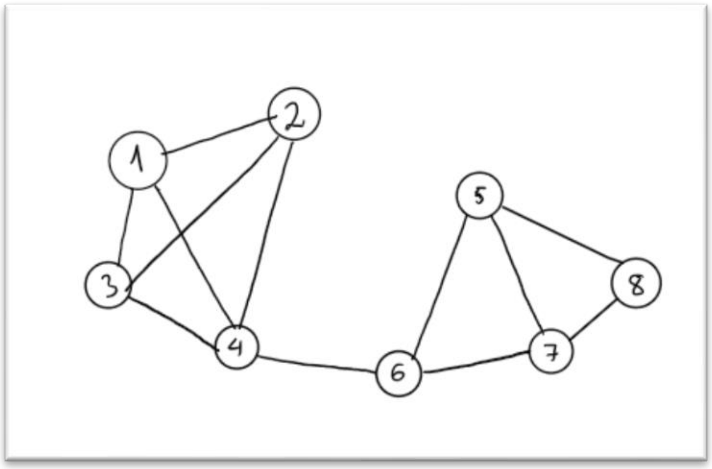
Ex 1

1.



2.

1 4 2 6 3 5 7 8

3.

1 4 6 5 8 7 2 3

Ex 2

Kruskal and Prim algorithm can work well with negative number because those algorithms only compare weight

Dijkstra algorithm does not work with negative number because this algorithm assumes that when add 2 edges together, weight only increase

Ex 3

Step	Selected islands	Unselected islands	Selected Bridge (Edge)
0	{1}	{2,3,4,5,6,7,8}	none
1	{1, 8}	{2,3,4,5,6,7}	(1,8)
2	{1,8,2}	{3,4,5,6,7}	(1,8) (8,2)
3	{1,8,2,4}	{3,5,6,7}	(1,8) (8,2) (2,4)

4	{1,8,2,4, 5}	{3,6,7}	(1,8) (8,2) (2,4) (4,5)
5	{1,8,2,4, 5, 3}	{6,7}	(1,8) (8,2) (2,4) (4,5) (5,3)
6	{1,8,2,4, 5, 3, 6}	{7}	(1,8) (8,2) (2,4) (4,5) (5,3) (2, 6)
7	{1,8,2,4, 5, 3, 6, 7}	{}	(1,8) (8,2) (2,4) (4,5) (5,3) (2, 6) (6,7)

Ex 4

Multiply all edges weight by -1

Sort graph in topological order

Map all vertices and its distance to original vertex in table

From start vertex, update distance from start vertex to its reachable vertices

Repeat above step for other vertices, update distance table

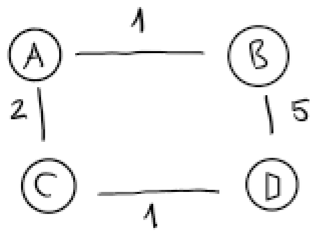
Multiply all edges weight by -1

From table, we can find the longest path from start to goal

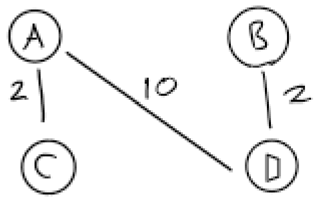
Ex 5

No, this algorithm does not work because it could never reach goal vertex or find the right solution

Example



Start A, goal D: algorithm will find A B D is the shortest path, but A C D is the right answer



Start A, goal D: we can never find D if follow algorithm

Ex 6

1.

deleteEdge(i,j) O(1)

deleteIncidentEdges(i) O(|V|)

2.

deleteEdge(i,j) O(d(i)+d(j))

deleteIncidentEdges(i) O(d(i)+ d(Si(i)) + … + d(Si(d(i)))