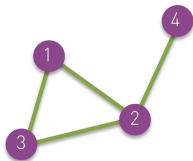


network *representations*

introduction to *network analysis* (*ina*)

Lovro Šubelj
University of Ljubljana
spring 2022/23

network *representations*



undirected graph

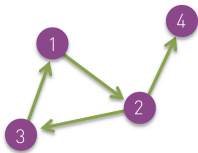
$$\begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

adjacency matrix

1: [2, 3]	{1, 2}
2: [1, 3, 4]	{1, 3}
3: [1, 2]	{2, 3}
4: [2]	{2, 4}

adjacency list

edge list



directed graph

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

adjacency matrix

[3] :1: [2]	(1, 2)
[1] :2: [3, 4]	(2, 3)
[2] :3: [1]	(2, 4)
[2] :4: []	(3, 1)

adjacency list

edge list

* adjacency list can be implemented with maps or trees & edge list does not represent isolated nodes

network *representations*

- *adjacency matrix* for elegant *analytical derivations*
most derivations based on matrix representation[†]
- *adjacency list* for efficient *algorithms implementation*
ideal complexity since most algorithms require incidence[†]
- *edge list* for efficient *network storing/manipulation*
easy editing since each edge stored only once

[†]some derivations can also be based on adjacency list & some algorithms require edge list

network *structures*

— *edge list edges data structure* complexity

data structure	link manipulation	random node	random link
array	none	$\mathcal{O}(m)$	$\mathcal{O}(1)$
array list	addition	$\mathcal{O}(m)$	$\mathcal{O}(1)$
hash map	any	$\mathcal{O}(m)$	$\mathcal{O}(m)$

— *adjacency list nodes data structure* complexity

data structure	node manipulation	random node	random link
array	none	$\mathcal{O}(1)$	$\mathcal{O}(m)$
array list	addition	$\mathcal{O}(1)$	$\mathcal{O}(m)$
hash map	any	$\mathcal{O}(n)$	$\mathcal{O}(m)$

— *adjacency list neighbors data structure* complexity

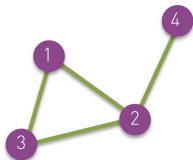
data structure	link manipulation	node incidence	random neighbor
array	none	$\mathcal{O}(k)$	$\mathcal{O}(1)$
array list	addition	$\mathcal{O}(k)$	$\mathcal{O}(1)$
hash map	any	$\approx \mathcal{O}(1)$	$\mathcal{O}(k)$
tree map	any	$\mathcal{O}(\log k)$	$\mathcal{O}(k)$

— *hash maps* for *construction* and *arrays* for *analysis*

— use *directed adjacency list* with *undirected flag*

‡ random link selection equivalent to random node selection by degree

network *formats*



undirected graph

```
# undirect
1 2
1 3
2 3
2 4
```

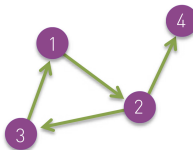
edge list

```
*vertices 4
1 "1"
2 "2"
3 "3"
4 "4"
*edges
1 2
1 3
2 3
2 4
```

Pajek format

```
# undirect
# 0 "1"
# 1 "2"
# 2 "3"
# 3 "4"
#
0 1
0 2
1 2
1 3
```

my format



directed graph

```
# directed
1 2
2 3
2 4
3 1
```

edge list

```
*vertices 4
1 "1"
2 "2"
3 "3"
4 "4"
*arcs
1 2
2 3
2 4
3 1
```

```
# directed
# 0 "1"
# 1 "2"
# 2 "3"
# 3 "4"
#
0 1
1 2
1 3
2 0
```

§ ad-hoc edge list and Pajek format most popular & other formats include GML, GraphML and JSON

network *data*

- easily obtained from *online sources*
- already present in many *standard datasets*
- *personal web pages* of network researchers
- popular *network repositories/collections*
 - Network Catalogue and Repository [[Netzschleuder](#)]
 - Colorado Index of Complex Networks [[ICON](#)]
 - Stanford Network Analysis Project [[SNAP](#)]
 - Koblenz Network Collection [[KONECT](#)]
 - Open Graph Benchmark [[OGB](#)]
 - Network Repository [[NetRepo](#)]
 - Pajek datasets [[Pajek](#)]

network *software*

- most popular *Python libraries*
 - igraph [<https://igraph.org>]
 - **NetworkX** [<https://networkx.org>]
 - graph-tool [<https://graph-tool.skewed.de>]
 - Snap.py [<https://snap.stanford.edu/snappy>]
 - Pajek [<http://mrvar.fdv.uni-lj.si/pajek>]
- most popular *network software*
 - **Gephi** [<https://gephi.org>]
 - visone [<https://visone.ethz.ch>]
 - Pajek [<http://mrvar.fdv.uni-lj.si/pajek>]