Voorbereiding Programmeerwedstrijden

najaar 2019

http://www.liacs.leidenuniv.nl/~vlietrvan1/vbpw/

Rudy van Vliet
kamer 140 Snellius, tel. 071-527 2876
rvvliet(at)liacs(dot)nl

college 5, 10 oktober 2019

Graph Traversal

Graph Algorithms

(Eind)Programmeerwedstrijd

donderdag 31 oktober, 14.00-18.00 uur

9. Graph Traversal

graph is unifying theme of computer science

9.2. Data Structures for Graphs

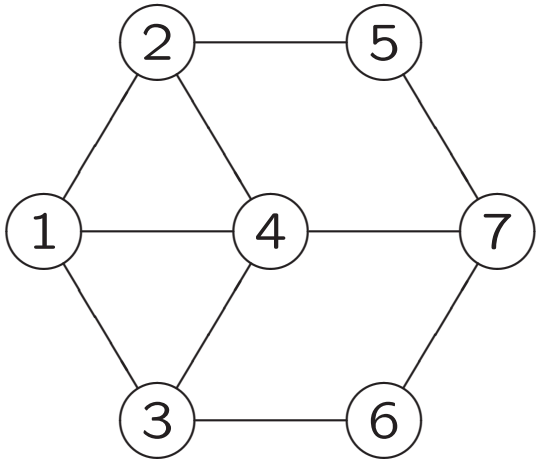
- adjacency matrix
  - \+ quick access, insertion, deletion of edge
  - − much space (e.g., Manhattan), slow iteration over neighbours

- adjacency list in lists
  - − slow access, deletion of edge
  - \+ less space, quick iteration over neighbours

- adjacency list in matrix

- table of edges (for Kruskal)

Adjacency List in Matrix



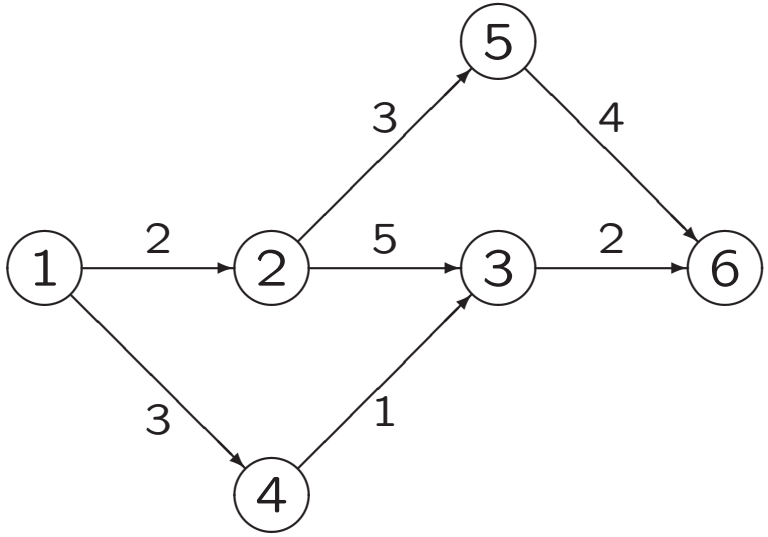
node	degree	neighbours			
		0	1	2	3
1	3	2	3	4	
2	3	1	4	5	
3	3	1	4	6	
4	4	1	2	3	7
5	2	2	7		
6	2	3	7		
7	3	4	5	6	

- quick iteration over neighbours
- no pointers

9.5. Topological Sorting

- of DAG
- all directed edges from left to right
- useful
  - for schedule respecting precedence constraints
  - for finding shortest / longest path from x to y (with dynamic programming)...
- implementation
  - variant of dfs
  - $*$ count incoming edges per vertex
    - $*$ maintain queue of vertices without incoming edges
- see Algoritmiek

Topological Sorting (Example)



finding longest path in DAG. . .

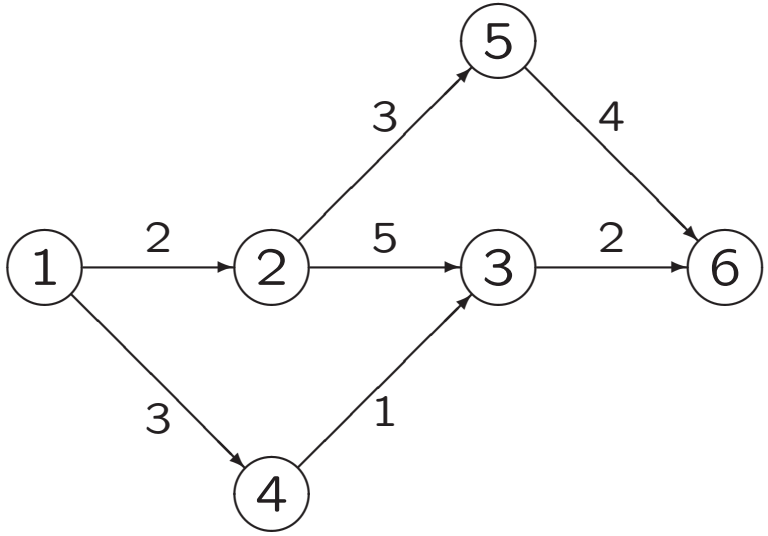
LKP2019. Exits in Excess

9.6.5. Edit Step Ladders

9.6.6. Tower of Cubes

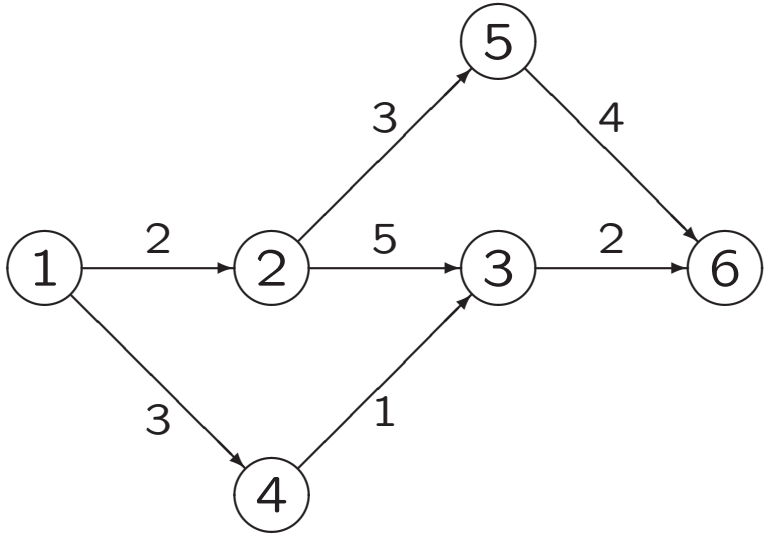
9.6.8. Hanoi Tower Troubles Again!

10.4. Network Flows and Bipartite Matching



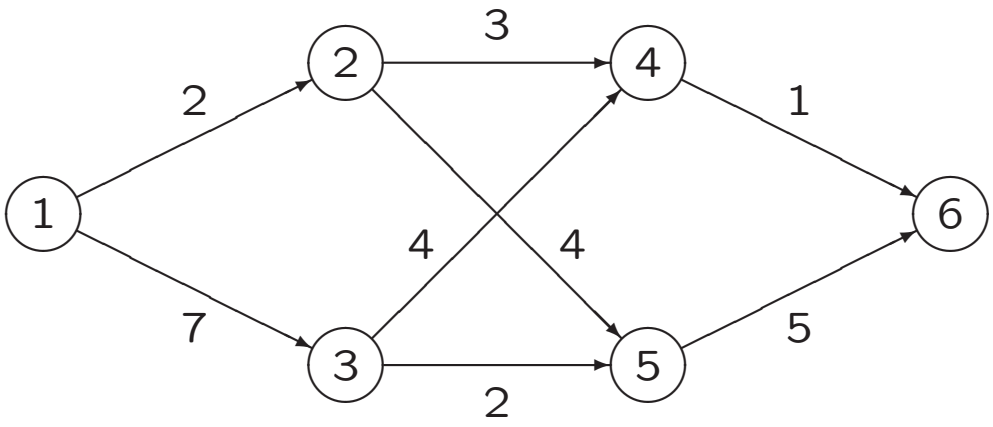
maximum flow from 1 to 6 is . . .

Ford-Fulkerson Algorithm (Example)



repeat finding augmenting paths

Ford-Fulkerson Algorithm (Example)

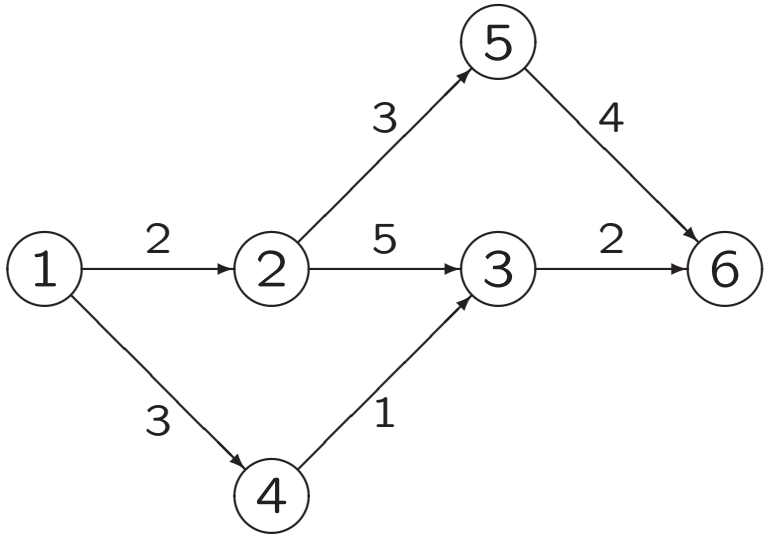


maximum flow from 1 to 6 is . . .

Ford-Fulkerson Algorithm

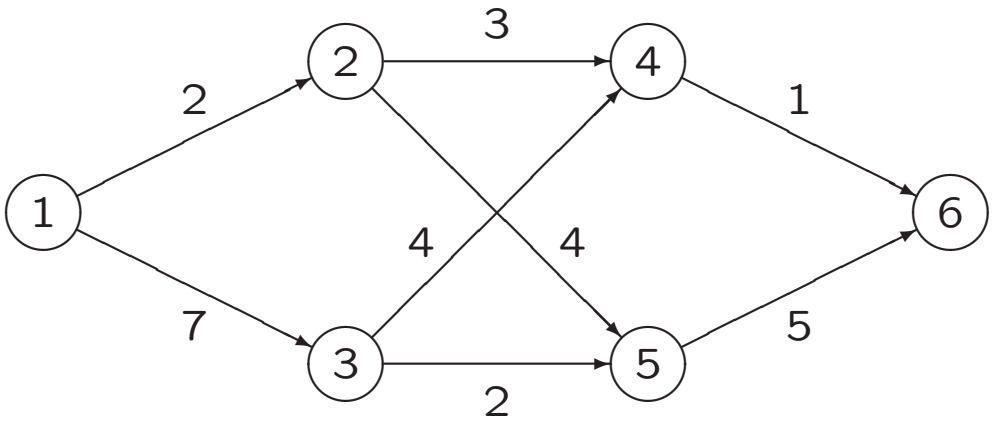
```
void netflow (flowgraph *g, int source, int sink)
{ int volume;    // weight of augmenting path

  add_residual_edges (g);

  dfs (g, source);
  volume = path_volume (g, source, sink, parent);

  while (volume>0)
  { augment_path (g, source, sink, parent, volume);
    dfs (g, source);
    volume = path_volume (g, source, sink, parent);
  }
}
```

Minimum Cut



minimum cut of 1 and 6 is . . .

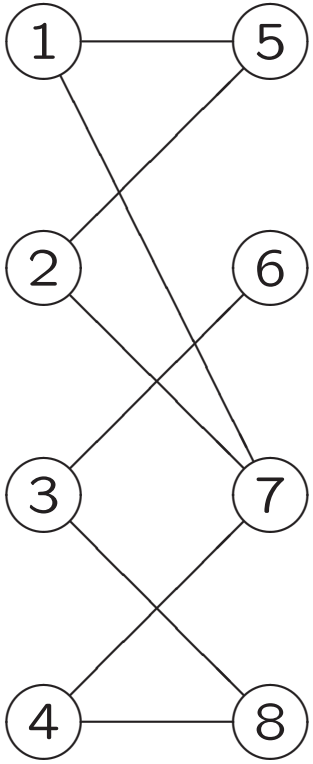
Minimum Cut



- minimum cut of 1 and 6 is . . .

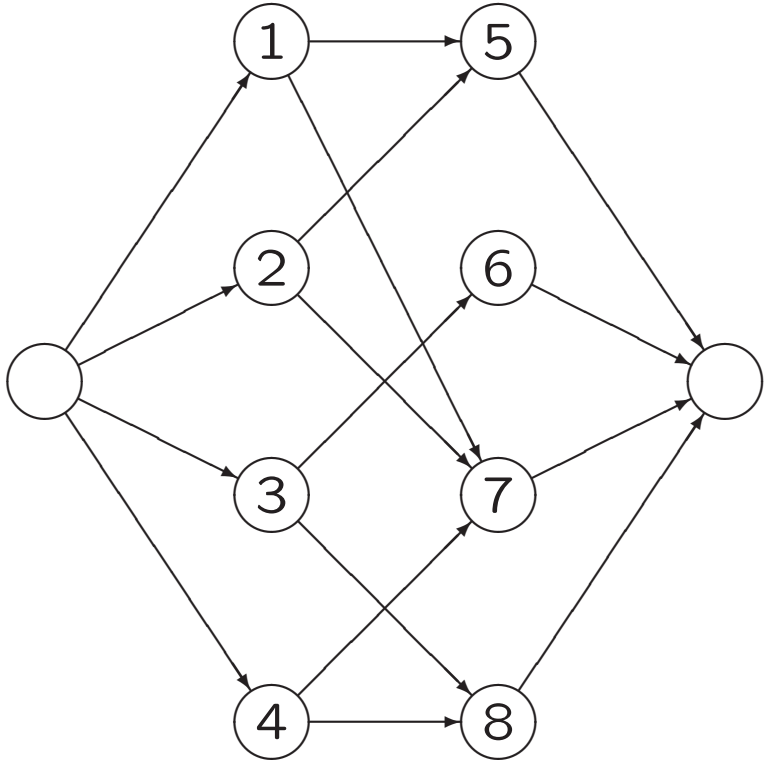
- finding minimum cut. . .

Maximum Bipartite Matching



maximum matching. . .

Maximum Bipartite Matching



maximum matching $\approx$ maximum flow from source to sink

9.6.8. Hanoi Tower Troubles Again!