Martin Ziegler

Syllabus

5. Graph Problems

– Recap on Graphs: un/directed, weighted

– Shortest Paths: single-source, all-pairs

– Minimum Spanning Tree: Prim, Kruskal

– Maximum Flow: Ford-Fulkerson, Edmonds-Karp

– Maximum (weighted) Bipartite Matching

– Minimum Cut

5. Graph Problems

Martin Ziegler

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Basic graph concepts:

- simple: no *multi*-edges
  nor loops
- in-/out-/degree
- (un-/directed) path
- (strongly) connected component
- subgraph, induced graph

$E \subseteq V \times V$ directed edges

E symmetric: *un*directed edges

$w:E \to \mathbb{N}$ edge weights

∞/0: absent
1: present
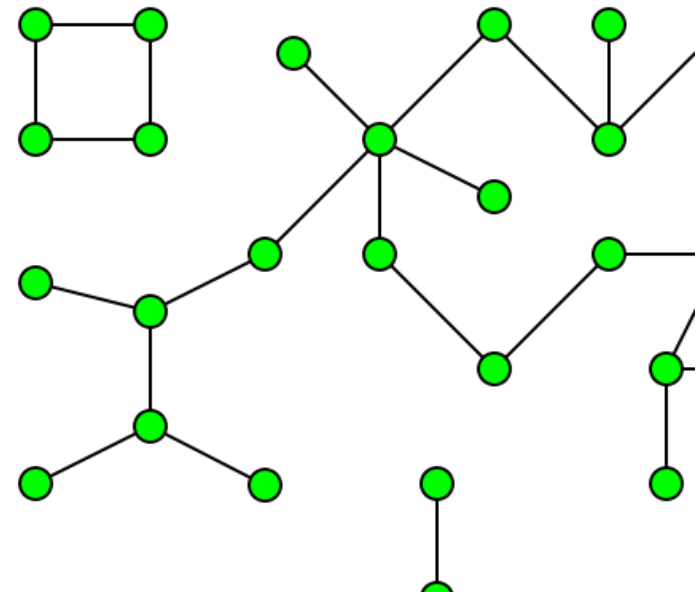
Handshaking lemma:

$$\#E = \sum_{v \in V} \text{indeg}(v) = \sum_{v \in V} \text{outdeg}(v)$$
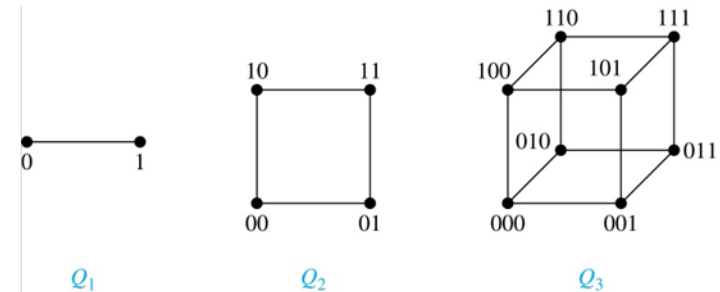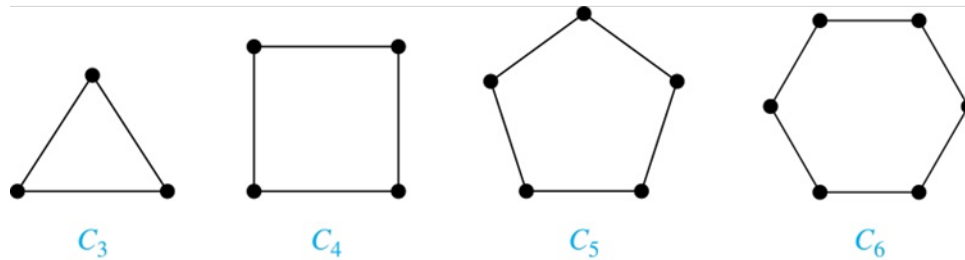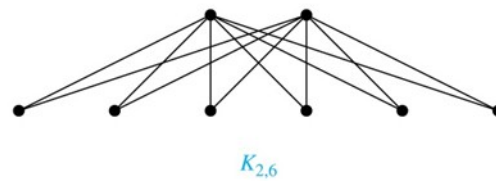
Adjacency/weight matrix

$$A_G \in \mathbb{N}^{V \times V} \qquad \text{Powers of } A_G$$
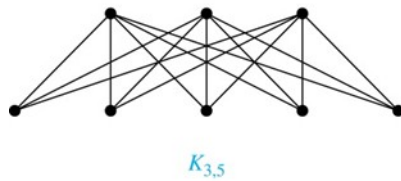
5. Graph Problems

graph examples

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

$K_{2,3}$ $\quad$ $K_{3,3}$

$P(0,0)$ $P(0,1)$ $P(0,2)$ $P(0,3)$

$P(1,0)$ $P(1,1)$ $P(1,2)$ $P(1,3)$

$P(2,0)$ $P(2,1)$ $P(2,2)$ $P(2,3)$

$P(3,0)$ $P(3,1)$ $P(3,2)$ $P(3,3)$

$K_{3,5}$ $\quad$ $K_{2,6}$

C_3 $\quad$ C_4 $\quad$ C_5 $\quad$ C_6 $\quad$ Q_1 $\quad$ Q_2 $\quad$ Q_3

diameter? planar?

K_1 $\quad$ K_2 $\quad$ K_3 $\quad$ K_4 $\quad$ K_5 $\quad$ K_6
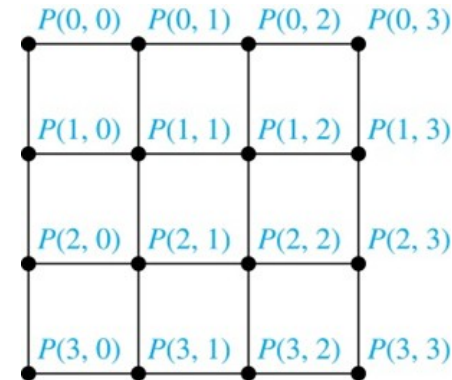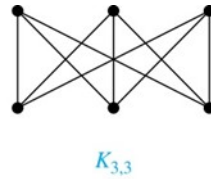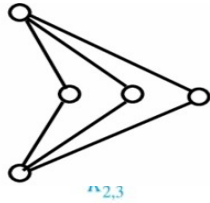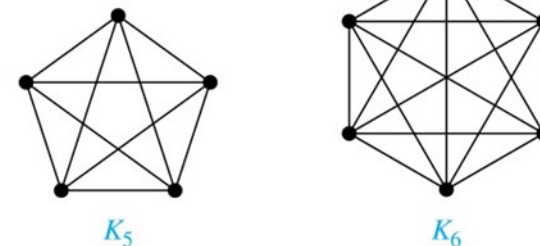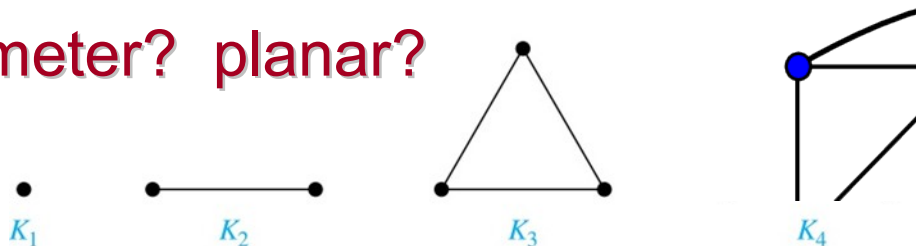
5. Graph Problems

Connectedness

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: A_G; $s,t \in V$

Adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: Is there a (directed) path from s to t in G?

$A_{u,v} = \infty$
no edge

$A_{u,u} = 0$

DFS(v) // Is t reachable in G from v?

If $v=t$ Return (**true**);

If v is marked *visited*
        Return (**false**);

Mark v as *visited*;

For each neighbor u of v do
    if DFS(u) Return (**true**);
Return (**false**);

Reachable(G,s,t)

For each vertex $v \in V$
        Mark v as *un*visited;
Return $DFS(s)$
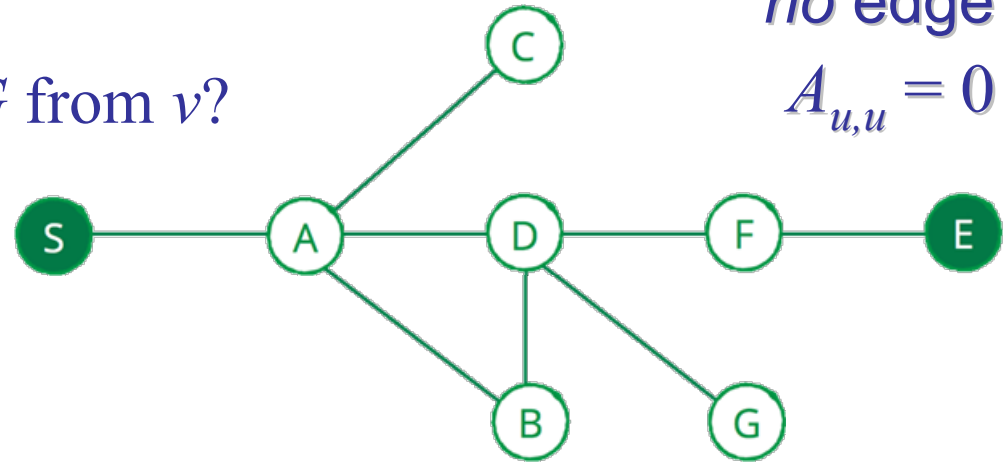
5. Graph Problems Shortest path(s)

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: A_G $s,t \in V$ Adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: weight $d(s,t)$ of lightest path from s to t.

Input: A_G $s \in V$ Adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: For every $t \in V$, weight $d(s,t)$ of lightest path from s to t.

Input: A_G Adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: For every $s,t \in V$, weight $d(s,t)$ of lightest path from s to t.

Remark: Shortest paths (on *non*-negative edge weights) are simple paths:
- W.l.o.g. consider only positive edge weights: otherwise merge vertices.
- In a shortest path $(s,v_1,v_2,\ldots v_k,\ldots v_l,\ldots t)$ from s to t,
  all segments $(v_k,\ldots v_l)$ are shortest paths.

5. Graph Problems

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: A_G; $s \in V$ Adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: For every $t \in V$, <u>weight</u> $d(s,t)$ of lightest path from s to t.

Dijkstra's Algorithm:

Mark all vertices *un*visited.

set of *un*visited vertices

Initialize $Q:=V$.   tentativ distance from s

For each vertex v let $d_v:=\infty$; $d_s:=0$.

While $Q \neq \varnothing$ do   Correctness???
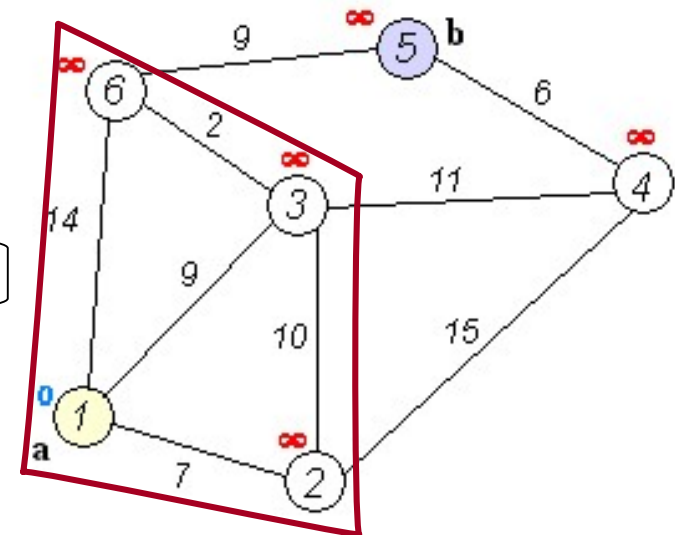
Extract from Q a vertex u with least d_u. Mark u as *visited*.

For each *un*visited neighbor u of v do

If $d':=d_u+A_{uv} < d_v$ then decrease $d_v:=d'$.

$O(n \cdot \textit{extractMin} + m \cdot \textit{decreaseKey})$

5. Graph Problems

Shortest path(s)

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: A_G Adjacency/*weight* matrix $A_G \in \mathbb{N}^{V \times V}$

Output: For every $t \in V$, <u>weight</u> $d(s,t)$ of *lightest* path from *s* to *t*.

Loop invariant $d_v \geq d(s,v)$. Suppose $M := \{\, v :\; d_v > d(s,v) \,\} \neq \varnothing$.

Then $\delta := \min\{\, d(s,v) : v \in M \,\}$ and $v \in M$ with $d(s,v)=\delta$ exist.

For $(s , \dots , u , v)$ a *lightest* path to v, it holds $\delta > d(s,u) = d_u.$

Thus $d(s,v)= d(s,u)+A_{uv}$ and u gets extracted from Q <u>before</u> v.

For correctness, recall main loop: While $Q \neq \varnothing$ do

   Extract from Q a vertex u with least d_u. Mark u as *visited.*

   For each *un*visited neighbor u of v do

      If $d':=d_u+A_{uv} < d_v$ then decrease $d_v:=d'$.

in increasing order w.r.t. d

5. Graph Problems *All* shortest paths

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: A_G Adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: For **all** $s,t \in V$, <u>weight</u> $d(s,t)$ of lightest path from s to t.

Floyd-Warshall Algorithm: runtime $O(n^3)$

$A_{u,v} = \infty$
no edge

$A_{u,u} = 0$

For all pairs (u,v) of vertices, initialize $d_{u,v}:=A_{u,v}$

For each vertex $u \in V$

   For each vertex $v \in V$

      For each vertex $w \in V$

         If $d_{v,w} > d_{v,u} + d_{u,w}$ then

            $d_{v,w} := d_{v,u} + d_{u,w}$

Correctness

Dijkstra
(fixed $s \in V$):

$O(n \cdot extractMin + m \cdot decreaseKey)$

Repeat for
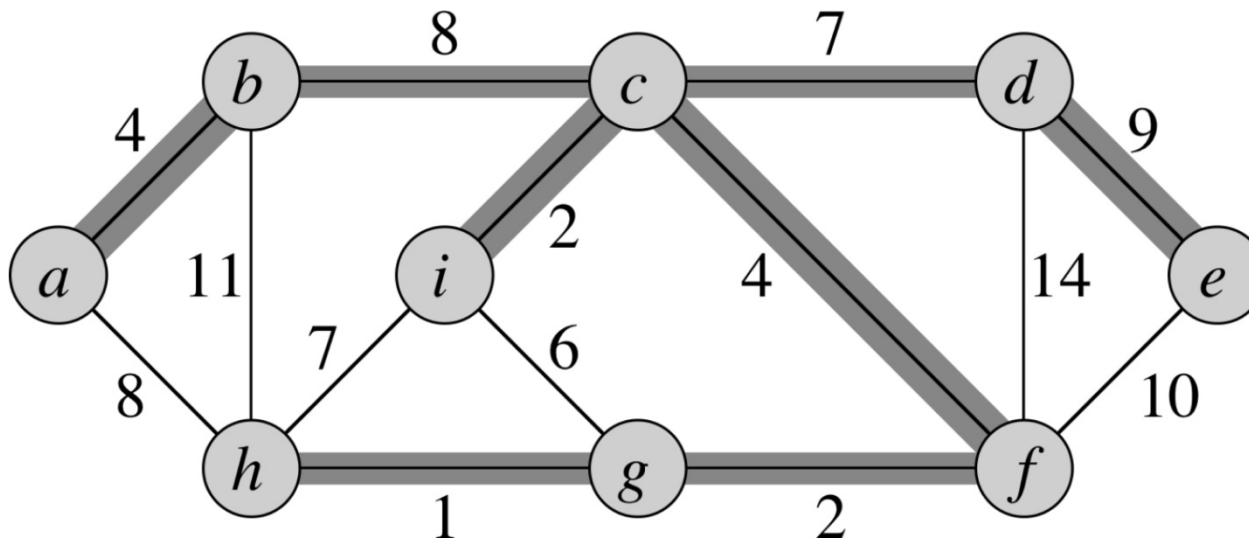each $s \in V$

5. Graph Problems   *Min.* Spanning Tree

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: A_G **Symmetric** adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: $T \subseteq E$ <u>spanning tree</u> of least weight

s.t. (V,T) connected

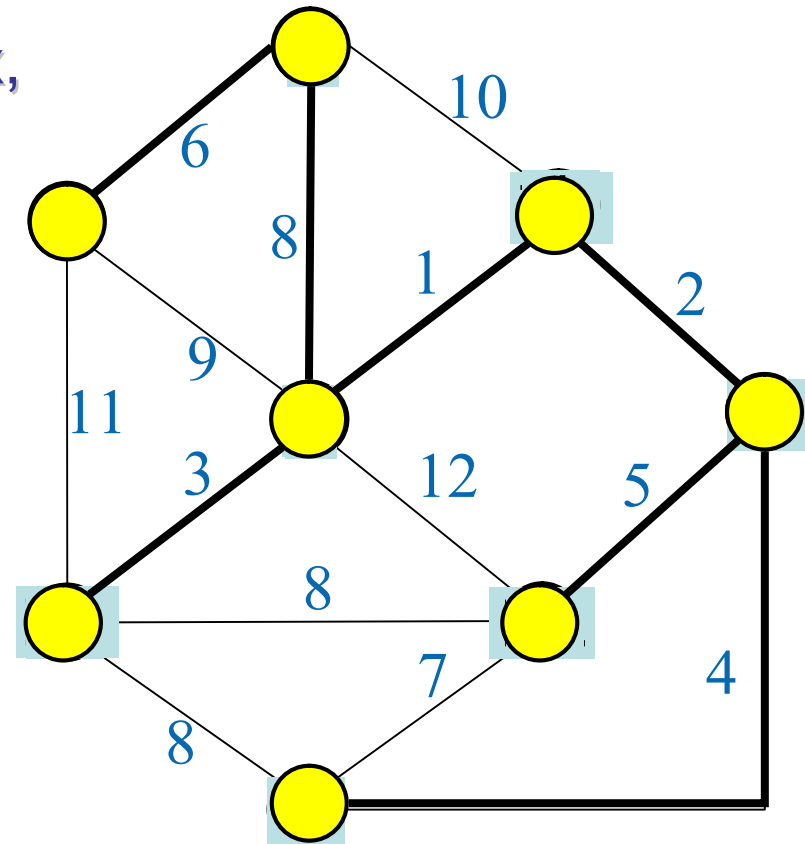$A_{u,v} = \infty$
no edge

$A_{u,u} = 0$

5. Graph Problems — Prim's Algorithm

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: A_G **Symmetric** adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: $T \subseteq E$ <u>spanning tree</u> of least weight

1. Initialize a tree with a single vertex, chosen arbitrarily from the graph.

2. Grow the tree by one edge:
   Of the edges that connect the tree to vertices not yet in the tree,
   find the minimum-weight edge,
   and transfer it to the tree.

3. Repeat step 2
   (until all vertices are in the tree).

5. Graph Problems

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: A_G **Symmetric** adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: $T \subseteq E$ <u>spanning tree</u> of least weight

$O(n \cdot extractMin + m \cdot decreaseKey)$

Initialize $F := \varnothing$, $Q := V$. Also:
$d_v := \infty$ and $e_v := 0$ for all $v \in V$.

While $Q \neq \varnothing$ do

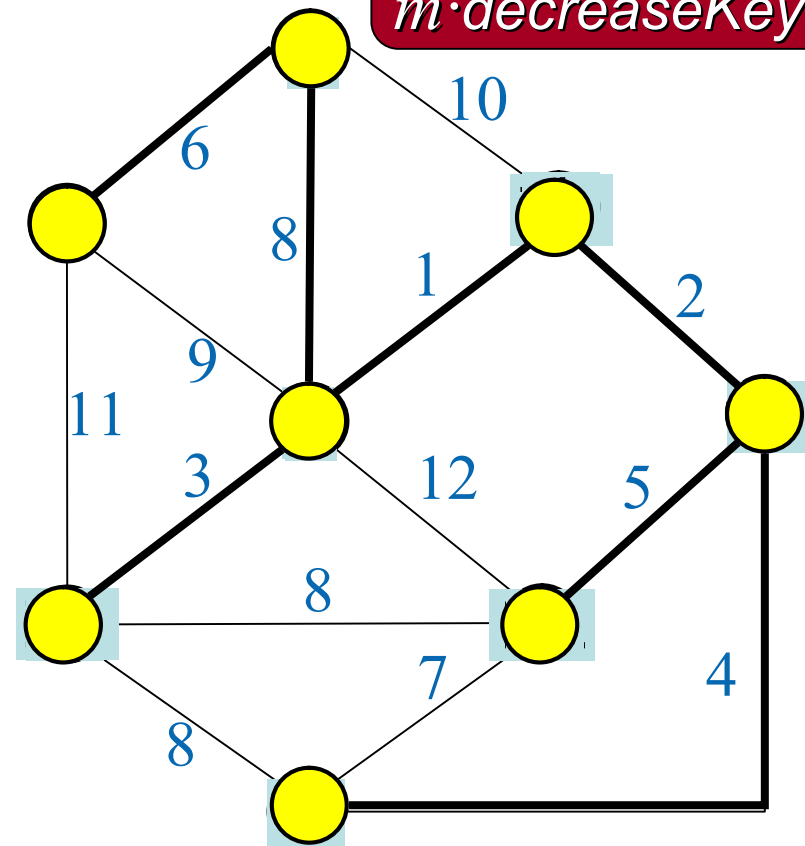 Extract from Q
   a vertex u with least d_u.
  If $e_u \neq 0$, add edge (u, e_u) to F.

  For each neighbor $v \in Q$ of u do

   If $A_{uv} < d_v$ then
   decrease $d_v := A_{uv}$; $e_v := u$;

5. Graph Problems

Kruskal Algorithm

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: A_G **Symmetric** adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: $T \subseteq E$ <u>spanning tree</u> of least weight

Initialize the forest (=set of trees) with edges $F:=\{\}$, i.e., such that each vertex $v \in V$ is a separate tree.
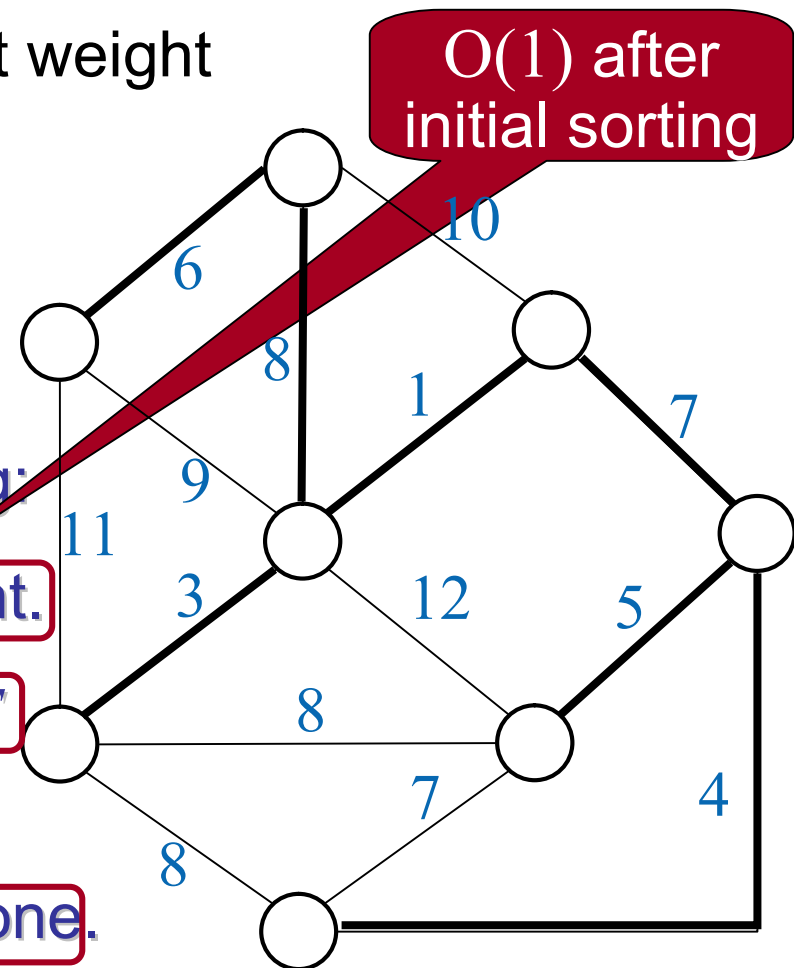
While $E \neq \{\}$ and F is not yet spanning:

Extract from E edge e of least weight.

If e connects two different trees of F

then add e to F, thus combining two trees into a single one.

O(1) after initial sorting

?

5. Graph Problems

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

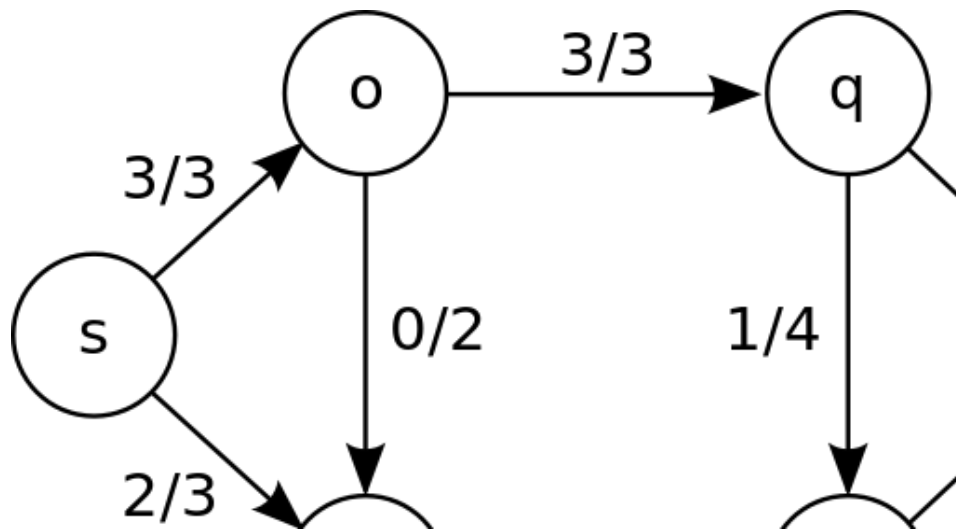**Input:** $s,t \in V$, A_G adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: $f:V^2 \to \mathbb{R}$ max. flow from s to t

Goal: maximize
$$\sum_{v:(s,v)\in E} f(s,v)$$
$$= \sum_{u:(u,t)\in E} f(u,t)$$

f flow (from s to t)

Lemma: There exists an *integral* maximal flow.

Def: A flow from s to t in G with weights $A \geq 0$ is a function $f:V^2 \to \mathbb{R}$ such that $\forall v \in V \setminus \{s,t\}$: $\sum_{u:(u,v)\in E} f(u,v) = \sum_{w:(v,w)\in E} f(v,w)$ and $f(u,v) = -f(v,u)$. It is admissible if it holds $f(u,v) \leq A_{u,v}$

5. Graph Problems   *Ford-Fulkerson*

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

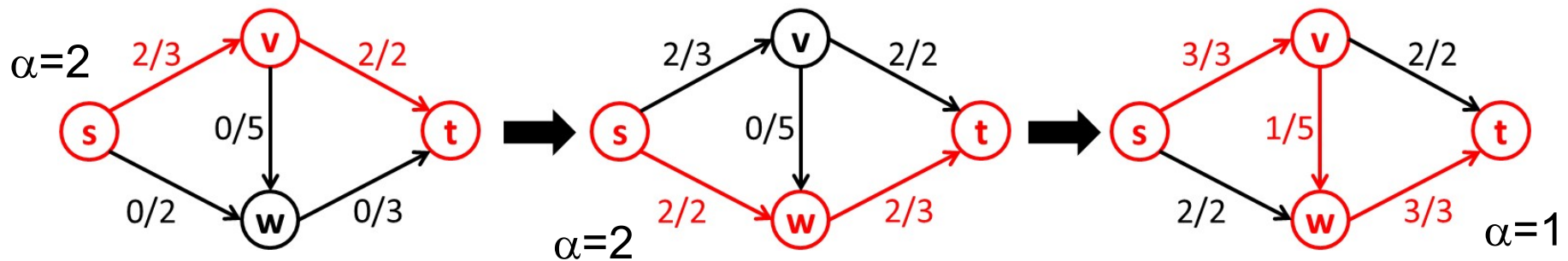**Input:** $s,t \in V$, A_G adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: $f: V^2 \to \mathbb{R}$ <u>max. flow</u> from s to t

Goal: maximize $|f| := \sum_{v:(s,v) \in E} f(s,v)$

The <u>residual</u> G_f of a graph G with flow f
has edges $E_f := \{ (u,v) : A_{u,v} > f(u,v) \lor f(v,u) > 0 \}$

$\alpha=2$ 2/3   2/2   0/5   0/2   0/3

$\alpha=2$ 2/3   2/2   0/5   2/2   2/3

3/3   2/2   1/5   2/2   3/3 $\alpha=1$

Ford-Fulkerson: Initialize $f \equiv 0$.   Correctness?   Runtime $O(m \cdot |f|)$
While there exists some path $P = (s=u_1, \ldots u_K = t)$ from s to t in G_f
Let $\alpha := \min\{ A_{u_k,u_{k+1}} - f(u_k,u_{k+1}) : k=1 \ldots K-1 \}$ and $f := f + \alpha \cdot P$.

5. Graph Problems   *Edmonds-Karp*

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

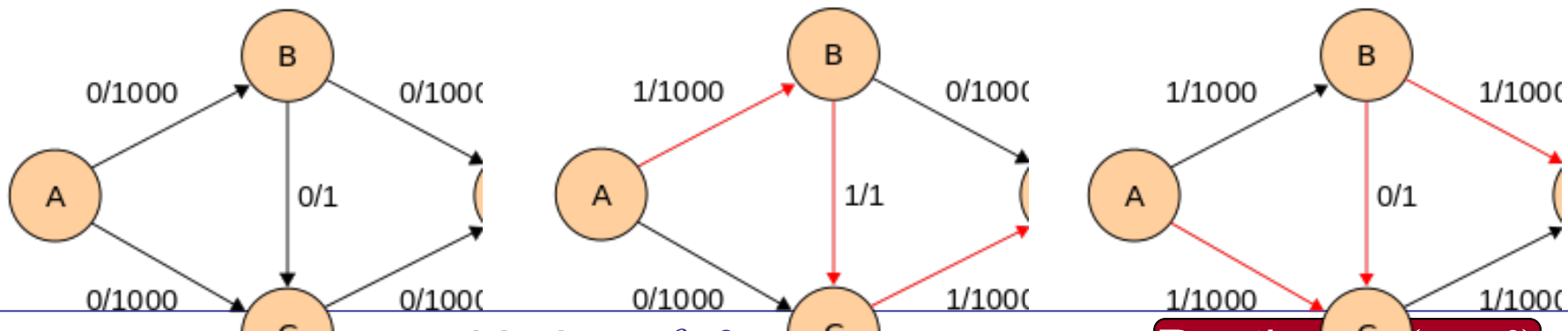**Input:** $s,t \in V$, A_G adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$

Output: $f: V^2 \rightarrow \mathbb{R}$ <u>max. flow</u> from s to t

> **Goal:** maximize
> $$|f| := \sum_{v:(s,v)\in E} f(s,v)$$

The <u>residual</u> G_f of a graph G with flow f

has edges $E_f := \{ (u,v) : A_{u,v} > f(u,v) \lor f(v,u) > 0 \}$

Edmonds-Karp: Initialize $f \equiv 0$.

shortest

Runtime $O(n \cdot m^2)$

While there exists *some* path $P = (s=u_1,\ldots u_K=t)$ from s to t in G_f

Let $\alpha := \min\{ A_{u_k,u_{k+1}} - f(u_k,u_{k+1}) : k=1\ldots K-1 \}$ and $f := f + \alpha \cdot P$.
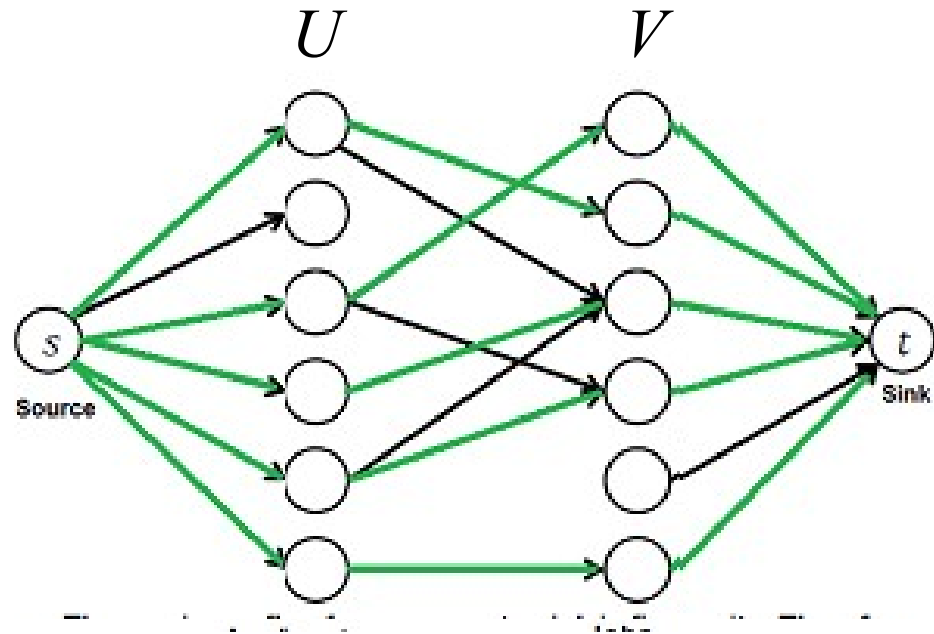
5. Graph Problems

max. Bipartite Matching

Specification: <u>Bipartite</u> graph $G=(U,V,E)$

Input: A_G adjacency/weight matrix $A_G \in \mathbb{N}^{\boxed{U \times V}}$

Output: $F \subseteq E$ max. (weighted) <u>matching</u>

Reduction to
max. weighted flow

U V

s Source   Sink t

Edmonds-Karp: Initialize $f \equiv 0$. $\boxed{\text{shortest}}$ $\boxed{\text{Runtime } O(n \cdot m^2)}$

While there exists some path $P = (s=u_1,\ldots u_K=t)$ from s to t in G_f

Let $\alpha := \min\{ A_{u_k,u_{k+1}} - f(u_k,u_{k+1}) : k=1\ldots K-1 \}$ and $f := f + \alpha \cdot P$.

5. Graph Problems

Specification: Graph $G=(V,E)$, $n=\#V$ vertices, $m=\#E$ edges

Input: $s,t \in V$, A_G adjacency/weight matrix $A_G \in \mathbb{N}^{V \times V}$
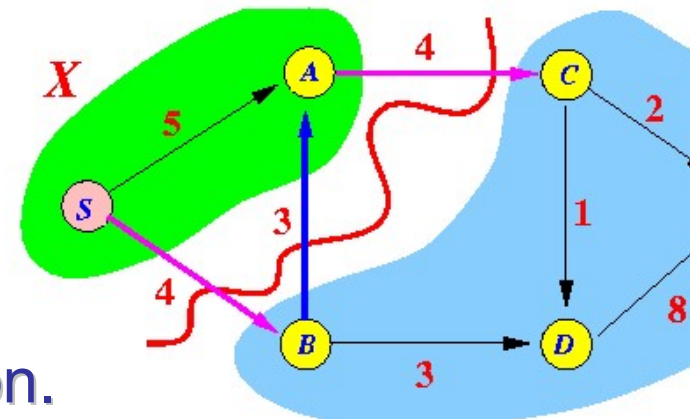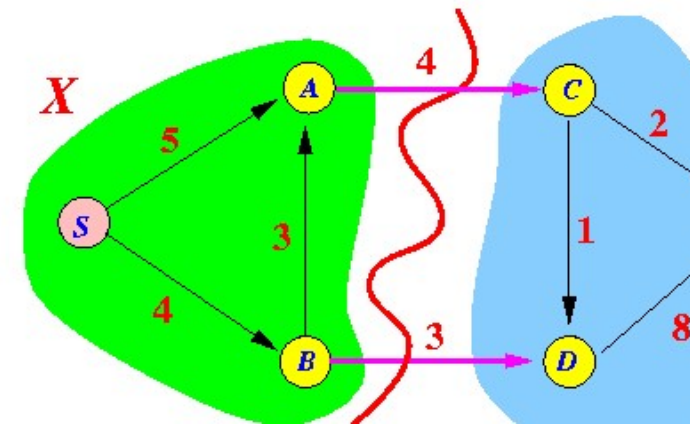
Output: $C \subseteq E$ <u>min.cut</u> between s,t

Def: A cut from s to t in G
is a subset $C \subseteq V$ s.t. $s \in C$, $t \notin C$.

It has capacity $\lambda(C) = \sum_{\substack{(u,v) \in E \\ u \in C, v \notin C}} A_{u,v}$

Theorem: $\min_{C \text{ cut } (s,t)} \lambda(C) = \max_{f \text{ flow } (s,t)} |f|$

Proof "≥": For every C,f: $\lambda(C) \geq |f|$.

"≤": Consider $C \subseteq V$ all vertices reachable
from s in G_f for max. f from Ford-Fulkerson.

Summary

5. Graph Problems

- – Recap on Graphs: un/directed, weighted

- – Shortest Paths: single-source, all-pairs

- – Minimum Spanning Tree: Prim, Kruskal

- – Maximum Flow: Ford-Fulkerson, Edmonds-Karp

- – Maximum (weighted) Bipartite Matching

- – Minimum Cut

- – Planarity Testing, Maximum Matching → CS500