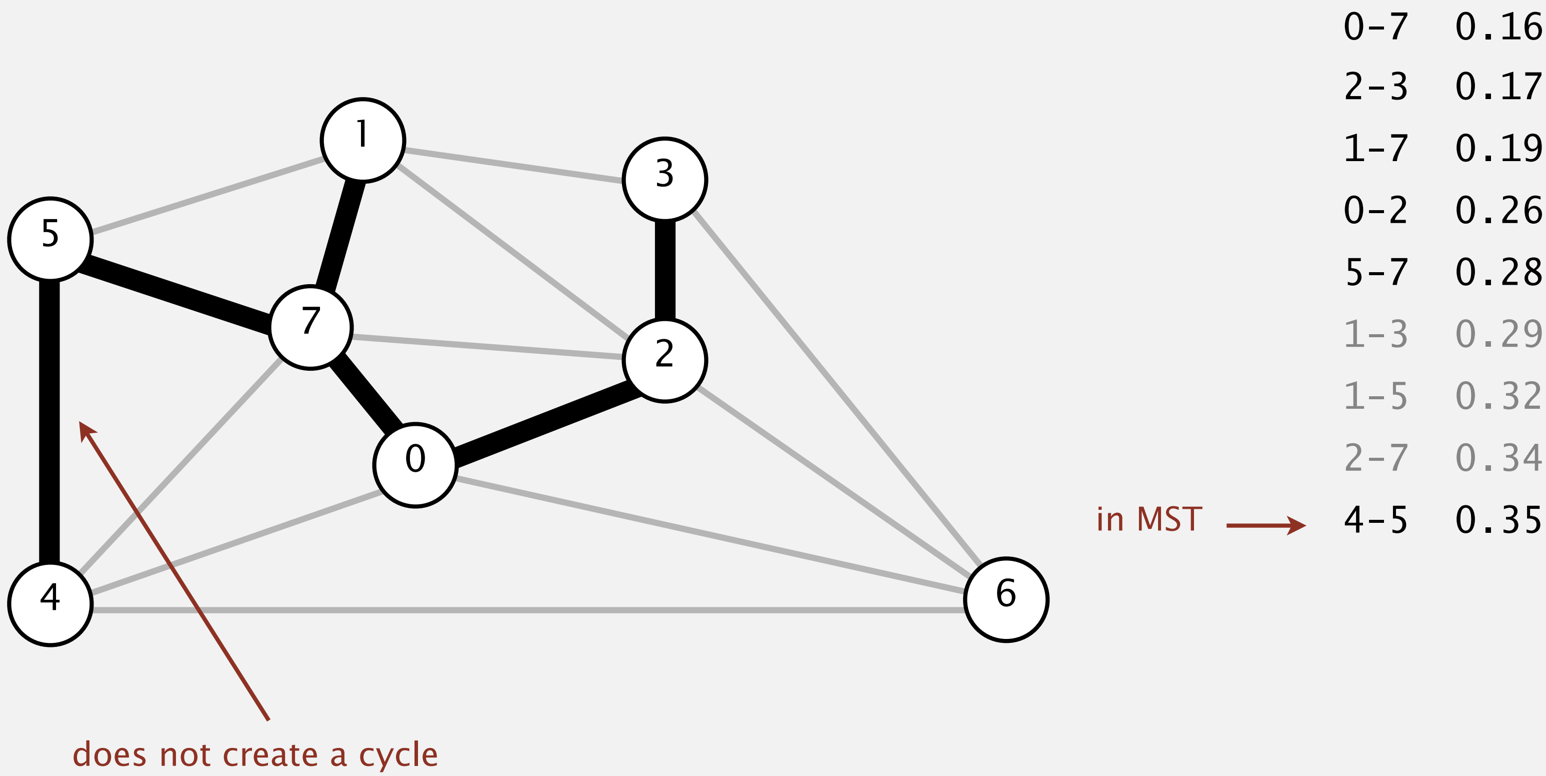
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34

not in MST → 2-7

Kruskal's algorithm demo

Consider edges in ascending order of weight.

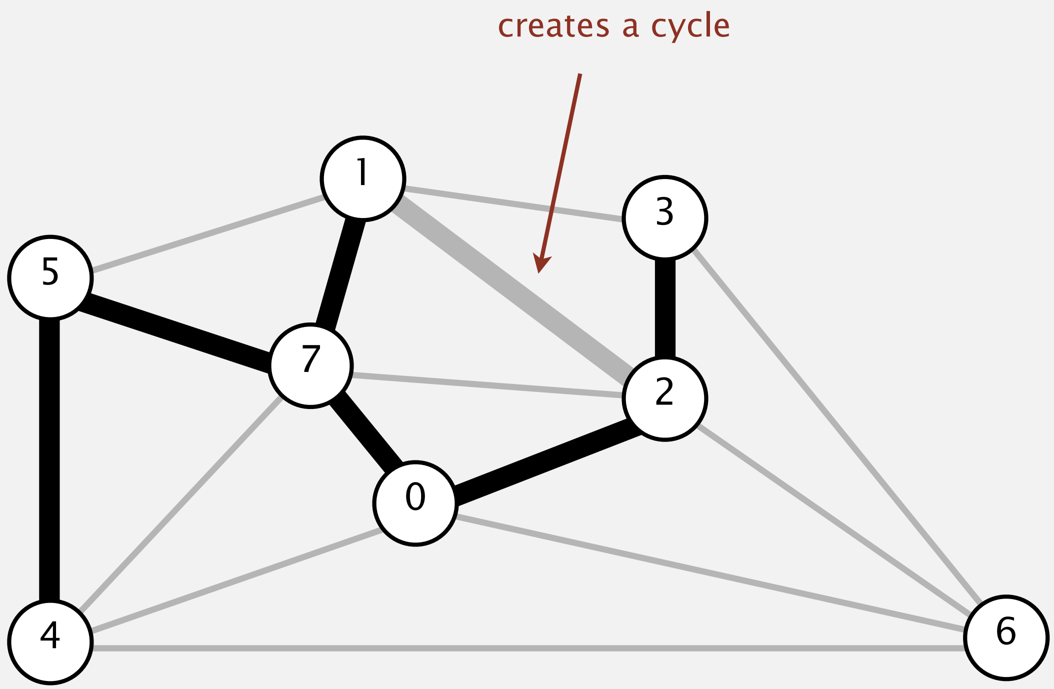
- Add next edge to tree T unless doing so would create a cycle.



Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



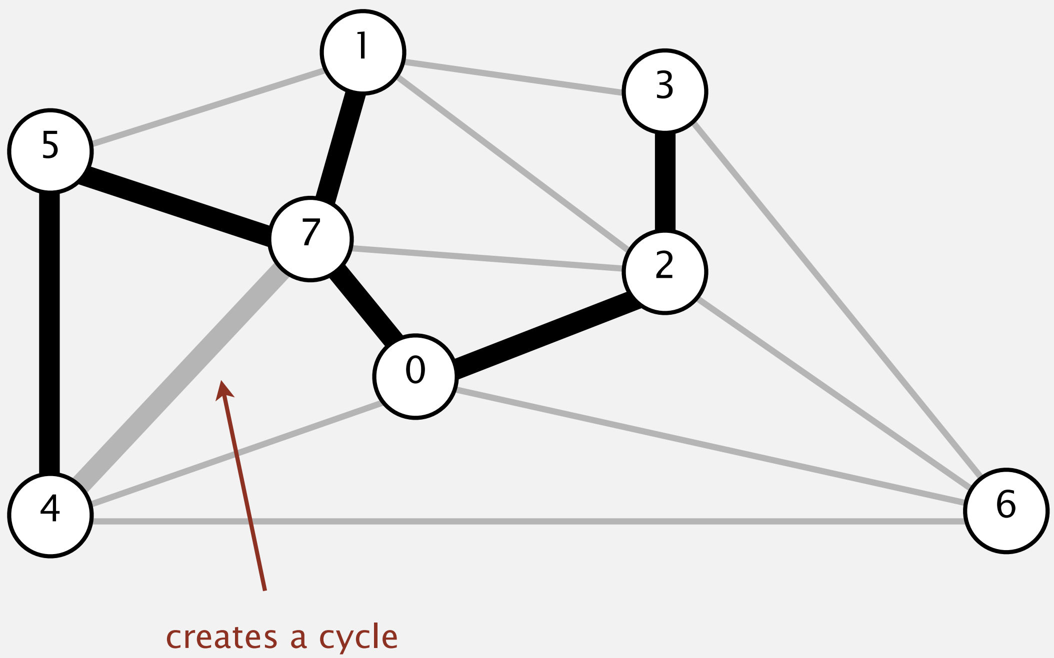
not in MST →

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



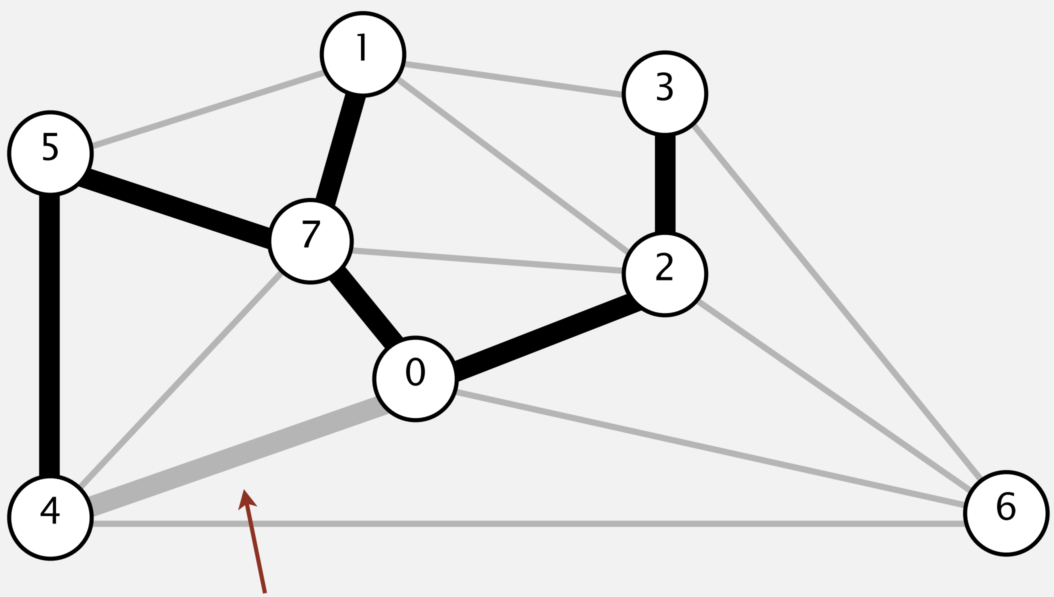
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37

not in MST

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



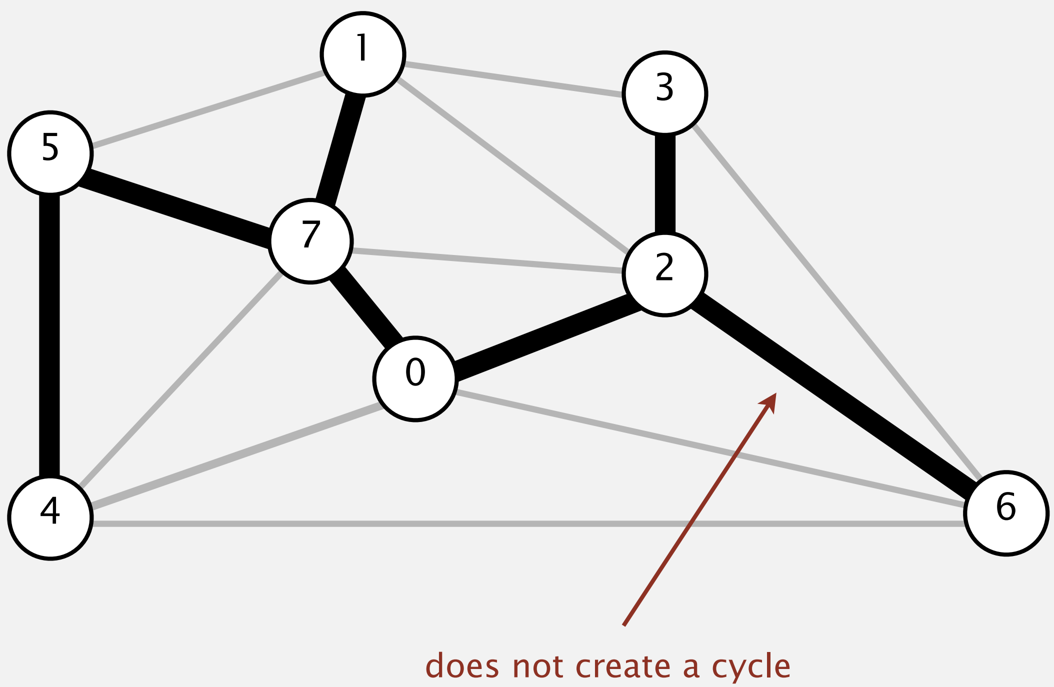
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38

not in MST

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.

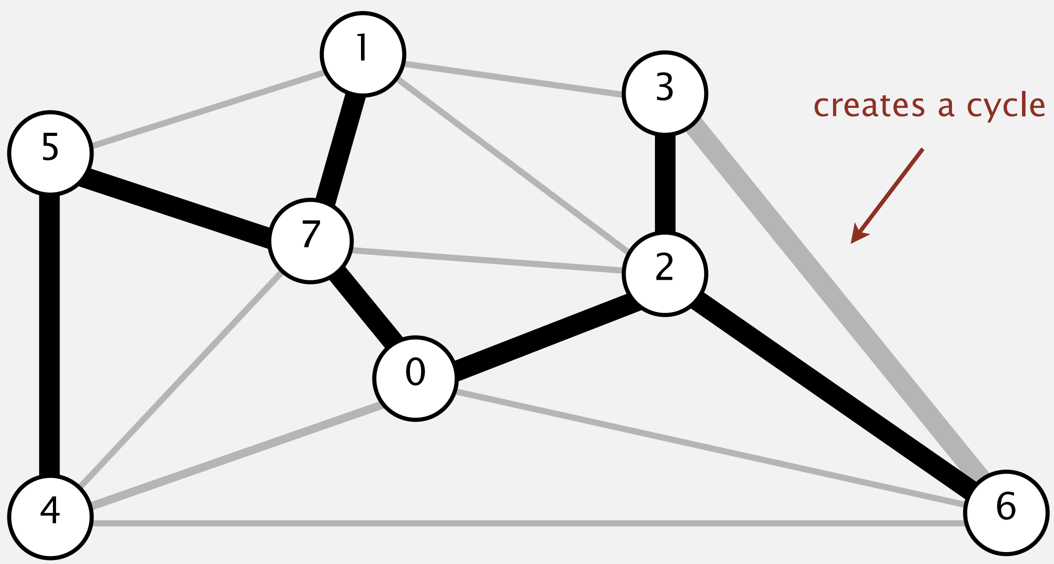


	0-7	0.16
	2-3	0.17
	1-7	0.19
	0-2	0.26
	5-7	0.28
	1-3	0.29
	1-5	0.32
	2-7	0.34
	4-5	0.35
	1-2	0.36
	4-7	0.37
	0-4	0.38
in MST →	6-2	0.40

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.

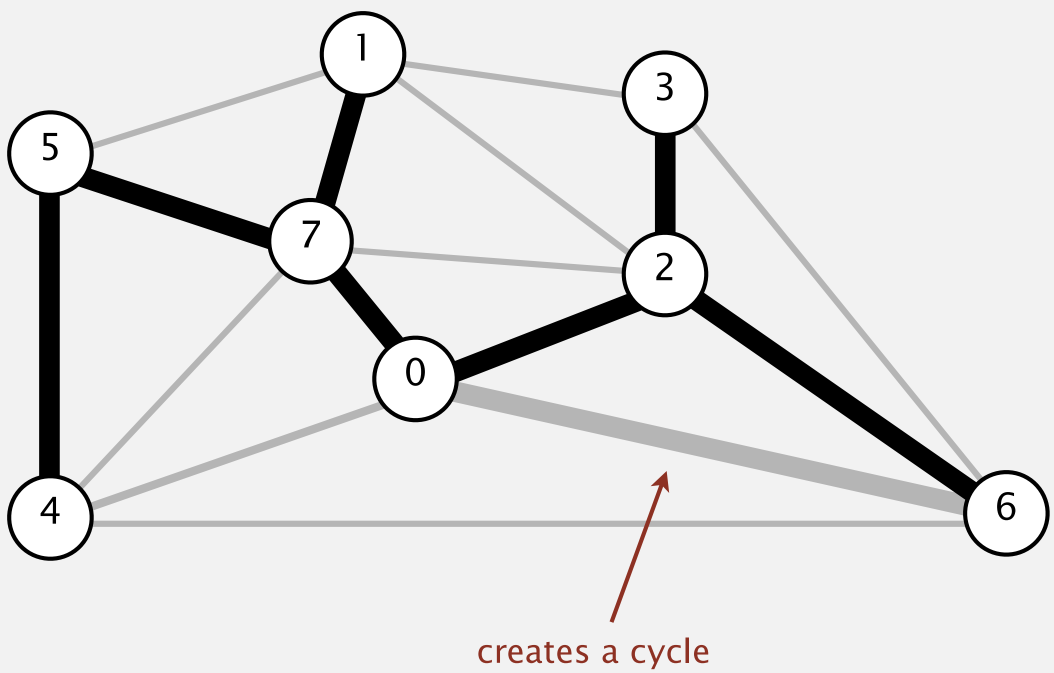


0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



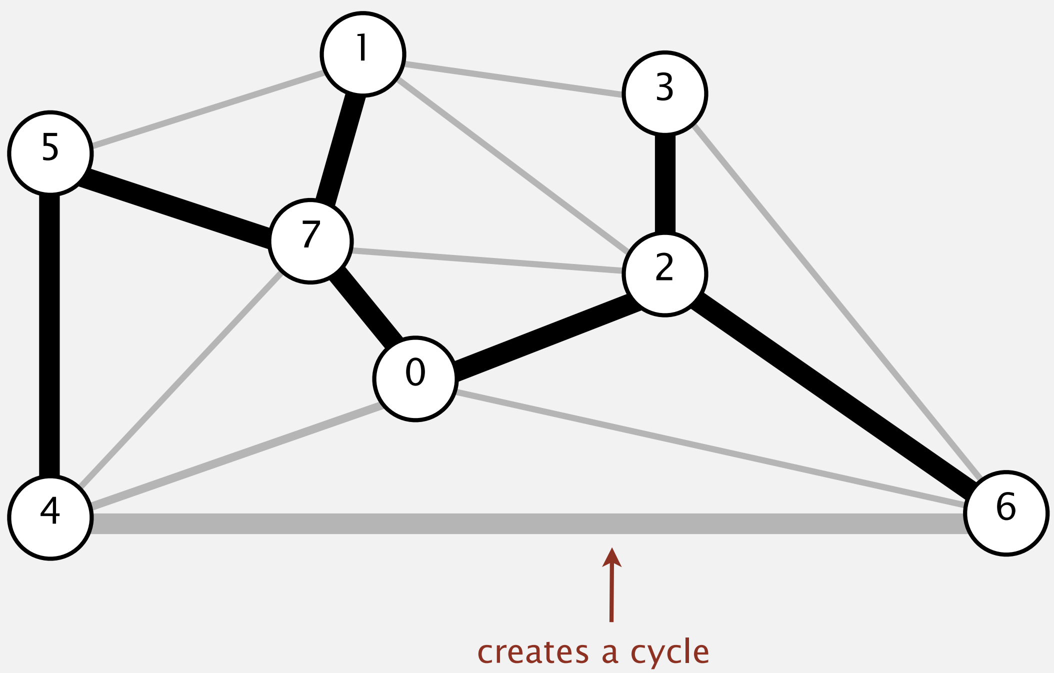
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

not in MST →

Kruskal's algorithm demo

Consider edges in ascending order of weight.

- Add next edge to tree T unless doing so would create a cycle.



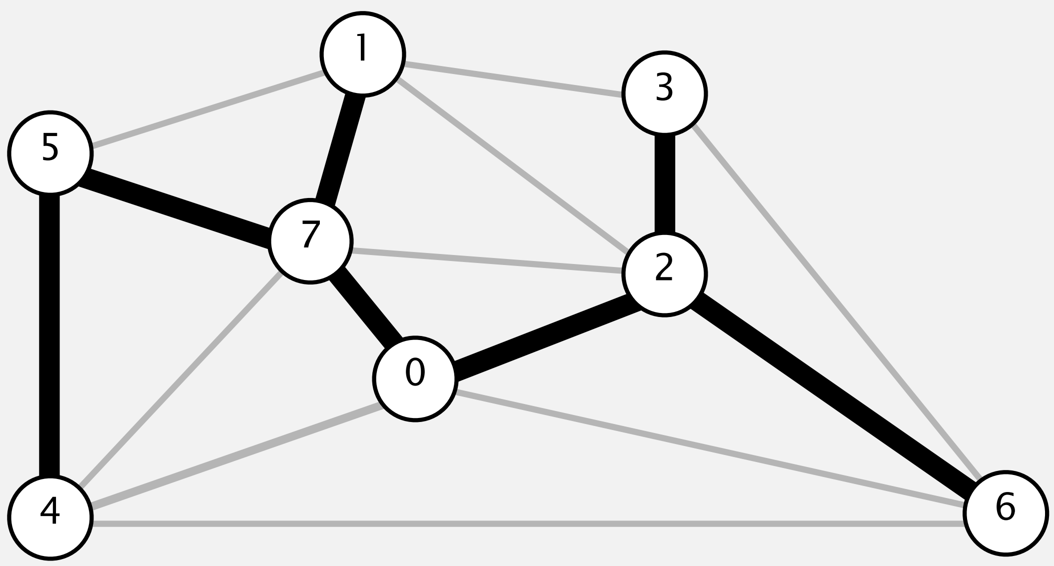
0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

not in MST → 6-4

Kruskal's algorithm demo

Consider edges in ascending order of weight.

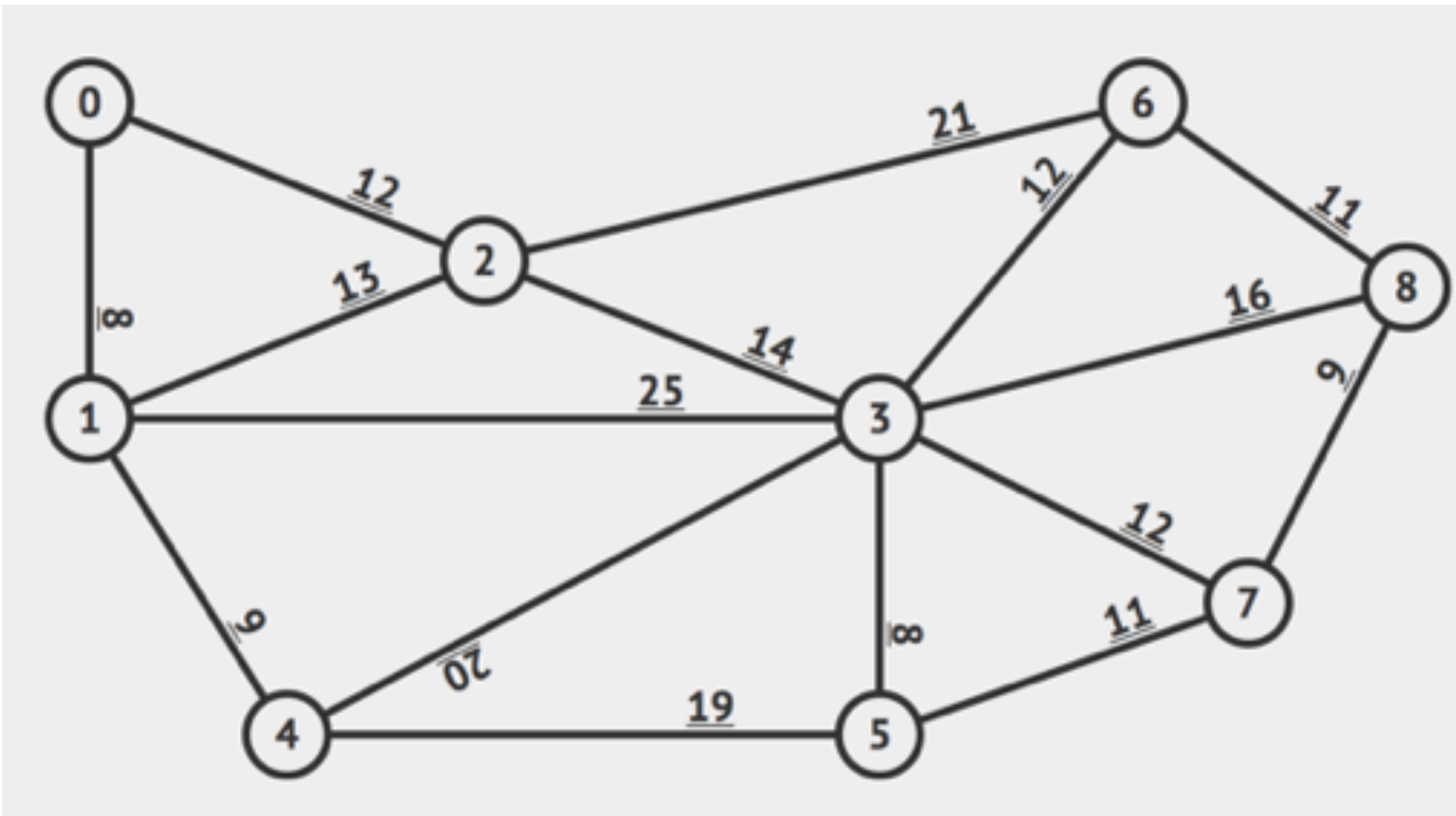
- Add next edge to tree T unless doing so would create a cycle.



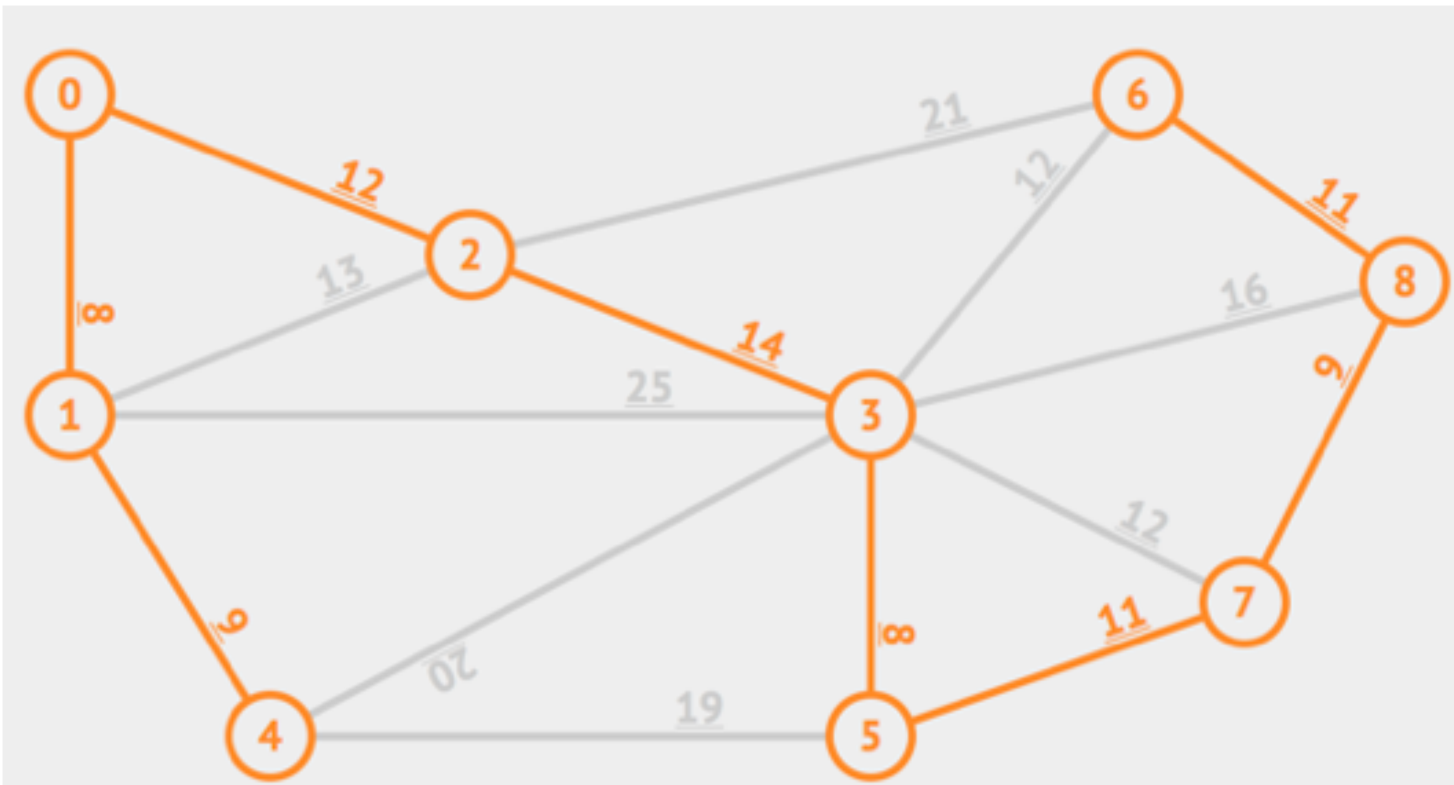
a minimum spanning tree

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

Practice Time



Answer



Lecture 28: Minimum Spanning Trees

- Introduction
- Kruskal's Algorithm
- Prim's Algorithm

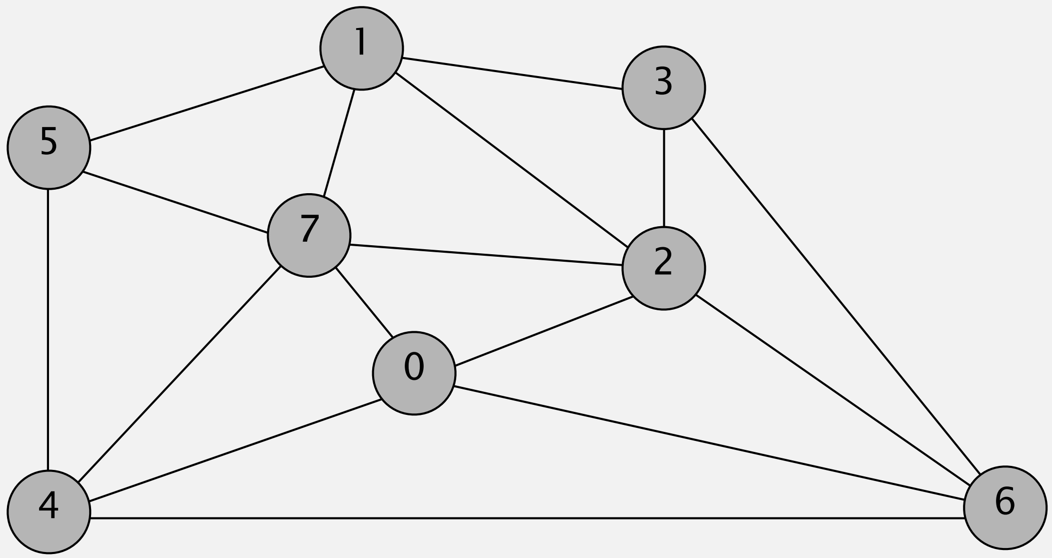
Prim's algorithm

- Start with a random vertex (here, 0) and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $|V| - 1$ edges.

- Two versions, lazy and eager. We will see lazy, here…
- Uses min-priority queue.
- Running time of $|E| \log |V|$ in worst case, as well.

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.

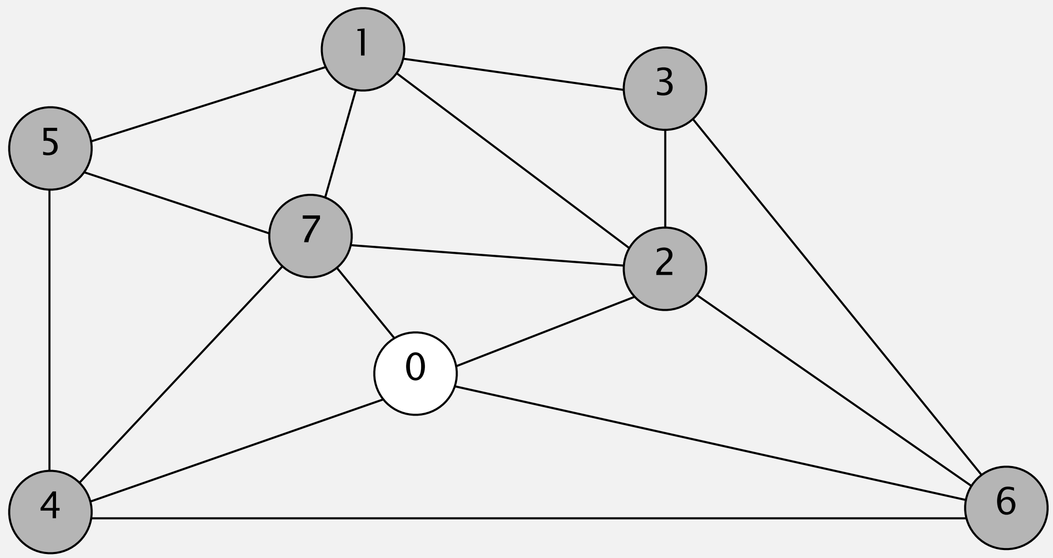


an edge-weighted graph

0-7	0.16
2-3	0.17
1-7	0.19
0-2	0.26
5-7	0.28
1-3	0.29
1-5	0.32
2-7	0.34
4-5	0.35
1-2	0.36
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58
6-4	0.93

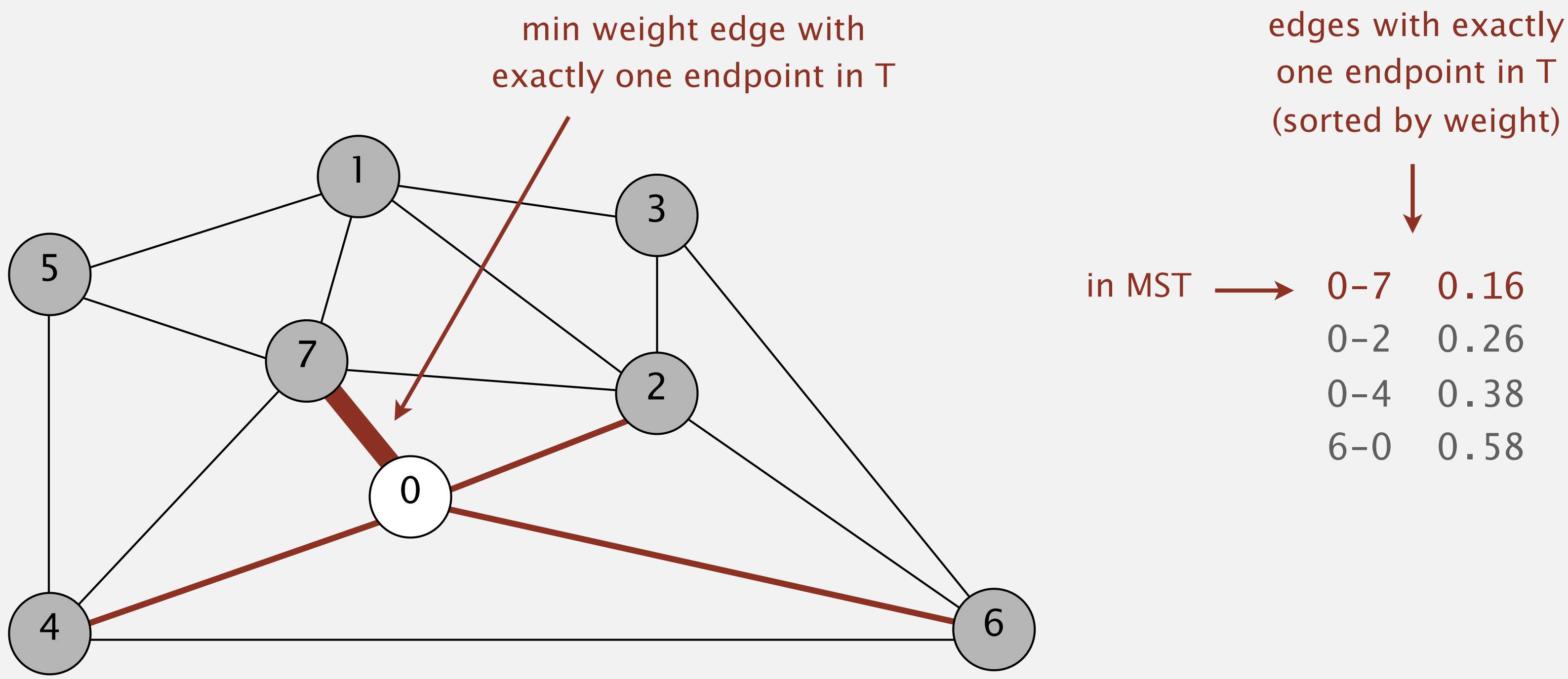
Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



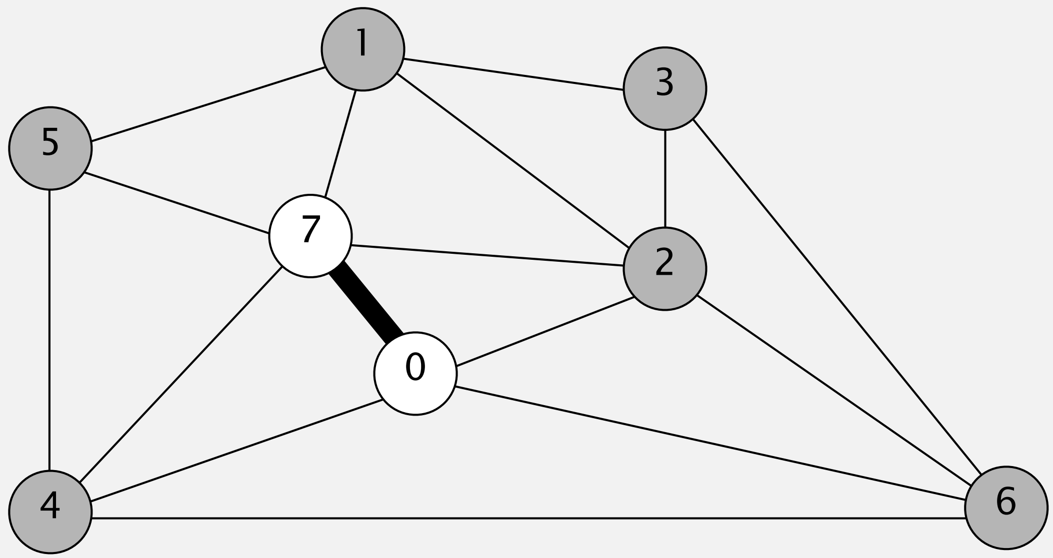
Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.

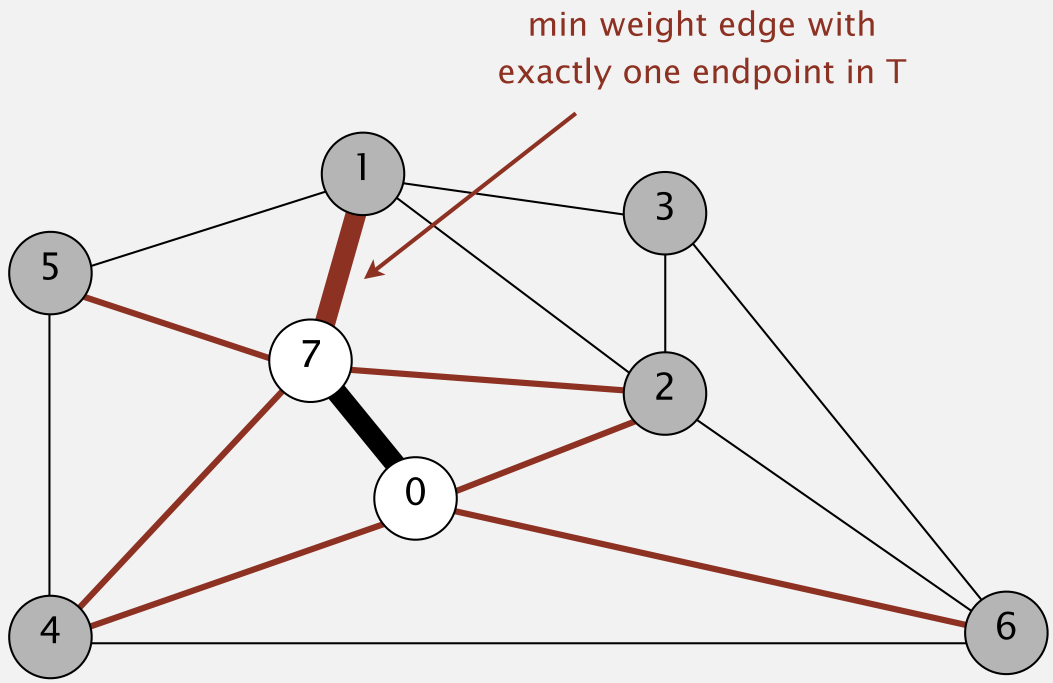


MST edges

0–7

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



edges with exactly one endpoint in T (sorted by weight)

in MST →

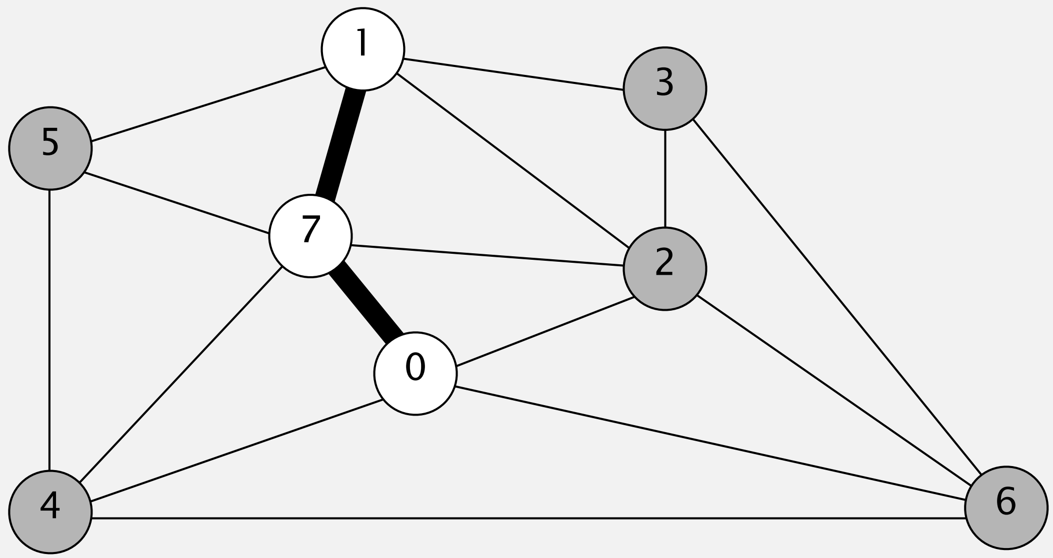
1-7	0.19
0-2	0.26
5-7	0.28
2-7	0.34
4-7	0.37
0-4	0.38
6-0	0.58

MST edges

0-7

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.

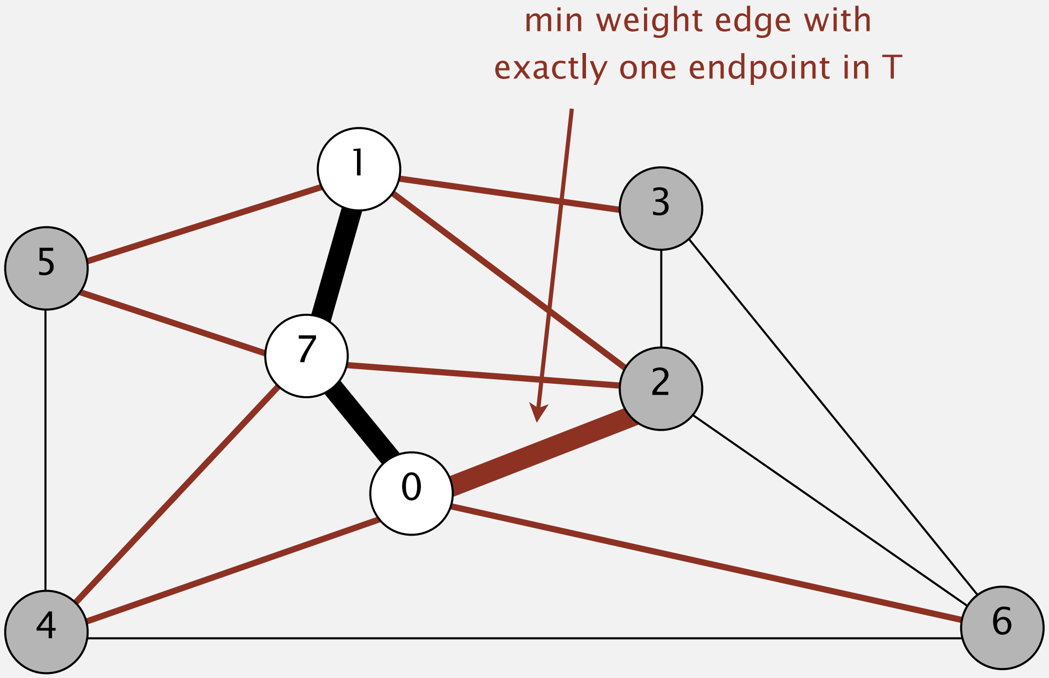


MST edges

0-7 1-7

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



edges with exactly one endpoint in T (sorted by weight)

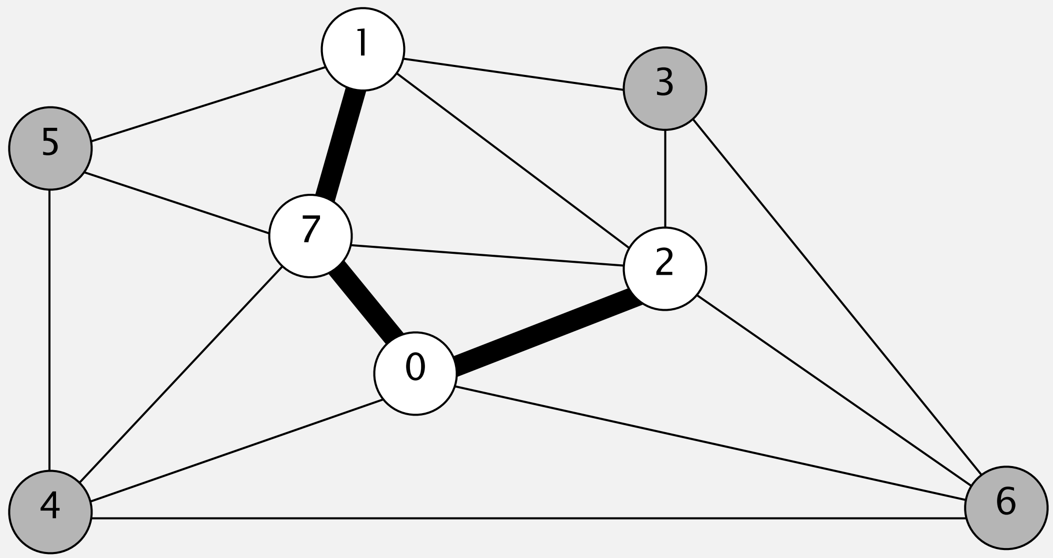
in MST →	0-2	0.26
	5-7	0.28
	1-3	0.29
	1-5	0.32
	2-7	0.34
	1-2	0.36
	4-7	0.37
	0-4	0.38
	6-0	0.58

MST edges

0-7 1-7

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.

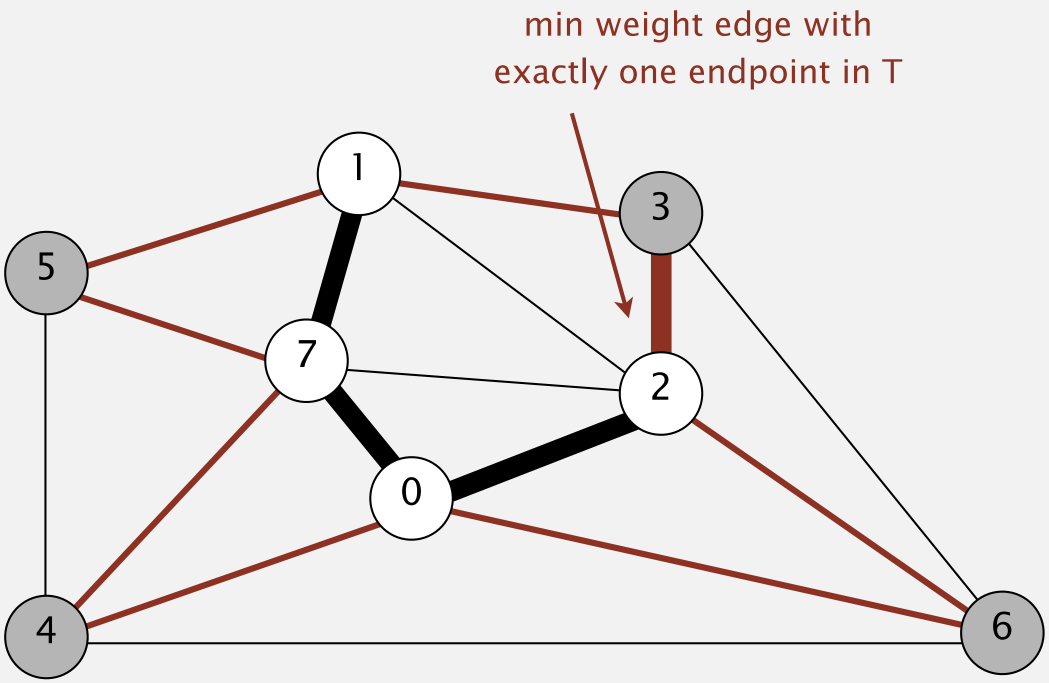


MST edges

0-7   1-7   0-2

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



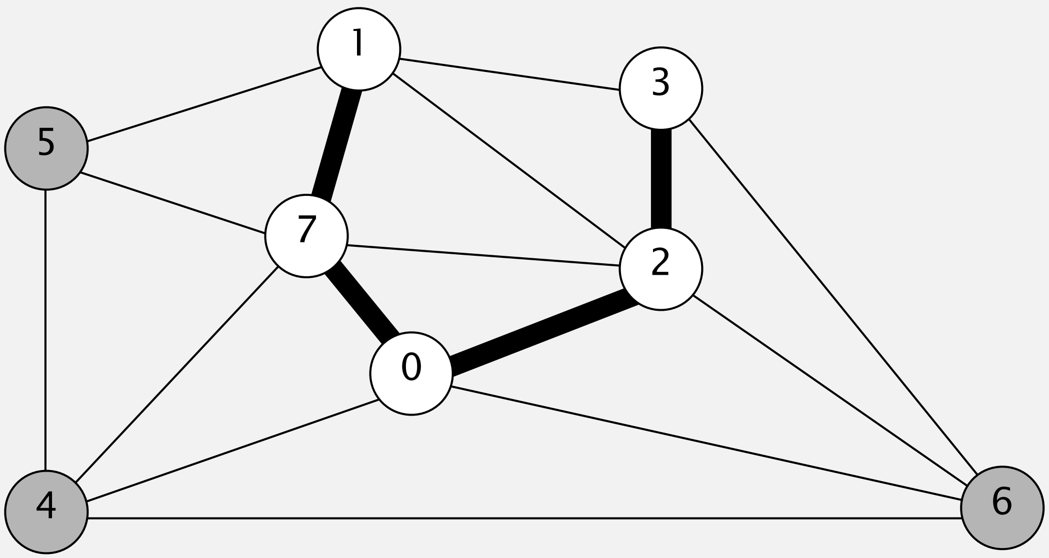
2-3	0.17
5-7	0.28
1-3	0.29
1-5	0.32
4-7	0.37
0-4	0.38
6-2	0.40
6-0	0.58

MST edges

0-7 1-7 0-2

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.

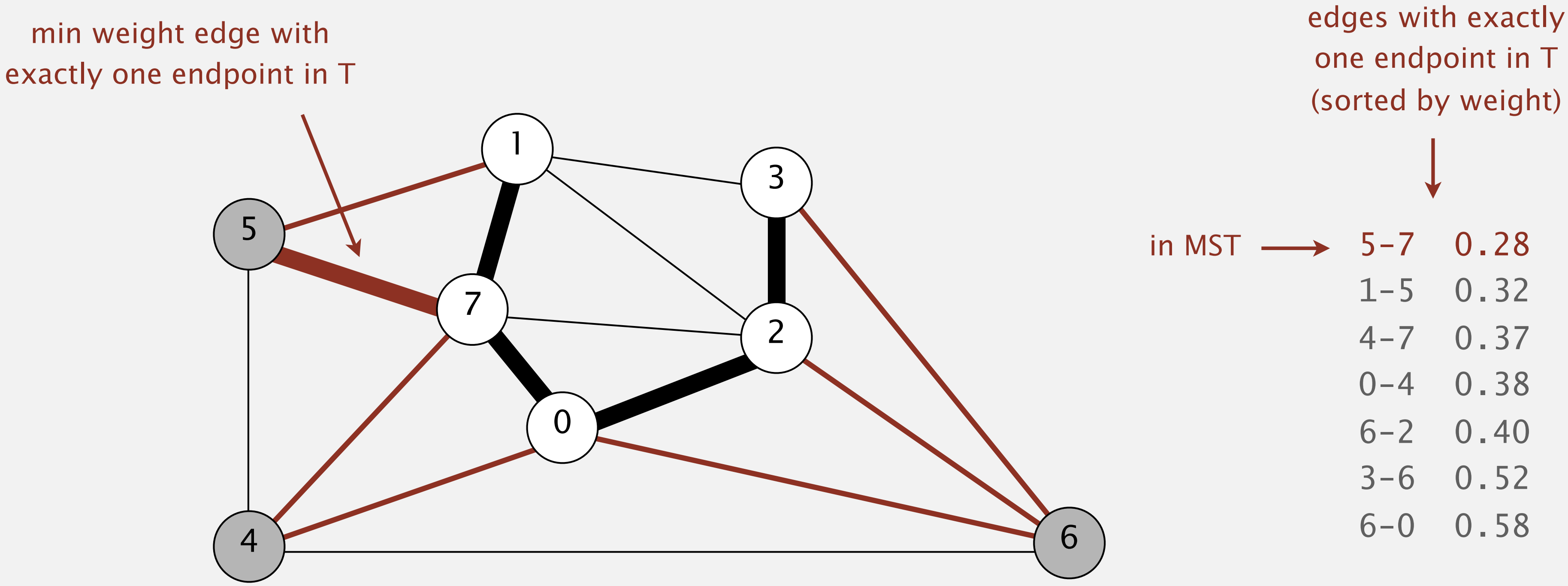


MST edges

0-7 1-7 0-2 2-3

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.

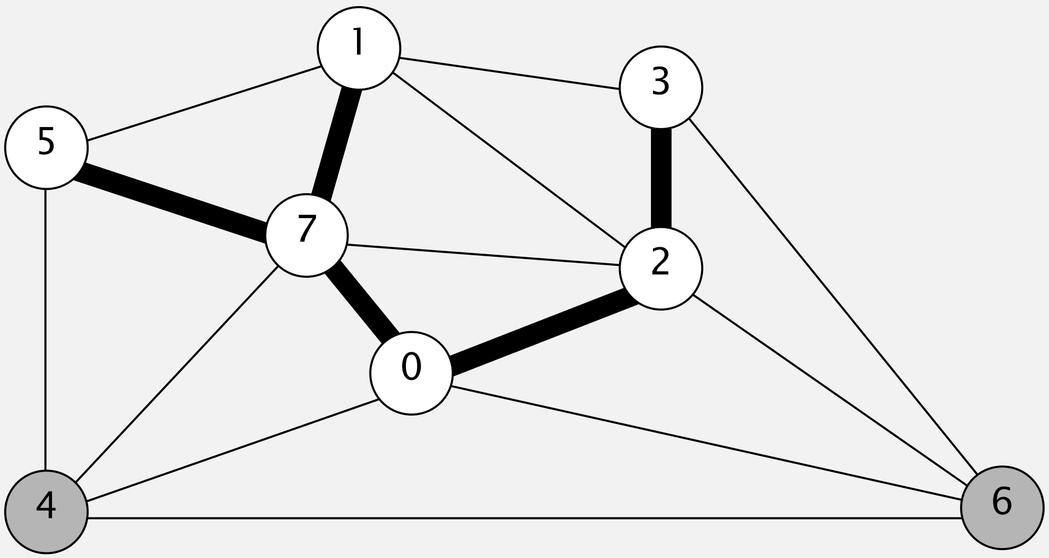


MST edges

0-7 1-7 0-2 2-3

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



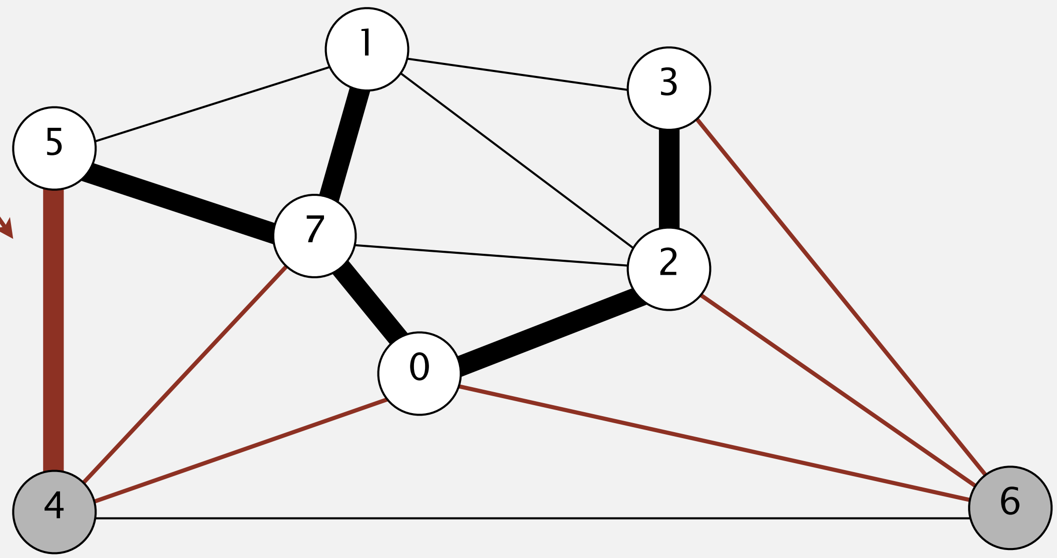
MST edges

0-7 1-7 0-2 2-3 5-7

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.

min weight edge with exactly one endpoint in T



edges with exactly one endpoint in T (sorted by weight)

in MST →

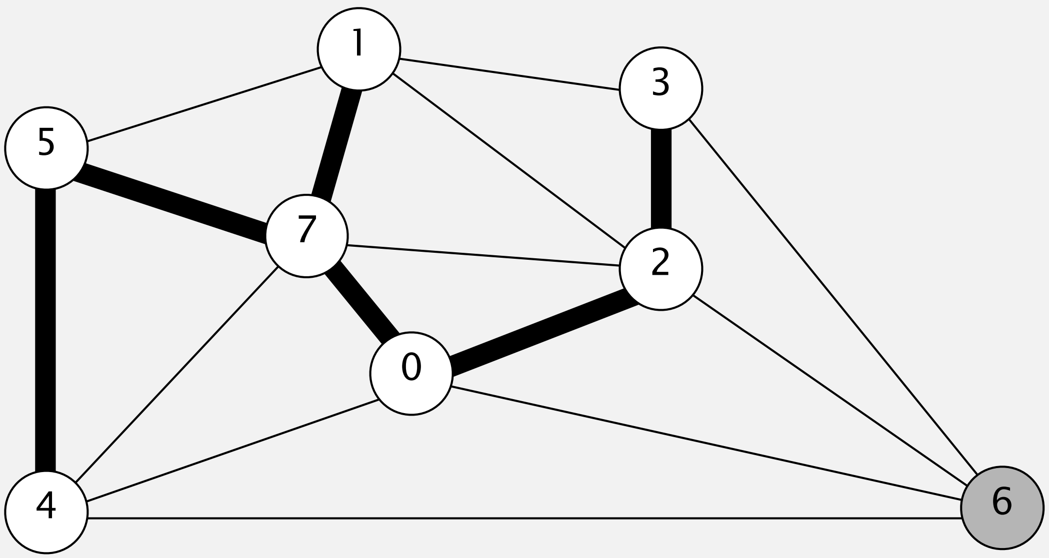
4-5	0.35
4-7	0.37
0-4	0.38
6-2	0.40
3-6	0.52
6-0	0.58

MST edges

0-7 1-7 0-2 2-3 5-7

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.

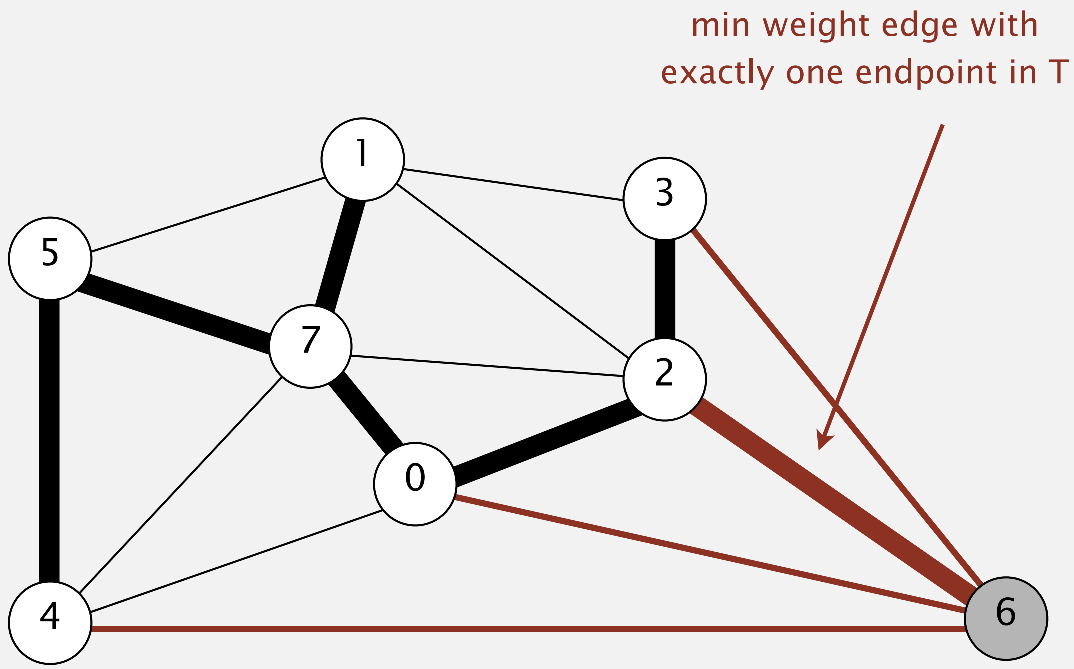


MST edges

0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm demo

- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



edges with exactly
one endpoint in T
(sorted by weight)

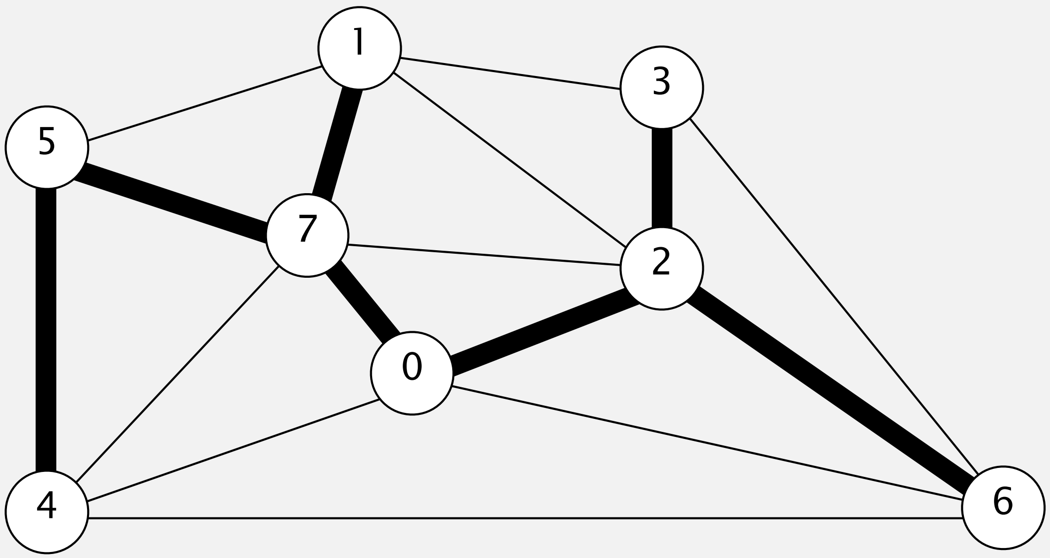
in MST →	6-2	0.40
	3-6	0.52
	6-0	0.58
	6-4	0.93

MST edges

0-7 1-7 0-2 2-3 5-7 4-5

Prim's algorithm demo

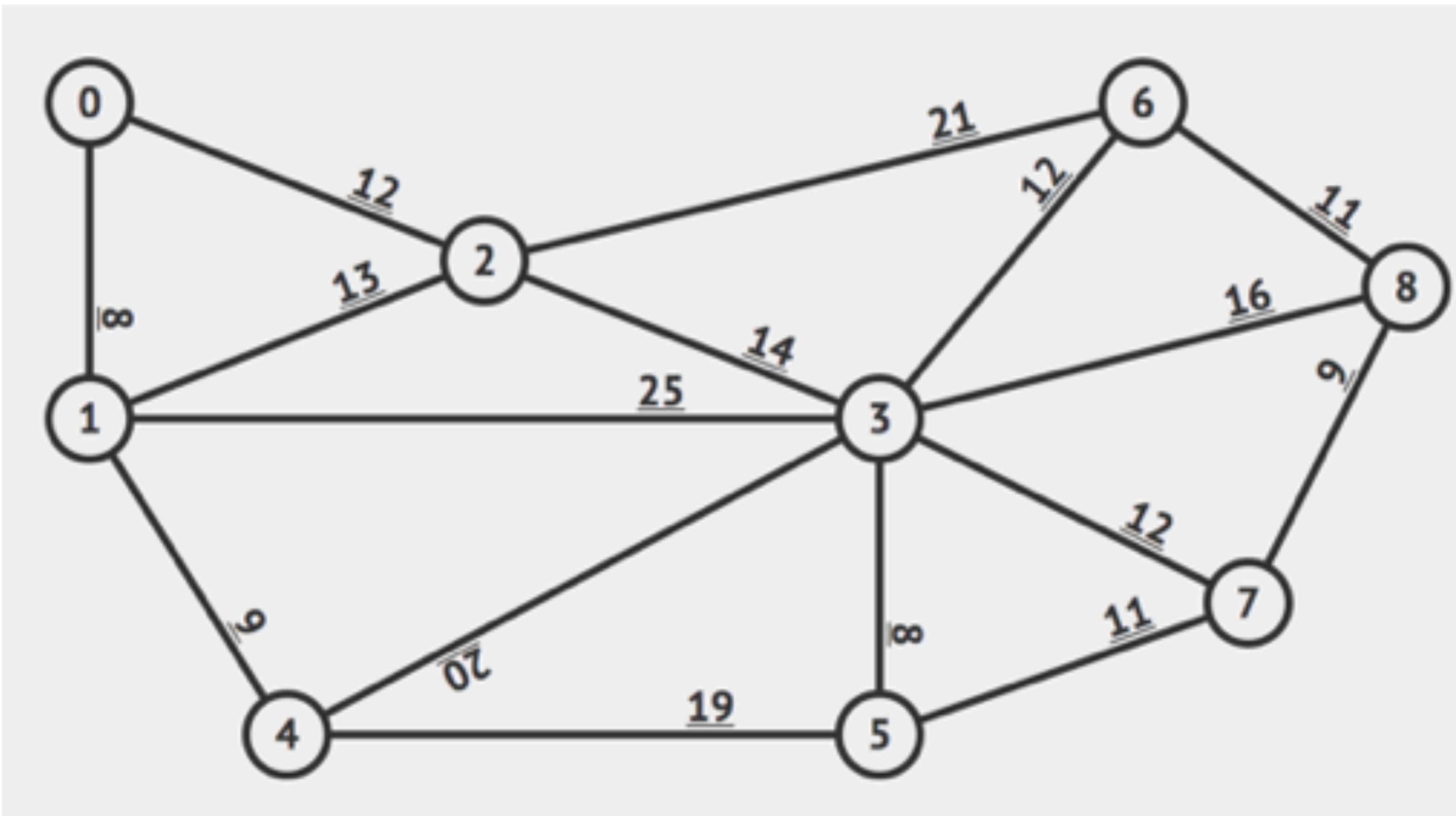
- Start with vertex 0 and greedily grow tree T.
- Add to T the min weight edge with exactly one endpoint in T.
- Repeat until $V - 1$ edges.



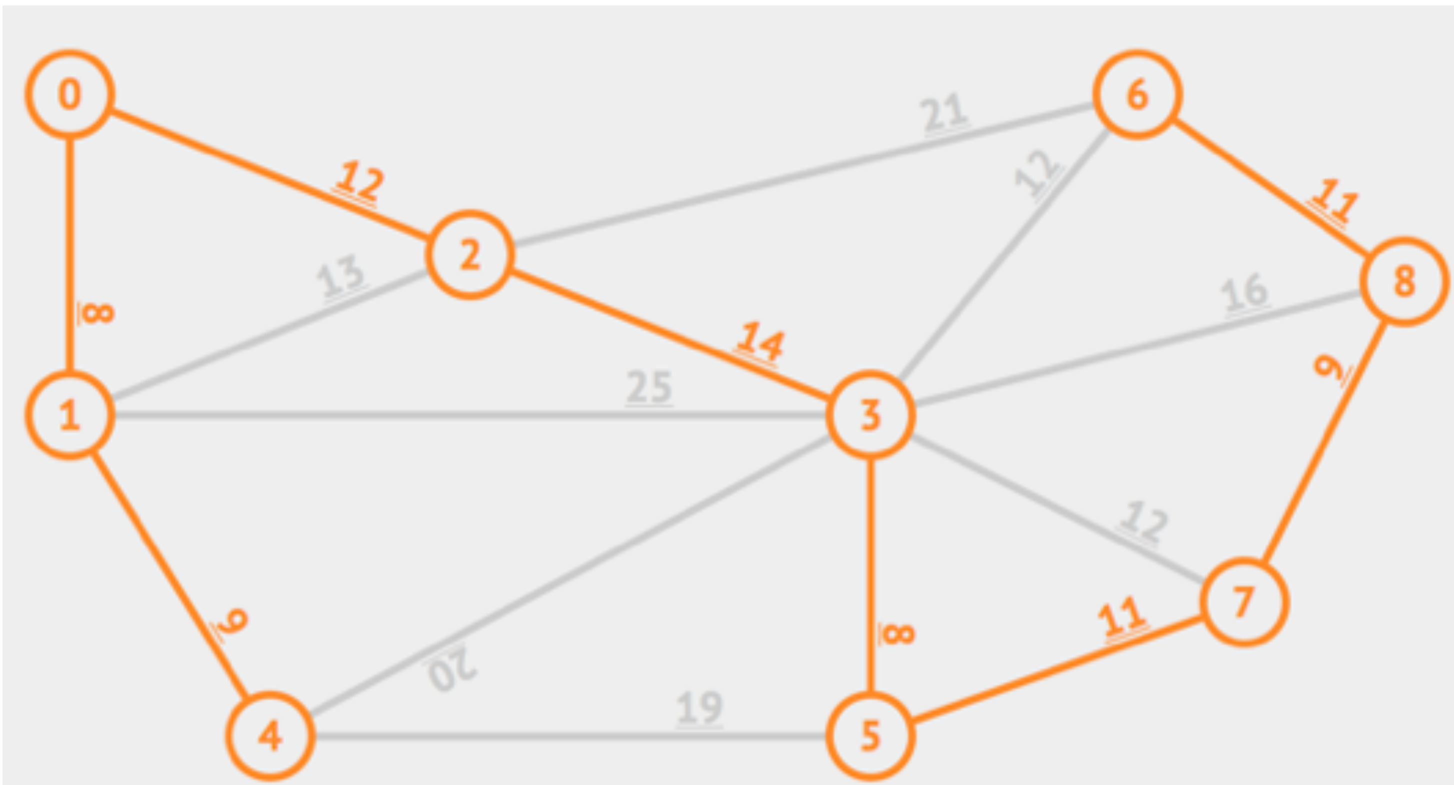
MST edges

0-7 1-7 0-2 2-3 5-7 4-5 6-2

Practice Time



Answer...



Lecture 28: Minimum Spanning Trees

- Introduction
- Kruskal's Algorithm
- Prim's Algorithm

Readings:

- Textbook: Chapter 4.3 (Pages 604-629)
- Website:
  - https://algs4.cs.princeton.edu/43mst/

Practice Problems:

https://visualgo.net/en/mst

Readings:

- Textbook: Chapter 4.3 (Pages 604-629)
- Website:
  - https://algs4.cs.princeton.edu/43mst/

Practice Problems:

https://visualgo.net/en/mst