Universitetet i Oslo
Institutt for Informatikk

A. Maus, R.K. Runde, I. Yu

INF2220: algorithms and data structures

Series 5

Topic Graphs1 - Solution

Issued: 21. 09. 2017

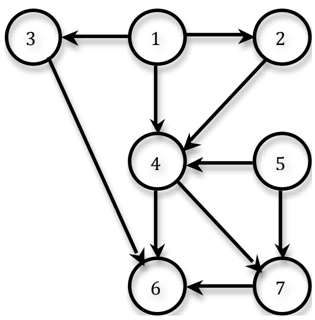
Classroom

Exercise 1

1. 1 Not connected, directed, cyclic
   2 Connected, undirected (it doesn't make so much sense to talk about cyclic undirected graphs)
   3 Connected, directed, cyclic
   4 Not connected, undirected
2. (a) Undirected, not necessesarily connected graph
   (b) Directed, not necessesarily connected graph
   (c) Directed, not necessesarily connected graph

Exercise 2

1.



2. Weakly connected.
3.

Adjacency matrix

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

Adjancency list

1	2	3	4	5	6	7
↓	↓	↓	↓	↓		↓
2	4	6	6	4		6
↓			↓	↓		
3			7	7		
↓						
4						

4.

	1	2	3	4	5	6	7
indegree:	0	1	1	3	0	3	2
outdegree:	3	1	1	2	2	0	1

5. 1, 2, 3, 5, 4, 7, 6

Exercise 3 (Not unique)

1. A, B, D, F, E, C

2. A, C, D, B, E

3. No, the graph is cyclic (several possible cycles)

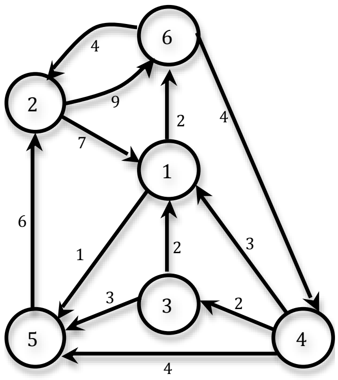
Exercise 4 topSort $O(|V| + |E|)$ + edge test $O(|V|)$

```
//modify topsort from M.A.Weiss
//the topsort part is the same from M.A.Weiss

Bool hamilstonskSti(G) {
  bool isHamiltonskSti = true;
  Queue<Vertex> q = new Queue<Vertex>();
  int counter = 0;
  Vertex [] topsort = new Vertex [|V|];

    for each Vertex v
        if(v.indegree == 0)
            q.enqueue(v);

    while(!q.isEmpty())
    {
      Vertex v = q.dequeue();
      topsort[counter++] = v;

      for each Vertex w adjacent to v
          if(--w.indegree == 0)
              q.enqueue(w);
    }

    //her comes the new part
    if(counter != |V|)
      ishamilstonskSti = false

    else
    {
      for (int =0;i<|V|-1; i++)
      {
        if (!(topsort[i],topsort[i+1]) in G){
          isHamilstonskSti = false;
          break;
        }
```

```
        }
    }
    return isHamilstonskSti;
}
```

Exercise 5

1.



2. It is strongly connected, then it must be weakly connected, of course.

3.

Adjacency matrix

	1	2	3	4	5	6
1	0	0	0	0	1	2
2	7	0	0	0	0	9
3	2	0	0	0	3	0
4	3	0	2	0	4	0
5	0	6	0	0	0	0
6	0	4	0	4	0	0

Adjancency list

1	2	3	4	5	6
↓	↓	↓	↓	↓	↓
5; 1	1; 7	1; 2	1; 3	2; 6	2; 4
↓	↓	↓	↓		↓
6; 2	6; 9	5; 3	3; 2		4; 4
			↓		
			5; 4		

The value after the ";" is the respective weight of the edge

4.

	1	2	3	4	5	6
indegree:	3	2	1	1	3	2
outdegree:	2	2	2	3	1	2

5.
- To 2: $\{$`(1,6;2), (6,2;4)`$\}$, total cost $= 2 + 4 = 6$
- To 3: $\{$`(1,6;2), (6,4;4), (4,3;2)`$\}$, total cost $= 2 + 4 + 2 = 8$
- To 4: $\{$`(1,6;2), (6,4;4)`$\}$, total cost $= 2 + 4 = 6$
- To 5: $\{$`(1,5;1)`$\}$, total cost $= 1$
- To 6: $\{$`(1,6;2)`$\}$, total cost $= 2$