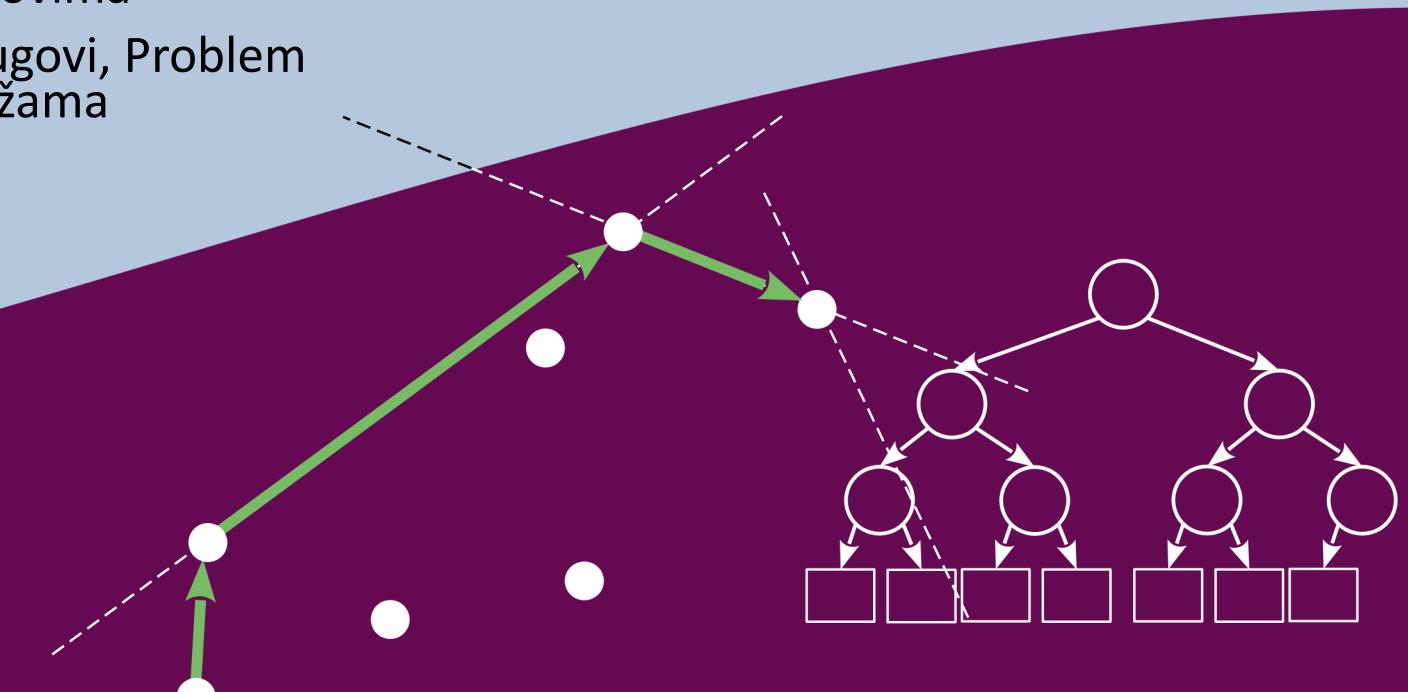
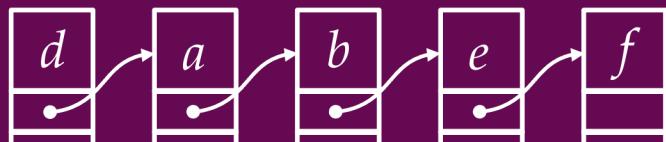


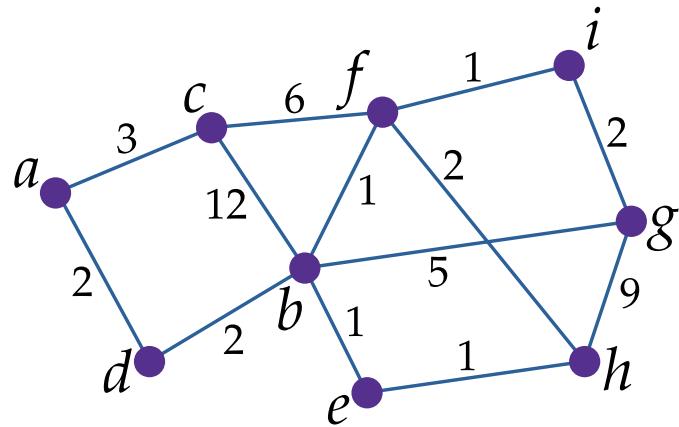
Napredni algoritmi i strukture podataka

11. predavanje: Algoritmi nad grafovima

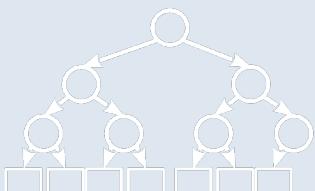
Problem kineskog poštara, Hamiltonovi krugovi, Problem trgovačkog putnika, Protok u mrežama



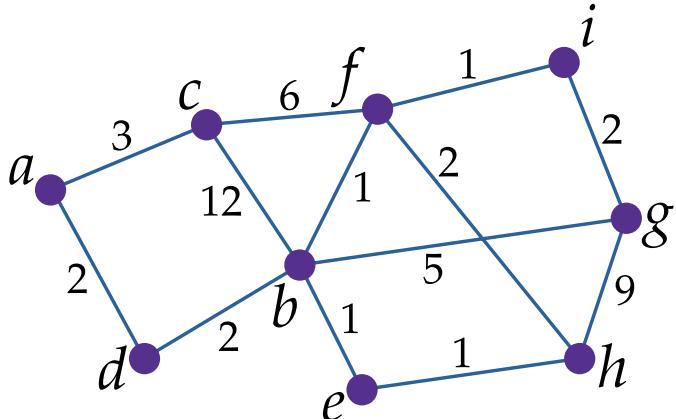
Problem kineskog poštara



- Problem kineskog poštara sličan je problemu sedam mostova
- Kineski poštar treba obići naselje modelirano težinskim grafom na najefikasniji mogući način
 - Bridovi predstavljaju ulice naselja, a težine na bridovima trošak (trud, *effort*) koji poštar mora uložiti za prelazak te ulice
 - Nakon obilaska naselja, poštar se vraća u vrh iz kojeg je krenuo
- Ako je naselje modelirano Eulerovim grafom koji sadrži Eulerov krug, tada je stvar jasna – Uvijek postoji točno jedan obilazaka
- Ako naselje nije modelirano Eulerovim grafom, tada imamo optimizacijski problem koji treba riješiti



Problem kineskog poštara



- Već smo utvrdili da Eulerov graf koji sadrži Eulerov krug mora imati sve vrhove parnog stupnja
- Ako imamo neki od vrhova koji imaju neparan stupanj, tada graf naselja moramo *Eulerizirati*, kako bismo dobili Eulerov krug
- Koncept Eulerizacije znači dodavanje bridova u graf tako da svi vrhovi grafa imaju paran stupanj



Eulerizacija grafa

- 1: Detect all vertices having an odd degree $ODD \subseteq V(G)$.
- 2: Calculate distances all-to-all between ODD vertices. You can use WFI (algorithm 4.9) for this purpose. We define the mapping set of shortest paths between pairs of vertices in ODD as

$$\forall v_i, v_j \in ODD : ODDPaths(v_i, v_j) = v_i v_{i+1} \dots v_{j-1} v_j$$

- 3: Construct a complete bipartite graph

$$X = ODD, Y = ODD$$

$$H = (X \cup Y, E')$$

such that weights of the bipartite graph edges are as:

$$w(v_{ii}) = \infty,$$

$$w(v_{ij}) = w_{ij}$$

where w_{ij} is the shortest distance between vertices $v_i, v_j \in ODD$, calculated in the step 2.

- 4: In the complete bipartite graph H find the optimal assignments M between vertices in X and Y , such that

$$M \subseteq X \times Y \Rightarrow \min \sum_{(v_i, v_j) \in M} w(v_i v_j)$$

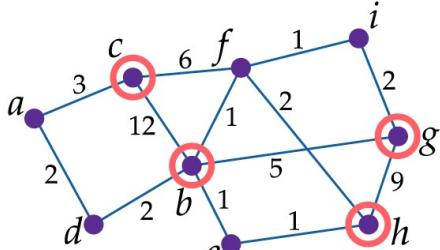
- 5: **for** $(v_i, v_j) \in M$ such that $\deg(v_i)$ is still odd **do**
- 6: **for** $v_r v_s \in ODDPaths(v_i, v_j)$ **do**
- 7: Add the edge $v_r v_s$ to the edges $E(G)$

Eulerizacija – korak 1 i 2

Step 1: Detecting the odd degree vertices

We start by detecting all vertices having an odd degree. We have the following vertices

$$ODD = \{b, c, g, h\}$$



Step 2: Using WFI, calculate shortest paths and distances in ODD

We get the following shortest distances

$$w(bc) = D_{bc}^9 = 7, w(bg) = D_{bg}^9 = 4$$

$$w(bh) = D_{bh}^9 = 2, w(cg) = D_{cg}^9 = 9$$

$$w(ch) = D_{ch}^9 = 8, w(gh) = D_{gh}^9 = 5$$

and shortest paths

$$ODDPaths(b, c) = bdac$$

$$ODDPaths(b, g) = bfig$$

$$ODDPaths(b, h) = beh$$

$$ODDPaths(c, g) = cfig$$

$$ODDPaths(c, h) = cfh$$

$$ODDPaths(g, h) = gifh$$

$$\mathbf{D}^0 = \begin{bmatrix} a & b & c & d & e & f & g & h & i \\ a & 0 & 0 & 3 & 2 & 0 & 0 & 0 & 0 \\ b & 0 & 0 & 12 & 2 & 1 & 1 & 5 & 0 \\ c & 3 & 12 & 0 & 0 & 0 & 6 & 0 & 0 \\ d & 2 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ e & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ f & 0 & 1 & 6 & 0 & 0 & 0 & 0 & 2 \\ g & 0 & 5 & 0 & 0 & 0 & 0 & 0 & 9 \\ h & 0 & 0 & 0 & 0 & 1 & 2 & 9 & 0 \\ i & 0 & 0 & 0 & 0 & 0 & 1 & 2 & 0 \end{bmatrix}$$

$$\mathbf{D}^9 = \begin{bmatrix} a & b & c & d & e & f & g & h & i \\ a & 0 & 4 & 3 & 2 & 5 & 5 & 8 & 6 \\ b & 4 & 0 & 7 & 2 & 1 & 1 & 4 & 2 \\ c & 3 & 7 & 0 & 5 & 8 & 6 & 9 & 8 \\ d & 2 & 2 & 5 & 0 & 3 & 3 & 6 & 4 \\ e & 5 & 1 & 8 & 3 & 0 & 2 & 5 & 1 \\ f & 5 & 1 & 6 & 3 & 2 & 0 & 3 & 2 \\ g & 8 & 4 & 9 & 6 & 5 & 3 & 0 & 5 \\ h & 6 & 2 & 8 & 4 & 1 & 2 & 5 & 0 \\ i & 6 & 2 & 7 & 4 & 3 & 1 & 2 & 3 \end{bmatrix}$$

- U prvom koraku pronađemo sve vrhove s neparnim stupnjem
- U drugom koraku izračunavamo najkraće udaljenosti svi-svi unutar skupa ODD
 - Uz to određujemo te najkraće putanje
 - U primjeru se koristio WFI, ima i drugih opcija

Eulerizacija – koraci 3,4 i 5

Step 3 and 4: Generate the bipartite graph H and find the optimal assignments M

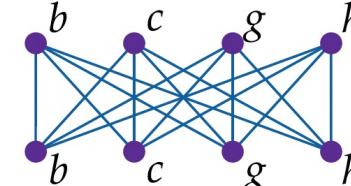
In this step we need to optimize the pairs of vertices of the complete bipartite graph H .

$$M_1 = \{(b,c), (g,h)\}, w(bc) + w(gh) = 7 + 5 = 12$$

$$M_2 = \{(b,g), (c,h)\}, w(bg) + w(ch) = 4 + 8 = 12$$

$$M_3 = \{(b,h), (c,g)\}, w(bh) + w(cg) = 2 + 9 = 11$$

The optimal assignment is M_3 , which is also giving the shortest Eulerian circuit in the final Eulerized graph.



b			c			g			h		
c	g	h	b	g	h	b	c	h	b	c	g
7	4	2	7	9	8	4	9	5	2	8	5

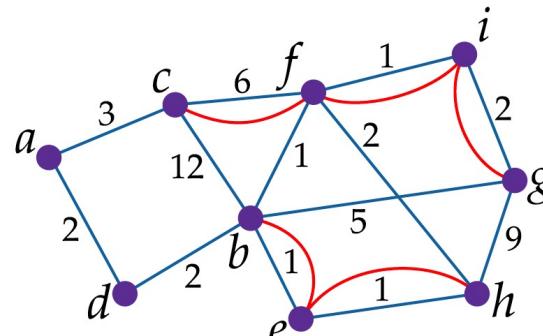
Step 5: Add edges from the shortest paths $ODDPaths$, according to the optimal assignments M

We iterate through M_3 to find the shortest paths edges.

1. $(b,h) \in M_3 : ODDPaths(b,h) = beh$. Edges be and eh are added to the graph G .

2. $(c,g) \in M_3 : ODDPaths(c,g) = cfig$. Edges cf , fi , and ig are added to the graph G .

Finally, we get the Eulerized graph as seen on the right side in which all vertices have even degrees.



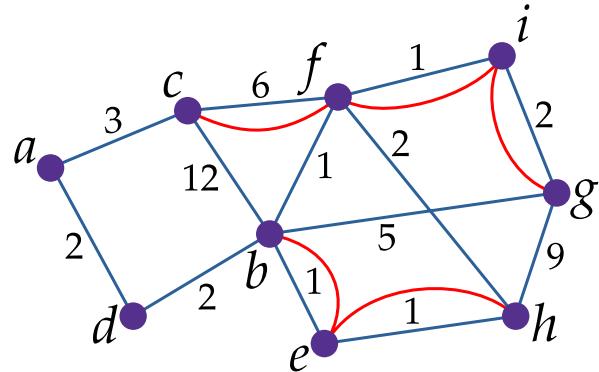
- Konstruiramo bipartitni graf iz kojeg izvlačimo optimalne parove, koji imaju minimalnu sumu težina putanja

- Optimalni put poštara
- Može se rješavati kroz tablicu

- Na kraju iz $ODDPaths$ uzmemu putanje za optimalne parove

- Dodajemo paralelne bridove po putanjama
- Početni vrh u putanji dobije samo jedan priležeći brid – čime mu stupanj postaje paran
- Međuvrhovi u putanji dobiju dva priležeća brida, čime im se ne mijenja parnost/neparnost trenutnog stupnja

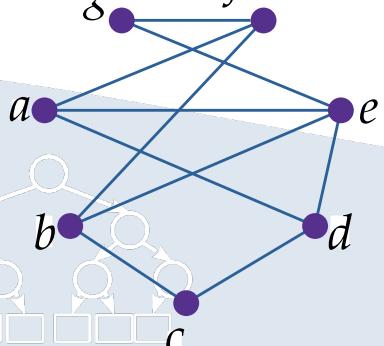
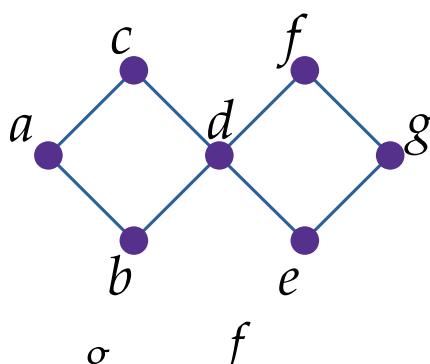
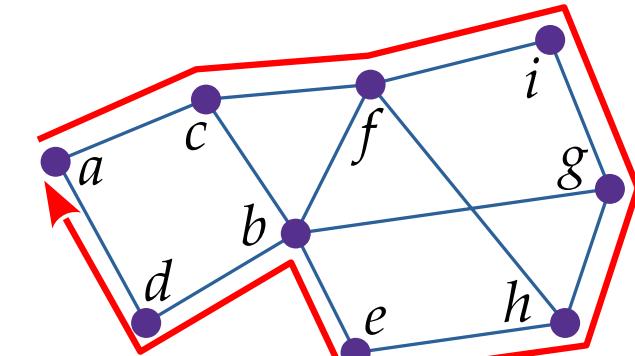
Problem kineskog poštara



- Nakon Eulerizacije na graf primijenimo neki od algoritama za detekciju Eulerovog kruga
 - Vizualno je Fleuryev algoritam jednostavniji
- Pronađe se Eulerov krug koji počinje i završava u vrhu iz kojeg kineski poštari kreće u obilazak
- Ukupni zbroj težina po svim bridovima Eulerovog kruga je ukupni trošak obilaska kineskog poštara
 - Svi nadodani bridovi u postupku Eulerizacije imaju istu težinu kao i originalni brid između dva vrha – radi se naime o jednoj te istoj ulici
 - Na primjer, originalni brid i nadodani brid cf su oba težine 6.

Hamiltonovi grafovi i ciklusi

- **Hamiltonov put (path)** – je put kroz graf koji uključuje sve vrhove grafa samo jednom
- **Hamiltonov ciklus (cycle)** – je Hamiltonov put koji počinje i završava u istom vrhu
- **Hamiltonov graf** – je graf koji je sačinjen od Hamiltonovog ciklusa
- Bitno je uočiti da za razliku od Eulerovog kruga, u Hamiltonovom ciklusu kroz svaki vrh možemo proći samo jednom i taj ciklus mora uključivati sve vrhove grafa – Hamiltonovog grafa
- Problem je NP-kompleksan

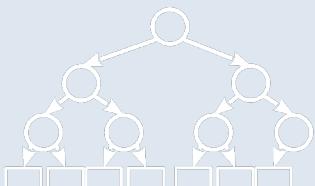


Zatvarač grafa

- Ako imamo graf G , tada je G' zatvarač (*closure*) grafa G koji se dobije

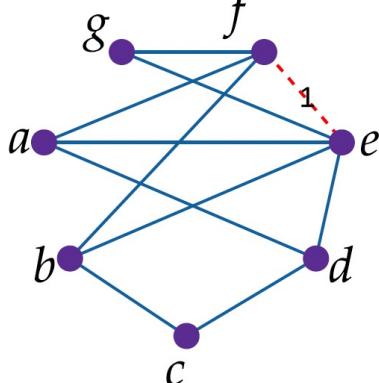
```
 $G' \leftarrow G$ 
 $i \leftarrow 1$ 
while there are vertices  $u$  and  $v$ , such that  $\deg(u) + \deg(v) \geq |V(G)|$  do
    add the edge  $uv$  to the edges  $E(G')$ , mark it as the  $i$ -th edge
     $i \leftarrow i + 1$ 
```

- Dodajemo brid na neki par vrhova uv , čiji je zbroj stupnjeva veći ili jednak broju vrhova u grafu ili $\deg(u) + \deg(v) \geq |V|$
 - Dodani brid označimo rednim brojem
- U trenutku kada više ne možemo dodati niti jedan brid, to se smatra zatvaračem G'

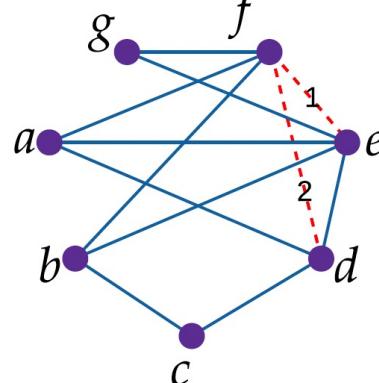


Zatvarač grafa

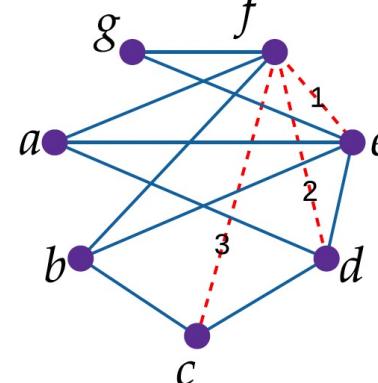
Iteration 1: $\deg(f) + \deg(e) = 7$



Iteration 2: $\deg(f) + \deg(d) = 7$

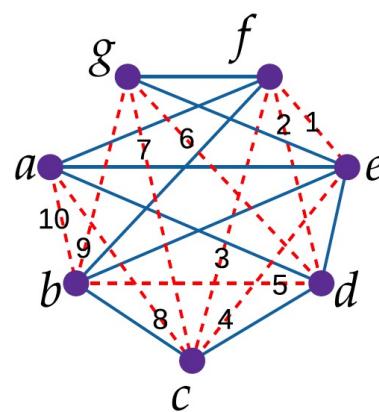


Iteration 3: $\deg(f) + \deg(c) = 7$

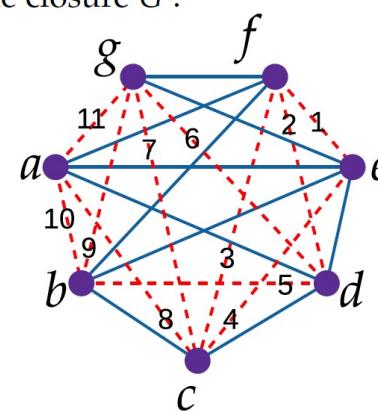


We continue to add edges for pairs of vertices whose degrees comply the requirement $\deg(u) + \deg(v) \geq |V|$.

Iteration 10: $\deg(a) + \deg(b) = 9$

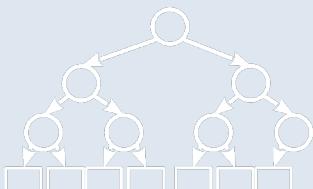


Iteration 11: $\deg(a) + \deg(g) = 10$. This is also the last step and the closure G' .

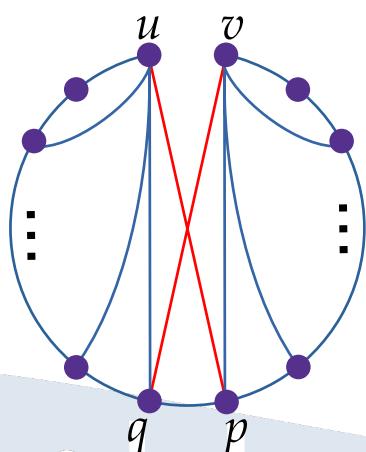
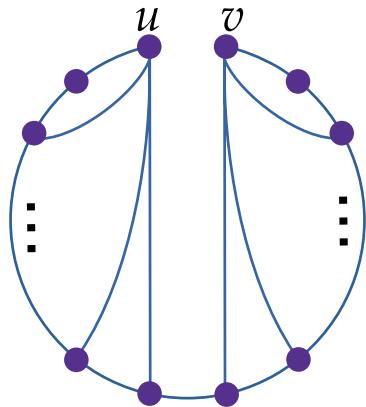


Bondy–Chvátal teorem

- **Teorem:** Jednostavan neusmjereni graf G je Hamiltonov samo i samo ako je njegov zatvarač G' Hamiltonov graf.
- **Dokaz 1:** Ako je jednostavan neusmjereni graf G Hamiltonov, tada postoji podskup bridova $E' \subseteq E(G)$ takav da čini Hamiltonov ciklus C_H . Zatvarač G' stvoren je dodavanjem bridova na graf G , čime vrijedi $E' \subseteq E(G) \subseteq E(G')$, što znači da je C_H sadržan i u zatvaraču G' .
- **Dokaz 2:** Između grafa G i njegovog zatvarača G' imamo slijed grafova
$$G_0, G_1, G_2, \dots, G_{k-1}, G_k$$
 - gdje su $G = G_0$ i $G' = G_k$
 - i svaki graf G_i ima jedan brid više u odnosu na graf G_{i-1}



Bondy–Chvátal teorem



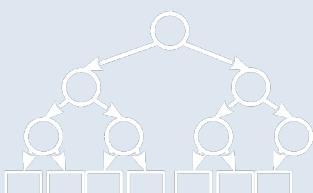
- Po Bondy–Chvátal teoremu, ako graf G_i sadrži Hamiltonov ciklus, tada ga sadrži i graf G_{i-1} . Ovo podliježe principu matematičke indukcije.
- Na gornjoj slici imamo graf G_{i-1} . Kako bismo dobili Hamiltonov ciklus, trebamo dodati brid uv i time dobiti G_i koji ga sadrži
 - Ako je $|V(G_{i-1})| = n$, tada vrijedi $\deg(u) + \deg(v) = n - 2$
 - Zbog toga ne možemo dodati brid uv
 - Da bismo dodali brid uv , moramo iz vrhova u i v imati dva dodatna brida, koji su na donjoj slici označeni kao up i vq
 - Time smo dobili $\deg(u) + \deg(v) = n$
 - Dodavanjem brida uv , u G_i imamo Hamiltonov ciklus C_H
 - Uočavamo da u G_{i-1} imamo Hamiltonov ciklus koji je $C_H - pq + up + vq$

QED

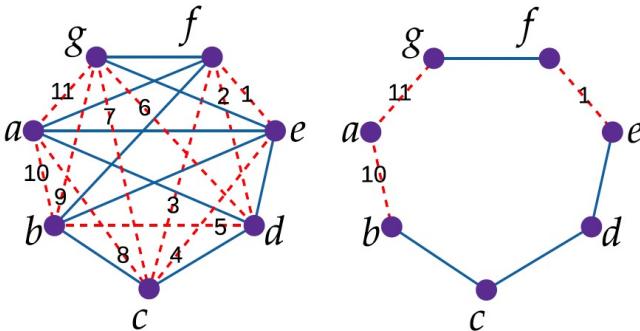
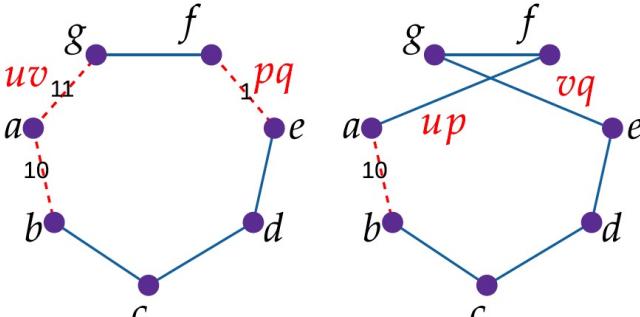
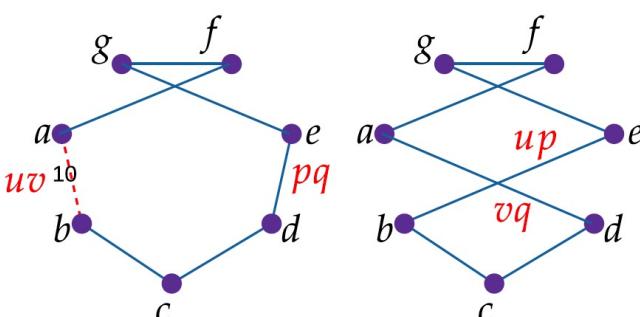
Praktična primjena Bondy–Chvátal teorema

- 1: $G_c \leftarrow$ Hamiltonian cycle in the closure G'
- 2: **while** there is added edge marked with ordinal in the graph G_c
do
- 3: pick an added edge uv from G_c having the highest ordinal i_{max}
- 4: pick an edge $pq \neq uv$ from G_c such that edges up and vq are from G' , crossed, and either from the original graph G or having ordinal less than i_{max}
- 5: remove edges uv and pq from the graph G_c
- 6: add edges up and vq to the graph G_c

- Vraćamo se od zatvarača G' natrag
 - U zatvaraču smo pronašli Hamiltonov ciklus G_c
 - U G_c uzimamo nadodani brid s najvišim rednim brojem kao brid uv – sjetimo se da smo ih u postupku stvaranja zatvarača G' označavali
 - U G_c odaberemo brid pq , takav da postoje prekriženi bridovi up i vq u G' koji su ili iz grafa G ili imaju manji redni broj od brida uv
 - U G_c uklonimo bridove uv i pq , a dodamo bridove up i vq
 - Ovo radimo sve do dok u G_c imamo nadodanih bridova – kada ih nema, KRAJ, pronašli smo Hamiltonov ciklus u grafu G

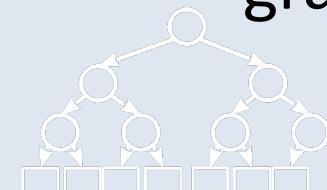


Praktična primjena Bondy–Chvátal teorema

Step 1: Detect the Hamiltonian cycle in the closure G'	
We start by detecting the Hamiltonian cycle in the closure G' .	
Step 2: Replace edges ag and fe .	
We find ag as the edge having the highest ordinal $i_{max} = 11$, which is identified as the edge uv . The other edge fe is taken as the edge pq . We take crossed edges from the G' , which are both from the original graph G .	
Step 3: Replace edges ab and ed .	
We find ab as the edge having the highest ordinal $i_{max} = 10$, which is identified as the edge uv . The other edge ed is taken as the edge pq . Taking the edge ed is convenient, as we have crossed edges ad and be from the original graph G . After this step, we have no more added edges, which means we moved backwards to the graph $G = G_0$. The result is the Hamiltonian cycle in the original graph G .	

Problem putujućeg trgovca (TSP)

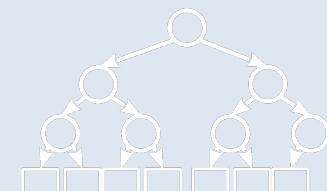
- Trgovac kreće iz određene točke i treba posjetiti sve svoje *kupce*
- Vrhovi grafa predstavljaju mjesta koja trgovac obilazi – uključujući i polazno/završno mjesto
 - Polazni vrh je ujedno i završni vrh – možemo ga gledati kao trgovčeve matično poduzeće
- Bridovi između vrhova imaju težine koje označavaju *trošak* putovanja od jednog kupca do drugog
 - Ovo možete zamisliti kao skup svih troškova koje trgovac treba uložiti za putovanje između dva kupca, recimo: vrijeme (broj dana putovanja), udaljenost, troškovi prijevoza, itd...
- Potrebno je pronaći najefikasniji put kojim trgovac obilazi sve kupce u grafu – za trgovca to znači najmanji trošak obilaska



Problem putujućeg trgovca

- Za razliku od kineskog poštara, problem putujućeg trgovca fokusiran je na vrhove grafa
 - Imamo vrh koji predstavlja vlastitu prodajnu agenciju – početni/završni vrh
 - Svaki vrh je jedan kupac koji se obilazi točno jednom
 - Brid, to jest put između dva kupca, se može koristiti proizvoljan broj puta
- Uočavamo da je za rješenje ovog problema potrebno pronaći najoptimalniji Hamiltonov ciklus u grafu
 - Pretpostavimo da u grafu imamo više Hamiltonovih ciklusa
 - Potrebno je pronaći onaj čija je suma težina svih bridova najmanja

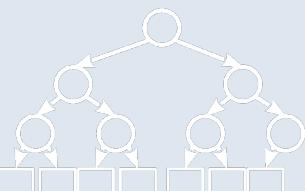
$$C_{H_{opt}} = \arg \min_{C_H} \sum_{e \in E(C_H)} w(e)$$



2-MST Heuristika

- 1: Construct the minimal spanning tree G_{MST} from the graph G .
- 2: Eulerize the minimal spanning tree G_{MST} by doubling all edges.
- 3: Find the Eulerian circuit C_E in Eulerized G_{MST} .
- 4: Turn the Eulerian circuit C_E into a Hamiltonian cycle C_H by skipping already visited vertices and jumping to the first unvisited.

- Metoda rješavanja problema putujućeg trgovca
- 1. Korištenjem adekvatnog algoritma stvorimo minimalno razapinjujuće stablo
- 2. Euleriziramo minimalno razapinjujuće stablo
- 3. Pronađemo Eulerov ciklus koristeći zadani početni/završni vrh
- 4. Pronađemo Hamiltonov ciklus iz Eulerovog ciklusa
 - Recimo DFS-om prolazimo kroz min.raz. stablo, a u trenutku kada se vraćamo na već obiđeni vrh, preskačemo na prvi neobiđeni
 - Iz Eulerovog ciklusa izbacujemo duple vrhove, a bridove između novih parova vrhova uzimamo iz originalnog grafa



2-MST Heuristika

Step 1: Construct the minimal spanning tree G_{MST}	
We start by constructing the minimal spanning tree G_{MST} from the graph G . For this purpose we can use any of the MST algorithms.	
Step 2: Eulerize the minimal spanning tree G_{MST} .	
Eulerize the minimal spanning tree by doubling all edges.	
Step 3: Find the Eulerian circuit C_E .	
Use any of the Eulerian circuit finding algorithms. Respect the starting vertex of the salesman. Notice that some of the vertices (marked red) are repeating in the Eulerian circuit. These vertices are not according to the definition of the Hamiltonian cycle.	
Step 4: Convert the Eulerian circuit C_E to the Hamiltonian cycle C_H .	
Remove all repeating vertices in the Eulerian circuit C_E . Connect the first two non-repeating vertices by using their edge (we need its weight) from the input graph G . In this example we removed repeating vertices e, c, and b, and connected d and a directly using its original edge of weight 3.	

Protok u mrežama

- Mreže se mogu modelirati kao grafovi, gdje:
 - Vrhovi predstavljaju mjesta, sjecišta, stanice
 - Bridovi predstavljaju spojne vodove, kao na primjer cijevi, trake, žice, to jest sve što podržava tok materijala, artikala, simbola
- Problem: pokušajmo izračunati najveći mogući protok vozila i opterećenje prometnica od Karlovca do Splita
 - Recimo da kapacitet prometnice definiramo brojem vozila po jedinici vremena koje ta prometnica propušta
- Slični problemi su:
 - Vodovodna mreža, elektroenergetska mreža (i druge distribucijske mreže), komunikacijsko-računalne mreže, evakuacijski planovi
- Tok može biti:
 - Kontinuiran – Recimo fluidi (plin, tekućina) ili krutine (m^3/s), električna struja (A) i slično
 - Diskretan – Uvijek postoji broj nekih artikala u jedinici vremena, recimo broj vozila, paketa, ljudi ili slično
- U mreži uvijek imamo vrhove koji generiraju tok (izvori - *sources*) i vrhove koji konzumiraju taj tok (odredišta - *destinations*)

Protok u mrežama

- Mreža s protokom (*flow network*) je spojeni usmjereni graf bez petlji i povratnih bridova

$$G = (V, E, c, f, s, d)$$

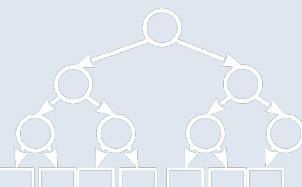
- gdje je c funkcija mapiranja kapaciteta na bridove mreže, f je funkcija mapiranja toka na bridove mreže, $s \in V$ je vrh koji generira tok, a $d \in V$ je vrh koji konzumira tok
- Kada kažemo da nema povratnih bridova, smatra se

$$\forall uv \in E \nexists vu \in E$$

- Funkcija mapiranja kapaciteta je

$$c: E \rightarrow \mathbb{R}^+$$

- Svaki brid mreže G ima naznačeni pozitivni kapacitet, što predstavlja maksimalni mogući tok kroz taj brid



Protok u mrežama

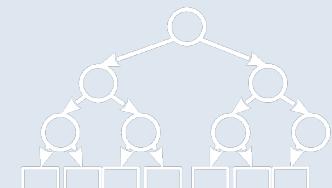
- Funkcija mapiranja toka je

$$f: E \rightarrow \mathbb{R}^+$$

- koristi se za mapiranje vrijednosti pozitivnog toka na brid mreže.

- Pomoćno mapiranje bridova koje će nam koristiti kod razrade protoka u mrežama

$$\begin{aligned}V_i, V_j &\subseteq V \\E(V_i, V_j) &= \{uv : uv \in E, u \in V_i, v \in V_j\}\end{aligned}$$



Protok u mrežama

- Tok f ima sljedeća svojstva:

- **Ograničenje kapacitetom:**

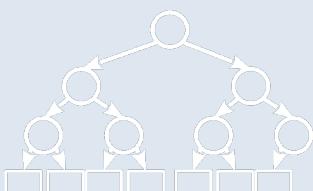
$$\forall uv \in E: 0 \leq f(uv) \leq c(uv)$$

- Maksimalni tok u nekom bridu mreže jednak je kapacitetu tog brida

- **Očuvanje toka:**

$$\forall u \in V \setminus \{s, d\}: \sum_{v_i u \in E(V, u)} f(v_i u) = \sum_{uv_j \in E(u, V)} f(uv_j)$$

- Za svaki vrh koji ne generira ili ne konzumira tok vrijedi da je ulazni tok jednak izlaznom toku
 - Ova ravnoteža ne vrijedi za vrhove koji generiraju i konzumiraju tokove
 - Očito je da za vrh koji generira tok, izlazni tok mora biti veći od ulaznog
 - Za vrh koji konzumira tok ulazni tok mora biti veći od izlaznog

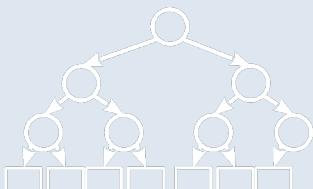


Protok u mrežama

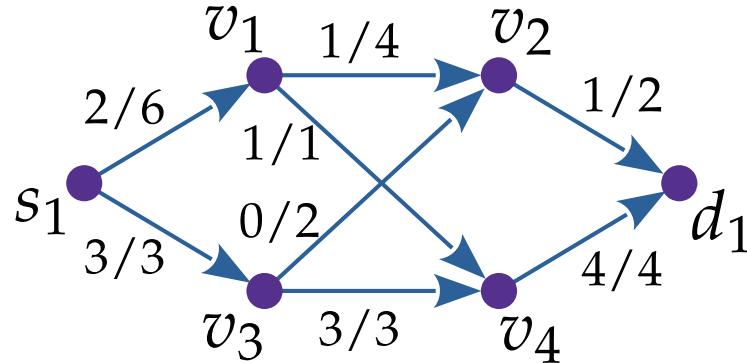
- **Jakost toka** definirana je kao

$$|f| = \sum_{sv_i \in E(s,V)} f(sv_i) - \sum_{v_j s \in E(V,s)} f(v_j s)$$

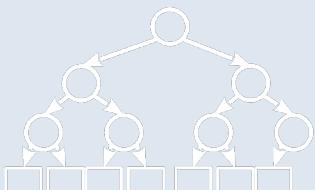
- gdje je s vrh generator toka
- Kao što je već naglašeno, za vrh koji generira tok, izlazni tok mora biti veći od ulaznog, a sama razlika predstavlja jakost toka
 - Kako bismo pojednostavnili, koristiti će se samo jedan vrh koji generira tok $s \in V$ i jedan vrh koji konzumira tok $d \in V$
 - Očito je da ako imamo više generatora toka, tada svaki vrh koji generira tok doprinosi ukupnoj jakosti toka $|f|$



Protok u mrežama



- U primjeru mreže imamo
 - vrh s_1 koji generira tok
 - vrh d_1 koji konzumira tok
 - svaki brid ima oznaku
$$f(uv)/c(uv)$$
- Da li je tok kroz mrežu maksimalan mogući?

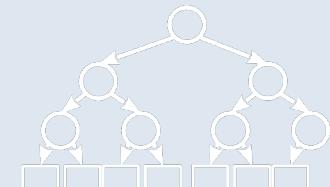


Ford-Fulkerson algoritam

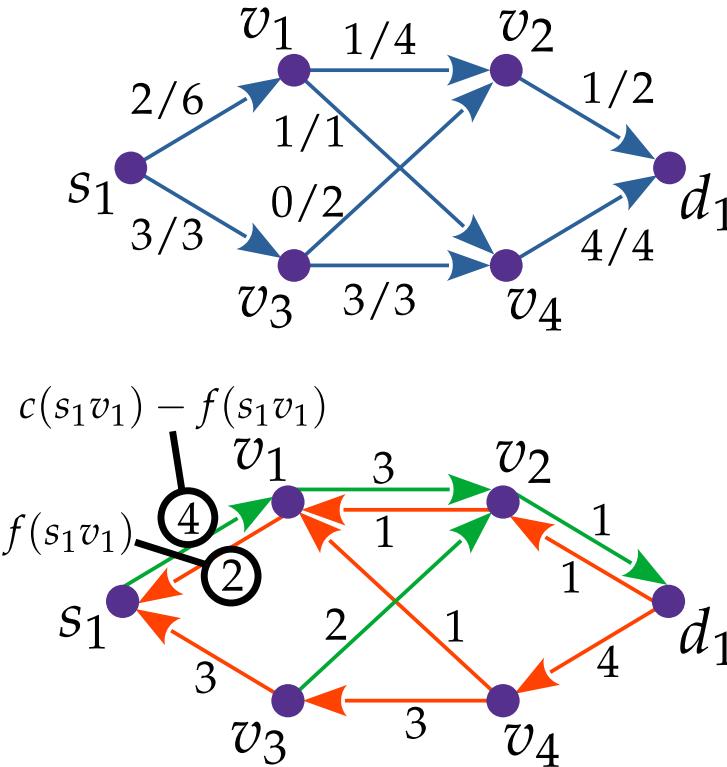
- **Ostatna mreža:** S obzirom na mrežu G , može se konstruirati ostatna mreža (*residual network*)

$$G_f = (V, E_f, c_f, f_f, s, d)$$

- gdje su E_f bridovi ostatne mreže i nisu isti kao i bridovi originalne mreže E
- c_f je funkcija mapiranja ostatnog kapaciteta na brdove ostatne mreže
- f_f je funkcija mapiranja ostatnog toka na brdove ostatne mreže
- Bridovi ostatne mreže mogu biti:
 - Unaprijedni (*forward*): $E_{f_1} = \{uv: uv \in E, c(uv) - f(uv) > 0\}$
 - Povratni (*reverse*): $E_{f_2} = \{vu: uv \in E, f(uv) > 0\}$
 - Tako da vrijedi: $E_f = E_{f_1} \cup E_{f_2}$



Ford-Fulkerson algoritam



- **Ostatni kapacitet** (*residual capacity*) na ostanjoj mreži G_f definiran je funkcijom mapiranja

$$c_f(uv) = \begin{cases} c(uv) - f(uv), & \exists uv \in E, c(uv) - f(uv) > 0 \\ f(vu), & \exists vu \in E, f(vu) > 0 \\ 0, & \text{otherwise} \end{cases}$$

- Za unaprijedne bridove vrijedi da je ostatni kapacitet razlika kapaciteta i toka u originalnoj mreži G
- Za povratne bridove vrijedi da je ostatni kapacitet jednak toku suprotnog brida u originalnoj mreži G

Ford-Fulkerson algoritam

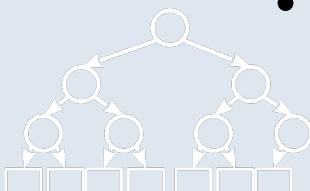
- Slično kao i kod originalne mreže G , ostatna mreža G_f može imati tokove, definirane funkcijom mapiranja f_f
- Za tokove u ostatnoj mreži definiramo pojam **uvećanja toka** (*flow augmentation*) kao

$$f \uparrow f_f(uv) = \begin{cases} f(uv) + f_f(uv) - f_f(vu), & \exists uv \in E \\ 0, & \text{otherwise} \end{cases}$$

- gdje se tok u originalnoj mreži f uvećava za tok u ostatnoj mreži f_f
- f_f naziva se i **uvećavajući tok** (*augmenting flow*)
- Slično vrijedi i za **uvećanje jakosti toka** (*flow strength augmentation*)

$$|f \uparrow f_f| = |f| + |f_f|$$

- dokaz provjerite u skripti



Ford-Fulkerson algoritam

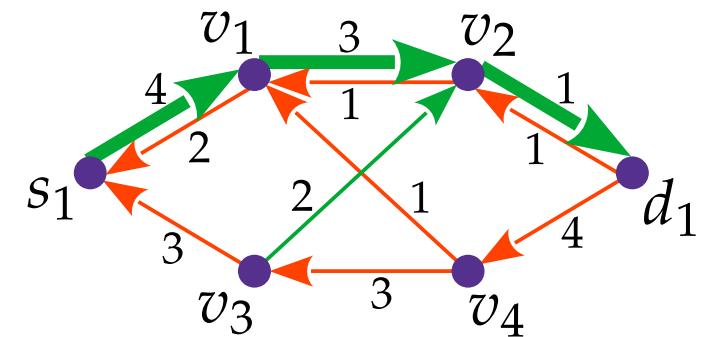
- **Uvećavajući put** (*augmenting path*): S obzirom na mrežu G i njenu ostatnu mrežu G_f , bilo koji put p od vrha s do vrha d naziva se uvećavajući put

- Svaki takav put daje nam mogućnost definiranja uvećavajućeg toka f_f na ostatnoj mreži G_f , a time i uvećanje toka u originalnoj mreži $f \uparrow f_f$
- Ostatni kapacitet uvećavajućeg puta p je

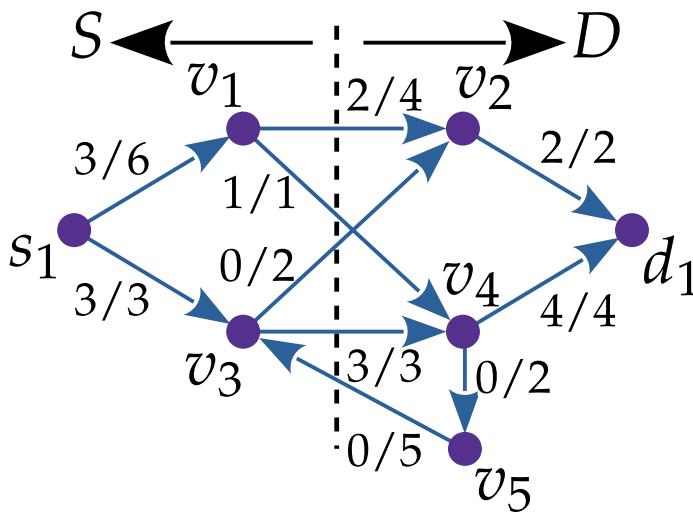
$$c_f(p) = \min_{\forall uv \in p} c_f(uv)$$

- čime se uvećavajući tok f_f na putu p ograničava na minimalni ostatni kapacitet na tom putu
- Uvećavajući tok na putu p ili f_{f_p} računa se kao
$$\forall uv \in p: f_{f_p}(uv) = c_f(p)$$
- Uvećanje toka u originalnoj mreži rezultira

$$|f \uparrow f_{f_p}| = |f| + |f_{f_p}| > |f|$$

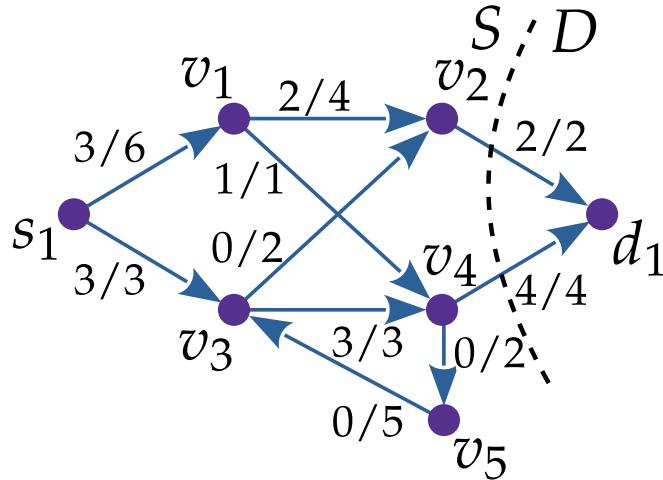


Presjek mreže



- Svaki mrežu možemo podijeliti na način da podijelimo vrhove mreže u dvije disjunktne particije
 $S \subset V, D \subset V$
 - tako da vrijedi $s \in S, d \in D, S \cap D = \emptyset$ i $S \cup D = V$
- Time smo definirali presjek mreže S, D kroz koji prolaze bridovi $E(S, D)$.
- Kapacitet presjeka mreže računa se
$$c(S, D) = \sum_{uv \in E(S, D)} c(uv)$$
- Tok presjeka mreže računa se
$$f(S, D) = \sum_{uv \in E(S, D)} f(uv) - \sum_{vu \in E(D, S)} f(vu)$$
- Zbog svojstva očuvanja toka vrijedi $|f| = f(S, D)$ (dokaz vidi u skripti)

Minimalni presjek mreže (min-cut)



- Presjek mreže S, D koji ima najmanji kapacitet $c(S, D)$ naziva se minimalnim presjekom (*minimal cut*) i dobiva se optimizacijom

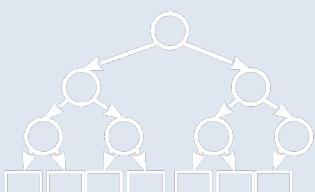
$$c_m(S, D) = \min_{S, D} c(S, D)$$

- Korolar:** Ograničenje kapacitetom (*network cut capacity constraint*) se može primijeniti na bilo koji presjek mreže S, D , čime vrijedi

$$0 \leq f(S, D) = |f| \leq c(S, D)$$

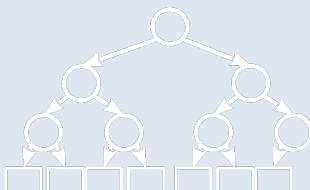
- Dokaz:**

$$\begin{aligned}|f| &= f(S, D) = \sum_{uv \in E(S, D)} f(uv) - \sum_{vu \in E(D, S)} f(vu) \\&\leq \sum_{uv \in E(S, D)} f(uv) \leq \sum_{uv \in E(S, D)} c(uv) = c(S, D)\end{aligned}$$



Max-flow min-cut teorem

- S obzirom na mrežu G koja ima tok f , generatora toka s , konzumera toka d , vrijede sljedeće tvrdnje:
 1. f je maksimalni tok u mreži G ,
 2. u ostatnoj mreži G_f nema više uvećavajućih puteva,
 3. postoji presjek mreže S, D takav da vrijedi $|f| = c(S, D)$
- **Dokaz (1 → 2):** Pretpostavimo da ne vrijedi 2., to jest postoji uvećavajući put p u ostatnoj mreži G_f . To znači da postoji tok f_{fp} u ostatnoj mreži G_f koji može uvećati tok f u originalnoj mreži G . Time dobivamo $|f \uparrow f_{fp}| > |f|$, što je u direktnoj kontradikciji sa 1.



Max-flow min-cut teorem

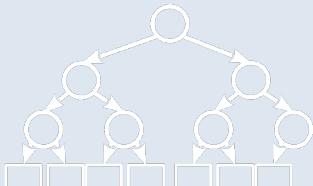
- **Dokaz (2 → 3):** Ako ne postoji uvećavajući put p u ostatnoj mreži G_f , tada uopće ne postoji put između vrhova s i d , što znači da postoji presjek S, D za koji ne postoji niti jedan brid između vrhova S i D u ostatnoj mreži G_f , što znači da vrijedi

$$\forall uv \in E(S, D) \nexists uv \in E_f \Rightarrow c_f(uv) = 0, c(uv) = f(uv)$$

S obzirom na prethodni korolar, za presjek S, D vrijedi

$$|f| = f(S, D) = c(S, D)$$

što je identično tvrdnji 3.



Max-flow min-cut teorem

- **Dokaz (3 → 1):** Korolar je direktni dokaz, to jest

$$|f| \leq c(S, D)$$

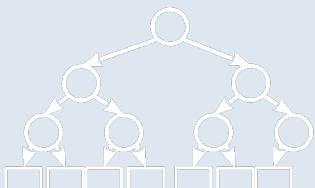
te vrijedi da je presjek kod kojeg vrijedi

$$|f| = c(S, D)$$

maksimalan tok u mreži

- Zbog ovog teorema vrijedi da je maksimalni tok jednak minimalnom presjeku mreže

$$|f| = c_m(S, D)$$

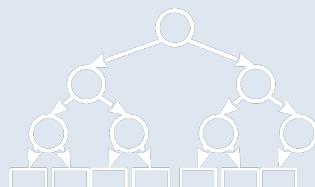


Ford-Fulkerson algoritam

```
procedure FORDFULKERSON( $G$ )
  for each edge  $uv$  in  $E(G)$  do
     $f(uv) \leftarrow 0$ 
    calculate the residual network  $G_f$                                 ▷ Def. 4.44
    while there is an augmenting path  $p$  in  $G_f$  do
      calculate  $f_{f_p}$                                               ▷ (4.99)
      perform the flow augmentation  $(f \uparrow f_{f_p})$                   ▷ (4.93)
      calculate the residual network  $G_f$                                 ▷ Def. 4.44
```

- Kompleksnost ovog algoritma je $O(f_m + E)$
- Gdje je f_m broj uvećavajućih puteva

- Inicijaliziramo osnovnu mrežu G postavljajući tokove na 0
- Izračunamo ostatnu mrežu G_f
 - Pronađemo neki uvećavajući put p u ostatnoj mreži G_f
 - Izračunamo uvećavajući tok f_{f_p}
 - Uvećamo tok u osnovnoj mreži G
 - Ažuriramo ostatnu mrežu G_f
 - Ponavljamo tako dugo dok postoji put između s i d u ostatnoj mreži G_f

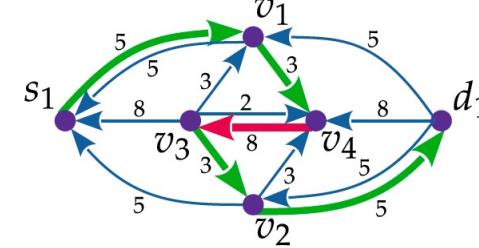
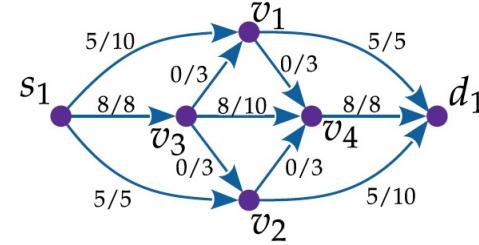


Ford-Fulkerson algoritam

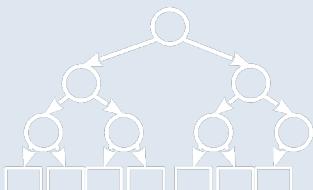
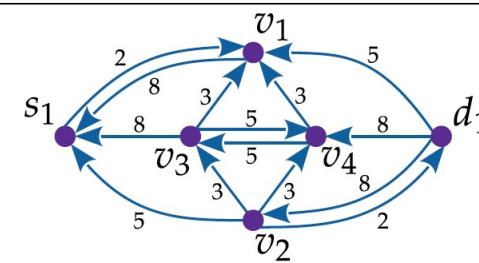
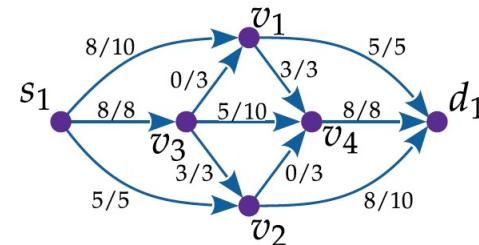
Step	G	G_f
1: The initial residual network has the residual capacities equal to the original flow network capacities. We choose the augmenting path $p = s_1v_1d_1$, which has the minimal residual capacity $c_f(p) = 5$. The augmenting path flow strength is therefore $ f_{fp} = 5$.		
2: We choose the augmenting path $p = s_1v_3v_4d_1$, which has the minimal residual capacity $c_f(p) = 8$. The augmenting path flow strength is therefore $ f_{fp} = 8$.		
3: We choose the augmenting path $p = s_1v_2d_1$, which has the minimal residual capacity $c_f(p) = 5$. The augmenting path flow strength is therefore $ f_{fp} = 5$.		

Ford-Fulkerson algoritam

4: We choose the augmenting path $p = s_1v_1v_4v_3v_2d_1$, which has the minimal residual capacity $c_f(p) = 3$. The augmenting path flow strength is therefore $|f_{f_p}| = 3$. Notice that we used reversed edge v_4v_3 from the residual network G_f .



5: We have no more available augmenting paths. The final maximal flow is $|f| = 21$.



Edmonds-Karp algoritam

- Nadogradnja Ford-Fulkerson algoritma kojom se biranje uvećavajućeg puta radi korištenjem BFS algoritma, tako da se prioritiziraju kraći putevi s obzirom na broj bridova.

