Voorbereiding Programmeerwedstrijden

najaar 2025

https://liacs.leidenuniv.nl/~vlietrvan1/vbpw/

Rudy van Vliet
kamer BM.2.03, Gorlaeus, tel. 071-527 2876
rvvliet(at)liacs(dot)nl

college 6, 7 oktober 2025

Graph Traversal

Graph Algorithms

Huiswerkopgave 3

8.6.6. Garden of Eden

- smarter solution: for all initial triples, slide through bitstring with feasible triples
- no need to remember (or construct) everything in between
- deadline donderdag 9 oktober, 08.59 uur

9.6.5. Edit Step Ladders

LKP2019. Exits in Excess

9.6.8. Hanoi Tower Troubles Again!

9.1. Flavors of Graphs

- undirected vs. directed
- weighted (edge / vertex) vs. unweighted
- cyclic vs. acyclic (e.g., tree, DAG)
- simple vs. non-simple (self loop / multi-edge)
- implicit vs. explicit
- labeled vs. unlabeled vertices (isomorphism)

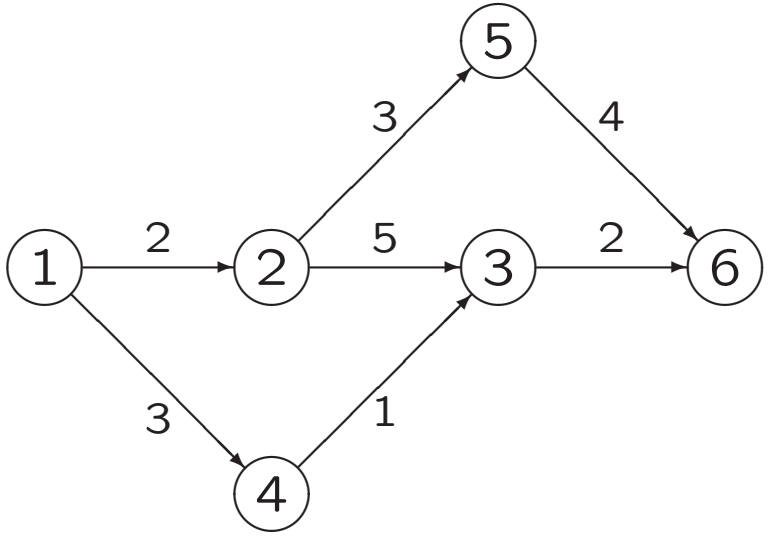
9.3. Graph Traversal: Breadth-First

- with queue
- mark visited vertices
- store parent to find back path
- useful
  - if order of visits does not matter (e.g., connected components)
  - for shortest paths on unweighted graphs
    (or graphs with only weights 0 and 1)
- see Algoritmiek

9.4. Graph Traversal: Depth-First

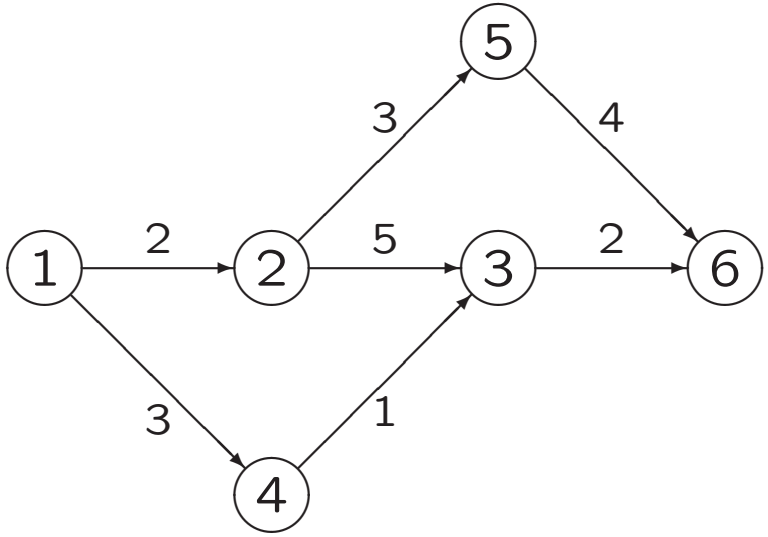
- with stack or recursion
- mark visited vertices
- store parent to find back path
- useful
  - for finding cycles
    - in undirected graph:
      - only tree edges (to unvisited node or parent) and back edges (to visited node $\neq$ parent)
      - back edge $\Longleftrightarrow$ cycle
    - in directed graph: a bit more involved
  - for finding connected components
- see Algoritmiek

10.4. Network Flows and Bipartite Matching



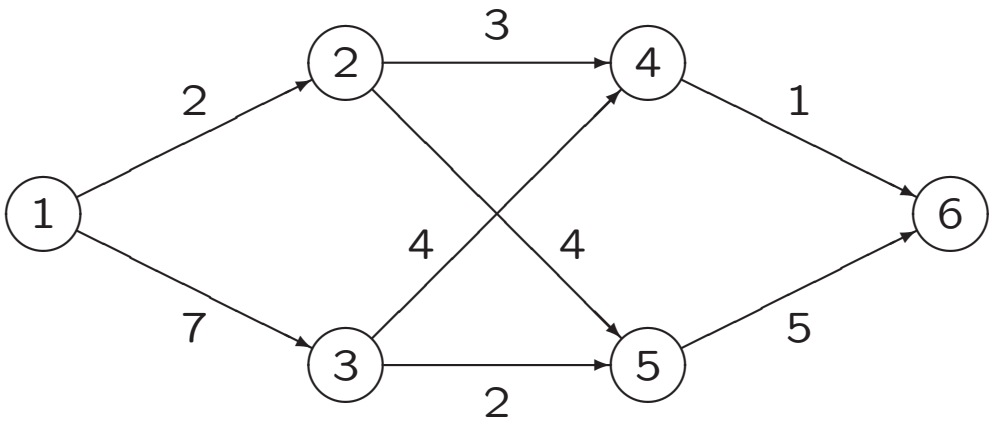
maximum flow from 1 to 6 is . . .

Ford-Fulkerson Algorithm (Example)



repeat finding augmenting paths

Ford-Fulkerson Algorithm (Example)

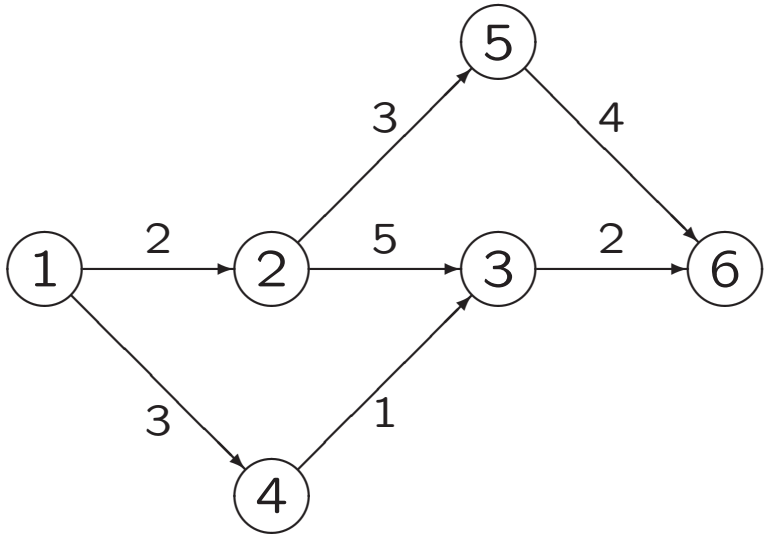


maximum flow from 1 to 6 is . . .

Ford-Fulkerson Algorithm

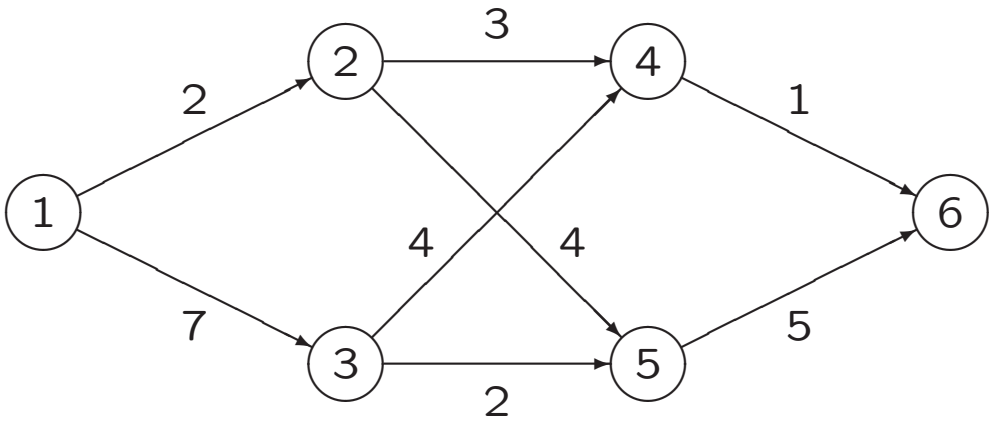
```
void netflow (flowgraph *g, int source, int sink)
{ int volume;    // weight of augmenting path

  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 . . .

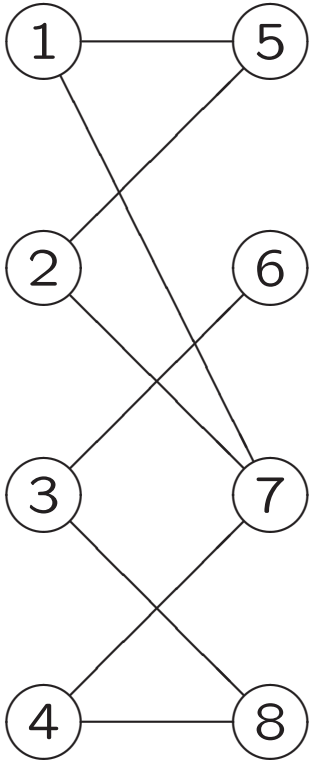
Minimum Cut



- minimum cut of 1 and 6 is . . .

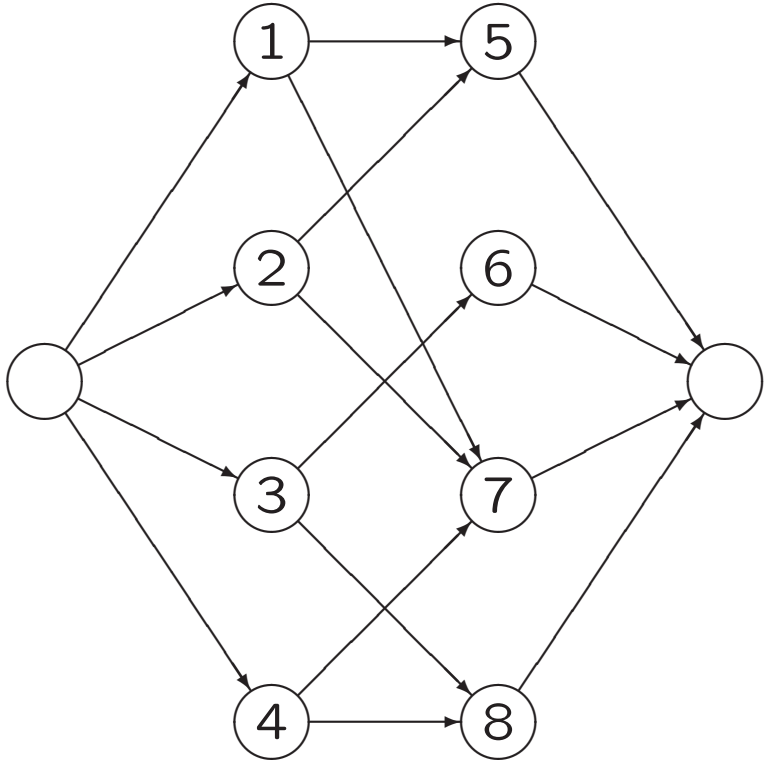
- finding minimum cut. . .

Maximum Bipartite Matching



maximum matching. . .

Maximum Bipartite Matching



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

Evacuation

9.6.8. Hanoi Tower Troubles Again!

10.3. Shortest Paths

- Dijkstra's algorithm

// invoer: samenhangende gewogen graaf $G = (V, E)$ en startknoop s
// uitvoer: array dat de lengtes van de kortste paden vanuit s bevat;
// na afloop is pad[v] = de lengte van een kortste pad van s naar v

for $v \in V$ **do**
    pad[v] := ∞;
od
pad[s] := 0;
$U := \emptyset$;

while ($U \neq V$) **do**
    vind knoop $v^* \in V \setminus U$ met pad[v^*] minimaal;
    $U := U \cup \{v^*\}$;
    for alle knopen v aangrenzend aan v^* **do**
        if pad[v^*] + gewicht(v^*, v) < pad[v] **then**
            pad[v] := pad[v^*] + gewicht(v^*, v);
        fi
    od
od

Complexiteit (met adjacency matrix):
$\Theta(n + 1 + n(n + 1 + n)) = \Theta(n^2)$

// invoer: samenhangende gewogen graaf $G = (V, E)$ en startknoop s
// uitvoer: array dat de lengtes van de kortste paden vanuit s bevat;
// na afloop is pad[v] = de lengte van een kortste pad van s naar v

for $v \in V$ **do**
    pad[v] := ∞;
od
pad[s] := 0;
$U := \emptyset$;

while ($U \neq V$) **do**
    vind knoop $v^* \in V \setminus U$ met pad[v^*] minimaal;
    $U := U \cup \{v^*\}$;
    for alle knopen v aangrenzend aan v^* **do**
        if pad[v^*] + gewicht(v^*, v) < pad[v] **then**
            pad[v] := pad[v^*] + gewicht(v^*, v);
        fi
    od
od

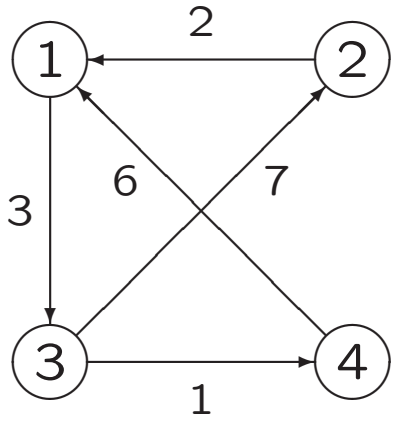
Complexiteit (met adjacency list en heap):
$O(n + 1 + n + n \log n + m \log n) = O(m \log n)$

LKP2021. Candy Contribution

10.3.2 All-Pairs Shortest Paths

- to find diameter of graph
- to find 'center' vertex
- n times Dijkstra's algorithm: $O(n \times n^2)$ with adjacency matrix
- Floyd's algorithm: $O(n^3)$

Floyd's Algorithm (Example)

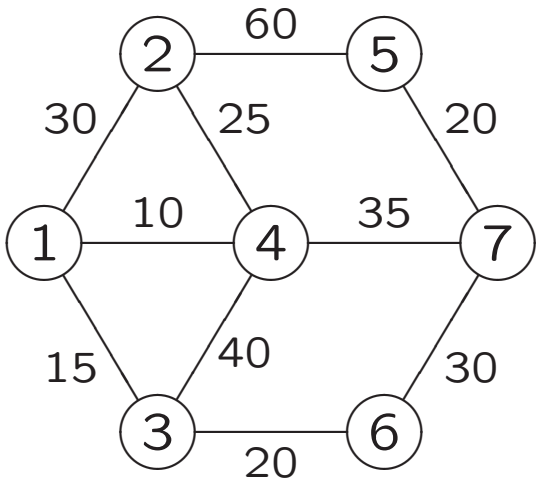


	1	2	3	4
1	0	∞	3	∞
2	2	0	∞	∞
3	∞	7	0	1
4	6	∞	∞	0

10.2. Minimum Spanning Trees

- Prim's algorithm
- Kruskal's algorithm
- see Datastructuren

Prim's Algorithm (Example)



9.6.3. The Tourist Guide