Akademia Górniczo-Hutnicza
im. Stanisława Staszica w Krakowie
AGH University of Science
and Technology

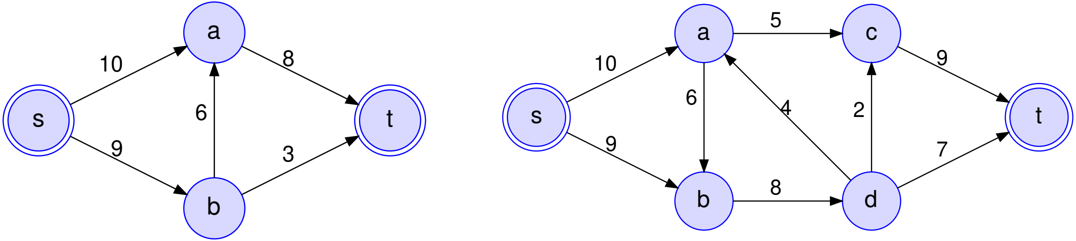
Grafy i ich zastosowania

Zestaw 5

Elzbieta.Strzalka@fis.agh.edu.pl
p. 232/D-10

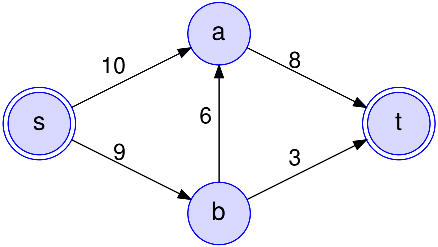
Sieć przepływowa – definicje

- **Sieć przepływowa** $\equiv$ digraf z nieujemną **przepustowością** (ang. *capacity*) $c(u, v)$ dla każdej krawędzi.
- Wierzchołki wyróżnione: jedno **źródło** s (ang. *source*) i jedno **ujście** t (ang. *target*).
- **Pozostałe wierzchołki leżą na ścieżkach z s do t** ($\Rightarrow$ sieć spójna, choć nie ściśle spójna).



Sieć transportowa, przepływ cieczy w rurociągach, przepływ prądu w sieciach elektrycznych, ...

Sieć przepływowa – definicje



Ale: każdy digraf niespełniający któregokolwiek z warunków można sprowadzić do sieci przepływowej.

Warunki sieci przepływowej

- Brak krawędzi **wchodzących** do s.
- Brak krawędzi **wychodzących** z t.
- Brak **pętli**.
- Pomiędzy dwoma wierzchołkami maksymalnie **jedna krawędź** (niezależnie od zwrotu)[a].

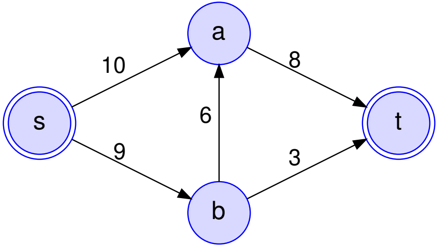
[a] Za: najnowsze wyd. *Wprowadzenia do algorytmów* Cormena; w starszych wydaniach krawędzie o przeciwnych zwrotach są dozwolone – zależy od definicji, jednak sieci te są równoważne.

Sieć przepływowa – definicje

www.agh.edu.pl

Przepływ $f(u, v)$ (*flow*)

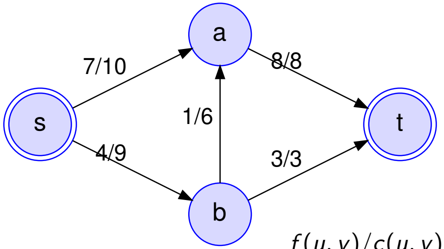
- **Przepustowość** $\equiv$ ile danym kanałem **można maksymalnie** przetransportować jednostek towaru (np. dziennie).
- **Przepływ** $\equiv$ ile **faktycznie** jednostek towaru przepływa danym kanałem.



Sieć przepływowa – definicje

Przepływ $f(u, v)$ (*flow*)

- **Przepustowość** $\equiv$ ile danym kanałem **można maksymalnie** przetransportować jednostek towaru (np. dziennie).
- **Przepływ** $\equiv$ ile **faktycznie** jednostek towaru przepływa danym kanałem.

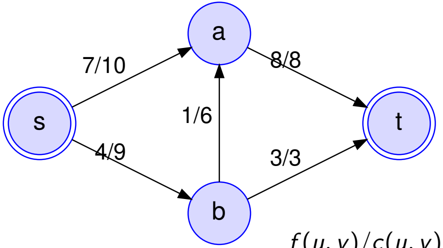


$f(u, v)/c(u, v)$

Sieć przepływowa – definicje

Przepływ $f(u,v)$ (*flow*)

- **Przepustowość** $\equiv$ ile danym kanałem **można maksymalnie** przetransportować jednostek towaru (np. dziennie).
- **Przepływ** $\equiv$ ile **faktycznie** jednostek towaru przepływa danym kanałem.



$f(u,v)/c(u,v)$

Warunki nałożone na przepływ

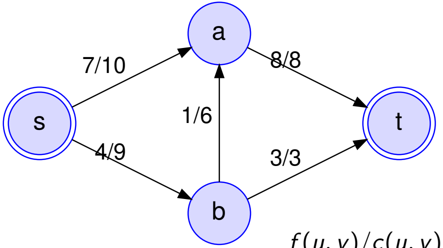
- **Warunek przepustowości**:

$$0 \leqslant f(u,v) \leqslant c(u,v). \tag{1}$$

Sieć przepływowa – definicje

Przepływ $f(u, v)$ (*flow*)

- **Przepustowość** $\equiv$ ile danym kanałem **można maksymalnie** przetransportować jednostek towaru (np. dziennie).
- **Przepływ** $\equiv$ ile **faktycznie** jednostek towaru przepływa danym kanałem.



$f(u, v)/c(u, v)$

Warunki nałożone na przepływ

- **Warunek przepustowości**:

$$0 \leqslant f(u, v) \leqslant c(u, v). \tag{1}$$

- **Warunek zachowania przepływu**:

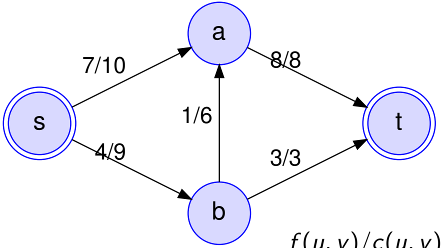
$$\sum_{v} f(v, u) = \sum_{v} f(u, v) \tag{2}$$

dla każdego wierzchołka $u \notin \{s, t\}$.

Sieć przepływowa – definicje

Przepływ $f(u, v)$ (*flow*)

- **Przepustowość** $\equiv$ ile danym kanałem **można maksymalnie** przetransportować jednostek towaru (np. dziennie).
- **Przepływ** $\equiv$ ile **faktycznie** jednostek towaru przepływa danym kanałem.



$f(u, v)/c(u, v)$

Warunki nałożone na przepływ

- **Warunek przepustowości**:

$$0 \leqslant f(u, v) \leqslant c(u, v). \qquad (1)$$

- **Warunek zachowania przepływu**:

$$\sum_v f(v, u) = \sum_v f(u, v) \qquad (2)$$

dla każdego wierzchołka $u \notin \{s, t\}$.

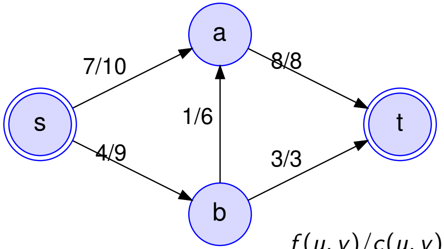
Wartość przepływu:

$$|f| = \sum_v f(s, v) = \sum_v f(v, t) \qquad (3)$$

Sieć przepływowa – definicje

Przepływ $f(u, v)$ (*flow*)

- **Przepustowość** $\equiv$ ile danym kanałem **można maksymalnie** przetransportować jednostek towaru (np. dziennie).
- **Przepływ** $\equiv$ ile **faktycznie** jednostek towaru przepływa danym kanałem.



$f(u, v)/c(u, v)$, $|f| = 11$

Warunki nałożone na przepływ

- **Warunek przepustowości**:

$$0 \leqslant f(u, v) \leqslant c(u, v). \tag{1}$$

- **Warunek zachowania przepływu**:

$$\sum_v f(v, u) = \sum_v f(u, v) \tag{2}$$

dla każdego wierzchołka $u \notin \{s, t\}$.

Wartość przepływu:

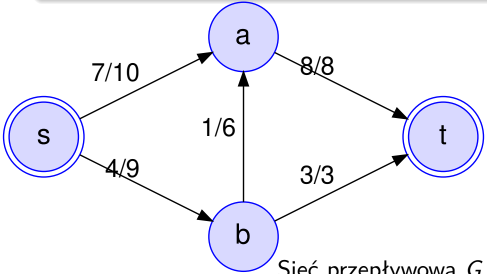
$$|f| = \sum_v f(s, v) = \sum_v f(v, t) \tag{3}$$

Sieć przepływowa – definicje

Sieć rezydualna G_f (łac. *residuum* – resztka, pozostałość)

- Złożona z krawędzi sieci G oraz krawędzi przeciwnych.
- Krawędzie mają określoną **przepustowość rezydualną** c_f:

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v), & \text{jeśli } (u,v) \text{ należy do } G, \\ f(v,u), & \text{jeśli } (v,u) \text{ należy do } G, \\ 0 & \text{w przeciwnym wypadku.} \end{cases} \tag{4}$$



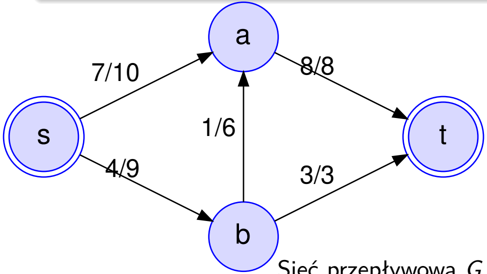
Sieć przepływowa G

Sieć przepływowa – definicje

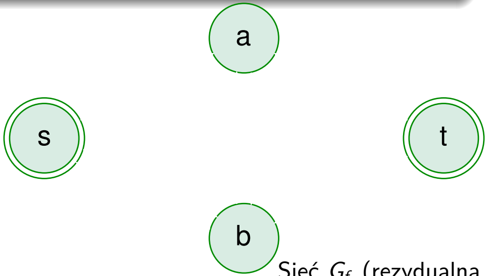
Sieć rezydualna G_f (łac. *residuum* – resztka, pozostałość)

- Złożona z krawędzi sieci G oraz krawędzi przeciwnych.
- Krawędzie mają określoną **przepustowość rezydualną** c_f:

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v), & \text{jeśli } (u,v) \text{ należy do } G, \\ f(v,u), & \text{jeśli } (v,u) \text{ należy do } G, \\ 0 & \text{w przeciwnym wypadku.} \end{cases} \tag{4}$$



Sieć przepływowa G



Sieć G_f (rezydualna dla G)

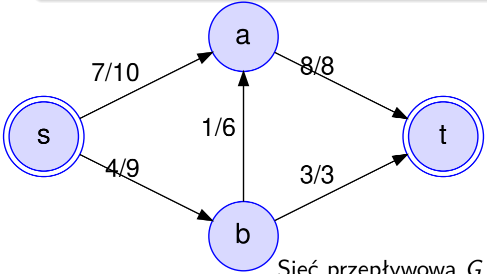
Sieć przepływowa – definicje

Sieć rezydualna G_f (łac. *residuum* – resztka, pozostałość)

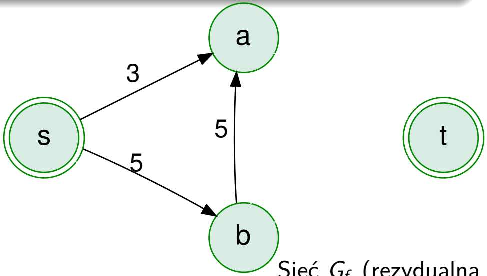
- Złożona z krawędzi sieci G oraz krawędzi przeciwnych.
- Krawędzie mają określoną **przepustowość rezydualną** c_f:

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v), & \text{jeśli } (u,v) \text{ należy do } G, \\ f(v,u), & \text{jeśli } (v,u) \text{ należy do } G, \\ 0 & \text{w przeciwnym wypadku.} \end{cases} \tag{4}$$

$c_f(u,v)$ tych krawędzi $\equiv$ ile jednostek jeszcze można tędy przetransportować?



Sieć przepływowa G



Sieć G_f (rezydualna dla G)

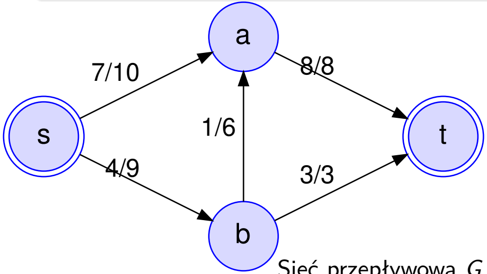
Sieć przepływowa – definicje

Sieć rezydualna G_f (łac. *residuum* – resztka, pozostałość)

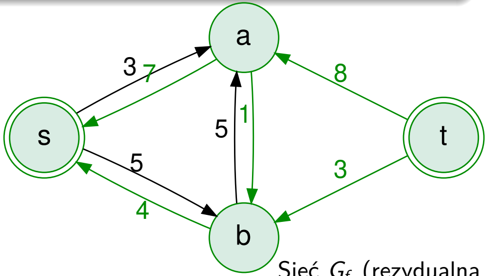
- Złożona z krawędzi sieci G oraz krawędzi przeciwnych.
- Krawędzie mają określoną **przepustowość rezydualną** c_f:

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v), & \text{jeśli } (u,v) \text{ należy do } G, \\ f(v,u), & \text{jeśli } (v,u) \text{ należy do } G, \\ 0 & \text{w przeciwnym wypadku.} \end{cases} \tag{4}$$

c_f krawędzi przeciwnych $\equiv$ ile jednostek można cofnąć?



Sieć przepływowa G



Sieć G_f (rezydualna dla G)

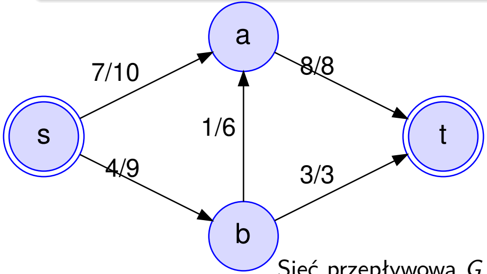
Sieć przepływowa – definicje

Sieć rezydualna G_f (łac. *residuum* – resztka, pozostałość)

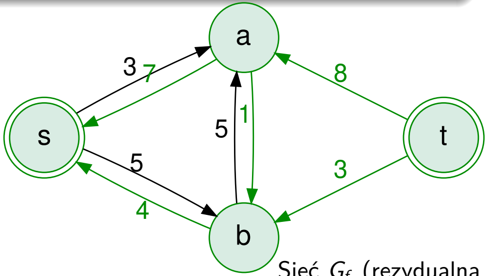
- Złożona z krawędzi sieci G oraz krawędzi przeciwnych.
- Krawędzie mają określoną **przepustowość rezydualną** c_f:

$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v), & \text{jeśli } (u,v) \text{ należy do } G, \\ f(v,u), & \text{jeśli } (v,u) \text{ należy do } G, \\ 0 & \text{w przeciwnym wypadku.} \end{cases} \tag{4}$$

$c_f(u,v) = 0 \Rightarrow$ w G_f nie istnieje taka krawędź.



Sieć przepływowa G



Sieć G_f (rezydualna dla G)

AGH

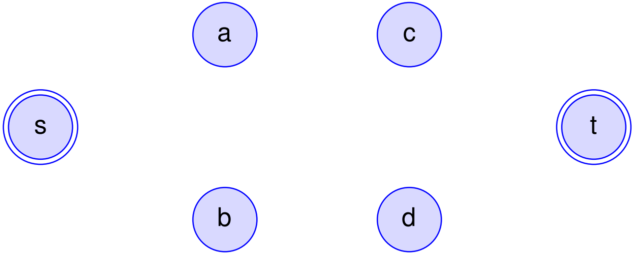
www.agh.edu.pl

Zestaw 5, zadanie 1

Napisać program do tworzenia losowej sieci przepływowej między pojedynczym źródłem i pojedynczym ujściem według następującej procedury. (...). Na tak otrzymanym digrafie przypisać każdemu łukowi liczbę naturalną z zakresu $[1, 10]$, mającą interpretację przepustowości. Zakodować i narysować otrzymaną sieć.

Losowanie sieci przepływowej

(...) Na potrzeby programu wprowadzić warstwy, które idą od źródła do ujścia. ***Źródło znajduje się w zerowej warstwie, a ujście w warstwie*** $N+1$. *Liczba pośrednich warstw wynosi* N *i jest parametrem programu* ($N \geqslant 2$, *a na potrzeby testowania* $N \leqslant 4$). *Pośrednie warstwy ponumerowane są od* 1 *do* N. ***W każdej pośredniej warstwie rozmieścić losowo od dwóch do*** N ***wierzchołków****. (...)*

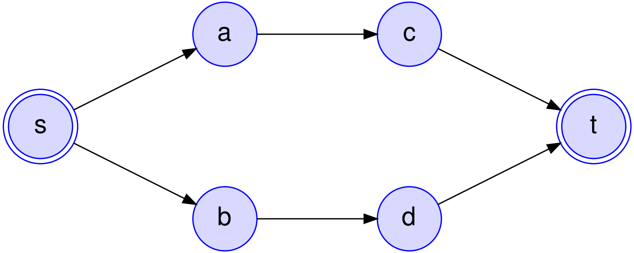


Przykład dla $N = 2$

- Losujemy wierzchołki.

Losowanie sieci przepływowej

*(...) Połączyć wierzchołki kolejnych warstw za pomocą **łuków skierowanych od warstwy** i **do warstwy** $i+1$ ($\forall_i = 0, \dots, N$), tak aby z każdego wierzchołka leżącego w warstwie i wychodził co najmniej jeden łuk i do każdego wierzchołka w warstwie $i+1$ wchodził co najmniej jeden łuk. (...)*

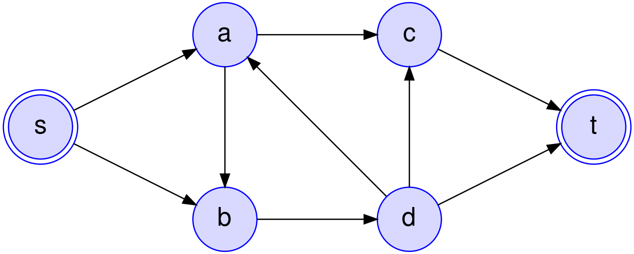


Przykład dla $N = 2$

- Losujemy wierzchołki.
- Losujemy krawędzie zgodnie z warstwami (spójność od s do t).

Losowanie sieci przepływowej

(...) Do otrzymanego w ten sposób digrafu należy następnie ***dodać*** $2N$ ***łuków w sposób losowy****. Łuki mają być losowane bez preferencji kierunku, tzn.* ***nie muszą być skierowane zgodnie z warstwami****. Należy jednak zwrócić uwagę, żeby* ***nie dodać łuku już istniejącego*** *i żeby* ***nie dodać łuku wchodzącego do źródła*** *albo* ***wychodzącego z ujścia****. (...)*



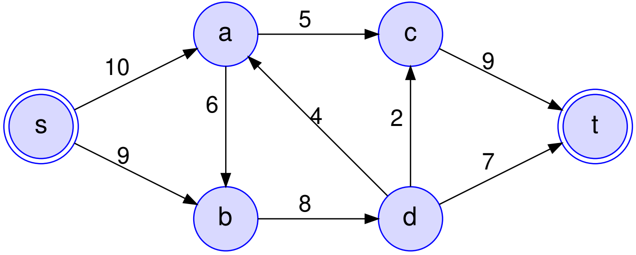
Tu dla uproszczenia wylosowano 3 *krawędzie zamiast* $2N = 4$.

Przykład dla $N = 2$

- Losujemy wierzchołki.
- Losujemy krawędzie zgodnie z warstwami (spójność od s do t).
- Losujemy $2N$ „prawie" dowolnych krawędzi.

Losowanie sieci przepływowej

(...) Na tak otrzymanym digrafie ***przypisać każdemu łukowi liczbę naturalną*** *z zakresu* $[1, 10]$*, mającą interpretację* ***przepustowości****. Zakodować i narysować otrzymaną sieć.*



Przykład dla $N = 2$

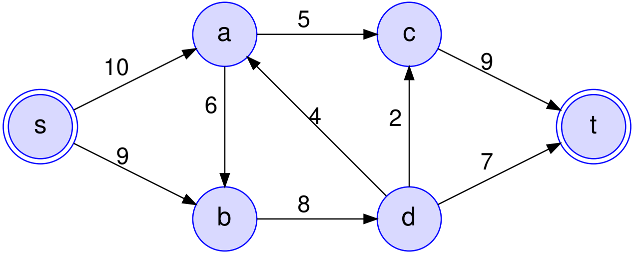
- Losujemy wierzchołki.
- Losujemy krawędzie zgodnie z warstwami (spójność od s do t).
- Losujemy $2N$ „prawie" dowolnych krawędzi.
- Losujemy $c(u, v)$.

www.agh.edu.pl

Zestaw 5, zadanie 2

Zastosować algorytm Forda-Fulkersona do znalezienia maksymalnego przepływu na sieci z zadania pierwszego. Ścieżki powiększające wybierać jako ścieżki o najmniejszej liczbie krawędzi. Do ich wyszukiwania użyć przeszukiwania wszerz.

Maksymalny przepływ



Ile wynosi **maksymalny przepływ** przez sieć o danej $c(u,v)$?

$\Downarrow$

Szukamy takiego rozłożenia $f(u,v)$, żeby $|f|$ było **maksymalne** ($|f_{\max}|$).

Algorytm Forda-Fulkersona

Algorithm: ford_fulkerson(G, s, t)

for każda krawędź (u, v) należąca do grafu G **do**
    $f(u, v) \leftarrow 0$
end for
while istnieje ścieżka rozszerzająca p z s do t w sieci G_f **do**
    $c_f(p) \leftarrow \min\{c_f(u, v)$ dla wszystkich krawędzi $(u, v) \in p\}$
    for każda krawędź $(u, v) \in p$ **do**
        if krawędź (u, v) należy do grafu G **then**
            $f(u, v) \leftarrow f(u, v) + c_f(p)$
        else
            $f(v, u) \leftarrow f(v, u) - c_f(p)$
        end if
    end for
end while

Algorytm Forda-Fulkersona

Algorithm: ford_fulkerson(G, s, t)

```
for każda krawędź (u, v) należąca do grafu G do
    f(u, v) ← 0
end for
while istnieje ścieżka rozszerzająca p z s do t w sieci G_f do
    c_f(p) ← min{c_f(u, v) dla wszystkich krawędzi (u, v) ∈ p}
    for każda krawędź (u, v) ∈ p do
        if krawędź (u, v) należy do grafu G then
            f(u, v) ← f(u, v) + c_f(p)
        else
            f(v, u) ← f(v, u) − c_f(p)
        end if
    end for
end while
```

- **Zerowanie** przepływów.

Algorytm Forda-Fulkersona

AGH

Algorithm: ford_fulkerson(G, s, t)

```
for każda krawędź (u, v) należąca do grafu G do
    f(u, v) ← 0
end for
while istnieje ścieżka rozszerzająca p z s do t w sieci G_f do
    c_f(p) ← min{c_f(u, v) dla wszystkich krawędzi (u, v) ∈ p}
    for każda krawędź (u, v) ∈ p do
        if krawędź (u, v) należy do grafu G then
            f(u, v) ← f(u, v) + c_f(p)
        else
            f(v, u) ← f(v, u) − c_f(p)
        end if
    end for
end while
```

- **Zerowanie** przepływów.
- Generowanie sieci rezydualnej + **znalezienie ścieżki rozszerzającej** p:

p – ścieżka w G_f, której każda krawędź ma $c_f > 0$.

Algorytm Forda-Fulkersona

Algorithm: `ford_fulkerson`(G, s, t)

```
for każda krawędź (u, v) należąca do grafu G do
    f(u, v) ← 0
end for
while istnieje ścieżka rozszerzająca p z s do t w sieci G_f do
    c_f(p) ← min{c_f(u, v) dla wszystkich krawędzi (u, v) ∈ p}
    for każda krawędź (u, v) ∈ p do
        if krawędź (u, v) należy do grafu G then
            f(u, v) ← f(u, v) + c_f(p)
        else
            f(v, u) ← f(v, u) − c_f(p)
        end if
    end for
end while
```

- **Zerowanie** przepływów.
- Generowanie sieci rezydualnej + **znalezienie ścieżki rozszerzającej** p:

  p – ścieżka w G_f, której każda krawędź ma $c_f > 0$.
- $c_f(p)$ – **przepustowość rezydualna ścieżki** $\equiv$ najmniejsza przepustowość rezydualna jej krawędzi.

Algorytm Forda-Fulkersona

AGH

Algorithm: `ford_fulkerson`(G, s, t)

```
for każda krawędź (u, v) należąca do grafu G do
    f(u, v) ← 0
end for
while istnieje ścieżka rozszerzająca p z s do t w sieci G_f do
    c_f(p) ← min{c_f(u, v) dla wszystkich krawędzi (u, v) ∈ p}
    for każda krawędź (u, v) ∈ p do
        if krawędź (u, v) należy do grafu G then
            f(u, v) ← f(u, v) + c_f(p)
        else
            f(v, u) ← f(v, u) − c_f(p)
        end if
    end for
end while
```

- **Zerowanie** przepływów.
- Generowanie sieci rezydualnej + **znalezienie ścieżki rozszerzającej** p:

  p – ścieżka w G_f, której każda krawędź ma $c_f > 0$.
- $c_f(p)$ – **przepustowość rezydualna ścieżki** $\equiv$ najmniejsza przepustowość rezydualna jej krawędzi.
- **Zwiększanie**/kasowanie przepływu wzdłuż ścieżki p.

Algorytm Forda-Fulkersona

Algorithm: `ford_fulkerson`(G, s, t)

for każda krawędź (u, v) należąca do grafu G **do**
    $f(u, v) \leftarrow 0$
end for
while istnieje ścieżka rozszerzająca p z s do t w sieci G_f **do**
    $c_f(p) \leftarrow \min\{c_f(u, v)$ dla wszystkich krawędzi $(u, v) \in p\}$
    for każda krawędź $(u, v) \in p$ **do**
        if krawędź (u, v) należy do grafu G **then**
            $f(u, v) \leftarrow f(u, v) + c_f(p)$
        else
            $f(v, u) \leftarrow f(v, u) - c_f(p)$
        end if
    end for
end while

- **Zerowanie** przepływów.
- Generowanie sieci rezydualnej + **znalezienie ścieżki rozszerzającej** p:

  p – ścieżka w G_f, której każda krawędź ma $c_f > 0$.
- $c_f(p)$ – **przepustowość rezydualna ścieżki** $\equiv$ najmniejsza przepustowość rezydualna jej krawędzi.
- Zwiększanie/**kasowanie** przepływu wzdłuż ścieżki p.

Algorytm Forda-Fulkersona

Algorithm: `ford_fulkerson`(G, s, t)

for każda krawędź (u, v) należąca do grafu G **do**
    $f(u, v) \leftarrow 0$
end for
while istnieje ścieżka rozszerzająca p z s do t w sieci G_f **do**
    $c_f(p) \leftarrow \min\{c_f(u, v)$ dla wszystkich krawędzi $(u, v) \in p\}$
    for każda krawędź $(u, v) \in p$ **do**
        if krawędź (u, v) należy do grafu G **then**
            $f(u, v) \leftarrow f(u, v) + c_f(p)$
        else
            $f(v, u) \leftarrow f(v, u) - c_f(p)$
        end if
    end for
end while

- **Zerowanie** przepływów.
- Generowanie sieci rezydualnej + **znalezienie ścieżki rozszerzającej** p:

  p – ścieżka w G_f, której każda krawędź ma $c_f > 0$.
- $c_f(p)$ – **przepustowość rezydualna ścieżki** $\equiv$ najmniejsza przepustowość rezydualna jej krawędzi.
- Zwiększanie/kasowanie przepływu wzdłuż ścieżki p.
- **Aktualizacja** G_f.

Algorytm Forda-Fulkersona

Jak znaleźć ścieżkę rozszerzającą p?

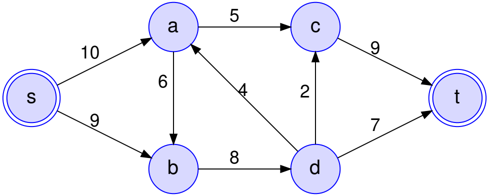
- W ogólności: dowolnie $\Rightarrow$ od tego zależy **złożoność** algorytmu F-F.
- Najgorszy przypadek przy $c(u, v) \in \mathbb{R}_+ : O(\infty)$:(
- Algorytm F-F + **przeszukiwanie wszerz (BFS)**: ścieżka p zawsze najkrótsza[a] $\Rightarrow$ **algorytm Edmondsa-Karpa**, $O(n \cdot k^2)$.
  n – liczba wierzchołków, k – liczba krawędzi

[a] Tym razem długość ścieżki definiujemy liczbą tworzących ją krawędzi.

Algorytm Edmondsa-Karpa – przykład działania

1. iteracja algorytmu

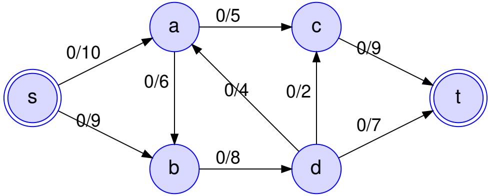
Sieć G, oznaczenia: $c(u, v)$



Algorytm Edmondsa-Karpa – przykład działania

1. iteracja algorytmu

Sieć G, oznaczenia: $f(u, v)/c(u, v)$

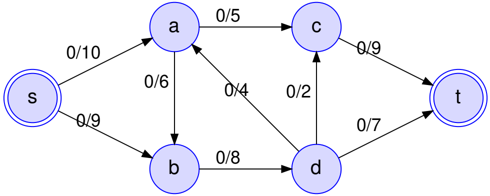


- Zerowanie przepływów.

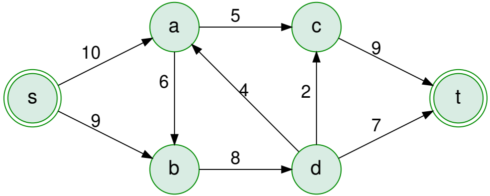
Algorytm Edmondsa-Karpa – przykład działania

1. iteracja algorytmu

Sieć G, oznaczenia: $f(u, v)/c(u, v)$



Sieć G_f, oznaczenia: $c_f(u, v)$



- Zerowanie przepływów.

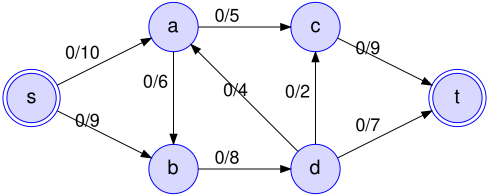
- Generowanie sieci rezydualnej.

Algorytm Edmondsa-Karpa – przykład działania

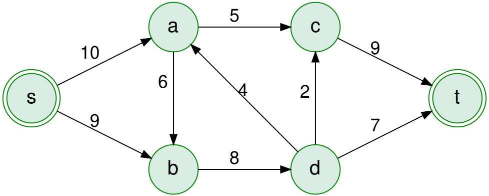
AGH

1. iteracja algorytmu

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



Sieć G_f, oznaczenia: $c_f(u,v)$



- Zerowanie przepływów.

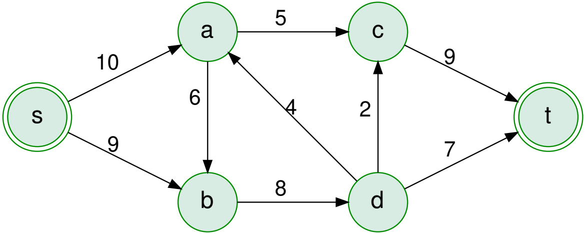
- Generowanie sieci rezydualnej.
- **Poszukiwanie ścieżki powiększającej o najkrótszej liczbie krawędzi $\Rightarrow$ BFS**

Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

Sieć G_f



Algorytm Edmondsa-Karpa – przykład działania

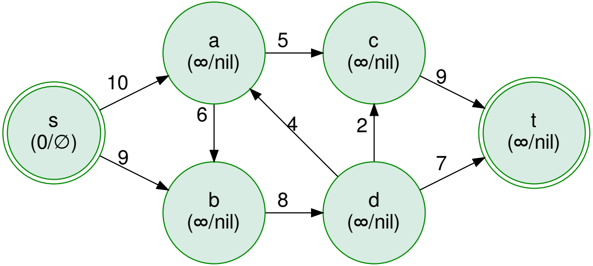
Przeszukiwanie wszerz (BFS)

- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

BFS(G_f, s)

Start: `init`(G_f, s).

Sieć G_f



Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

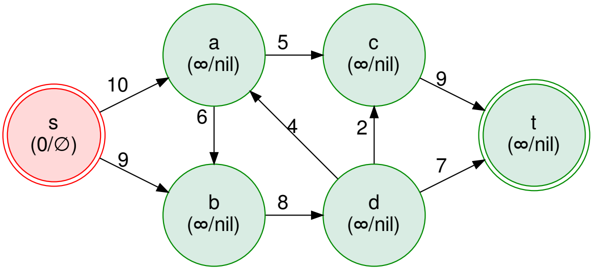
- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

BFS(G_f, s)

Start: `init`(G_f, s). Kolejka:

- [s]

Sieć G_f



Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

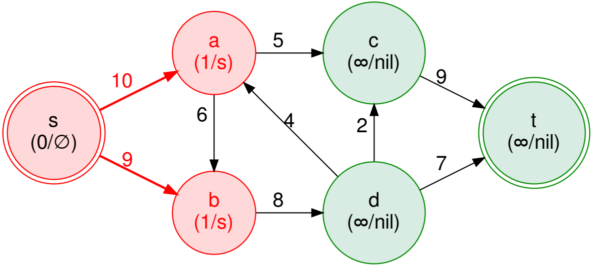
- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

BFS(G_f, s)

Start: `init`(G_f, s). Kolejka:

- [$\cancel{s}$]
- [a, b]

Sieć G_f



Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

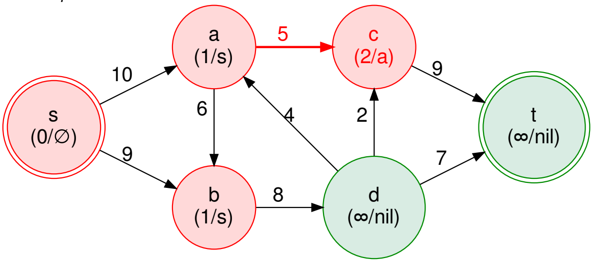
- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, …
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, …

BFS(G_f, s)

Start: $\texttt{init}(G_f, s)$. Kolejka:

- $[\cancel{s}]$
- $[\cancel{a}, b]$
- $[b, c]$

Sieć G_f



Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

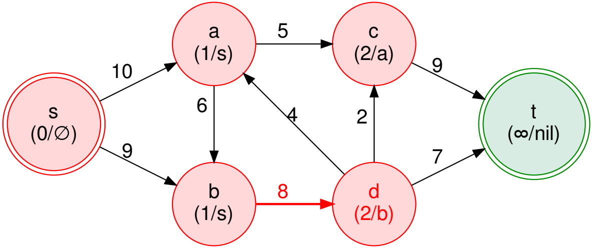
- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

BFS(G_f, s)

Start: `init`(G_f, s). Kolejka:

- $[\cancel{s}]$
- $[\cancel{a}, b]$
- $[\cancel{b}, c]$
- $[c, d]$

Sieć G_f



Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

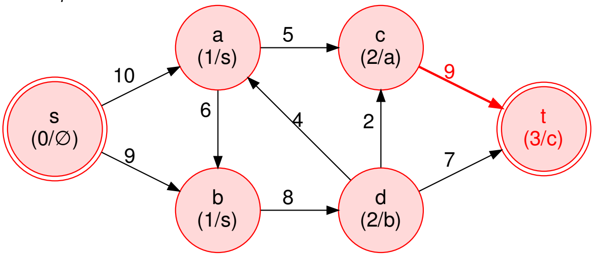
- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

BFS(G_f, s)

Start: `init`(G_f, s). Kolejka:

- $[\cancel{s}]$
- $[\cancel{a}, b]$
- $[\cancel{b}, c]$
- $[\cancel{c}, d]$
- $[d, t]$

Sieć G_f



Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

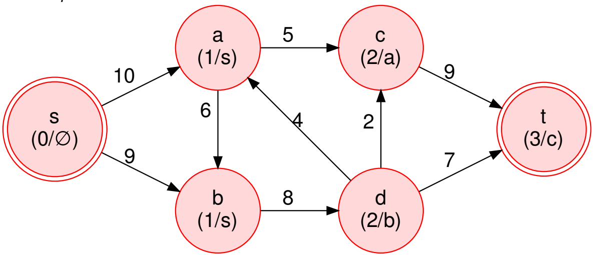
- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

BFS(G_f, s)

Start: $\mathtt{init}(G_f, s)$. Kolejka:

- $[\cancel{s}]$
- $[\cancel{a}, b]$
- $[\cancel{b}, c]$
- $[\cancel{c}, d]$
- $[\cancel{d}, t]$
- $[t]$

Sieć G_f



Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

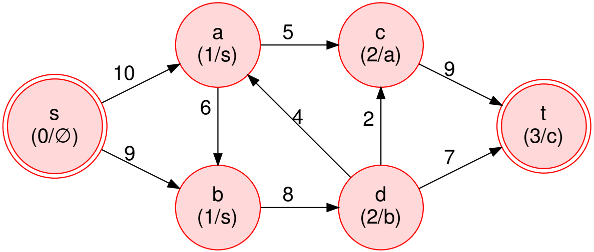
- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

BFS(G_f, s)

Start: `init`(G_f, s). Kolejka:

- $[\cancel{s}]$
- $[\cancel{a}, b]$
- $[\cancel{b}, c]$
- $[\cancel{c}, d]$
- $[\cancel{d}, t]$
- $[\cancel{t}]$

Sieć G_f



Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

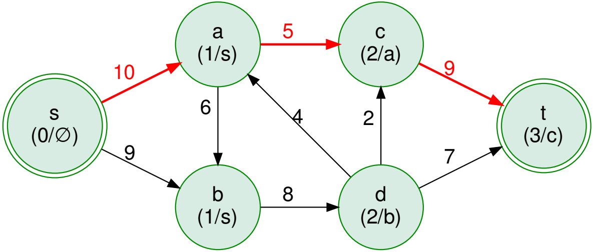
- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

BFS(G_f, s)

Start: `init`(G_f, s). Kolejka:

- [$\cancel{s}$]
- [$\cancel{a}, b$]
- [$\cancel{b}, c$]
- [$\cancel{c}, d$]
- [$\cancel{d}, t$]
- [$\cancel{t}$]

Sieć G_f, ścieżka p, $c_f(p) = ?$



Algorytm Edmondsa-Karpa – przykład działania

Przeszukiwanie wszerz (BFS)

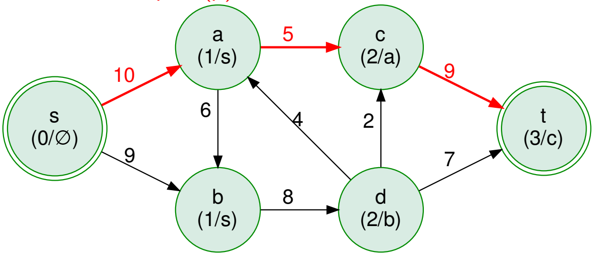
- Cel: **odwiedzić jak najwięcej wierzchołków, zanim zagłębimy się dalej.**
- Najpierw wszyscy sąsiedzi, później sąsiedzi sąsiadów, ...
- Implementacja z kolejką: dodajemy s do pustej kolejki. Ściągamy wierzchołek, odwiedzamy jego nieodwiedzonych sąsiadów, dodając ich do kolejki, ...

BFS(G_f, s)

Start: $\texttt{init}(G_f, s)$. Kolejka:

- $[\cancel{s}]$
- $[\cancel{a}, b]$
- $[\cancel{b}, c]$
- $[\cancel{c}, d]$
- $[\cancel{d}, t]$
- $[\cancel{t}]$

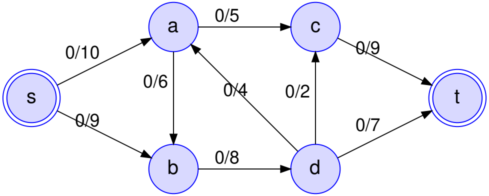
Sieć G_f, ścieżka p, $c_f(p) = 5$



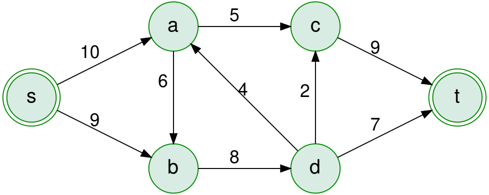
Algorytm Edmondsa-Karpa – przykład działania

1. iteracja algorytmu Edmondsa-Karpa – cd.

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



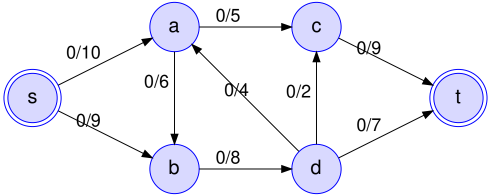
Sieć G_f, oznaczenia: $c_f(u,v)$



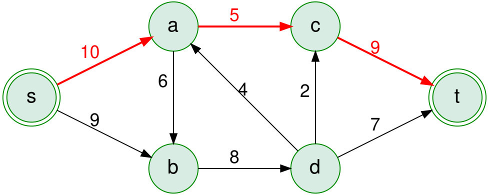
Algorytm Edmondsa-Karpa – przykład działania

1. iteracja algorytmu Edmondsa-Karpa – cd.

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



Sieć G_f, oznaczenia: $c_f(u,v)$

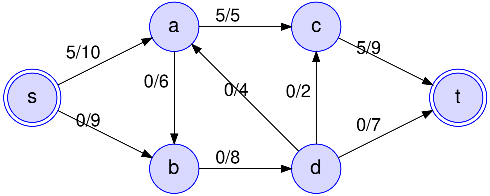


- BFS: znaleziona najkrótsza **ścieżka powiększająca** p: $s - a - c - t$, $c_f(p) = 5$.

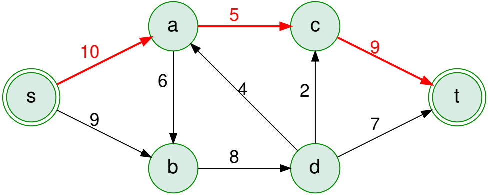
Algorytm Edmondsa-Karpa – przykład działania

1. iteracja algorytmu Edmondsa-Karpa – cd.

Sieć G, oznaczenia: $f(u, v)/c(u, v)$



Sieć G_f, oznaczenia: $c_f(u, v)$



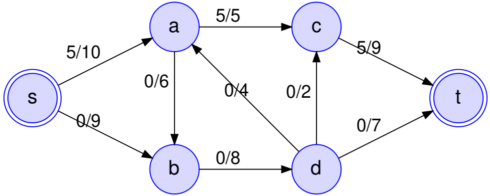
- Aktualizacja przepływów wzdłuż ścieżki p o $c_f(p) = 5$.

- BFS: znaleziona najkrótsza **ścieżka powiększająca** p: $s - a - c - t$, $c_f(p) = 5$.

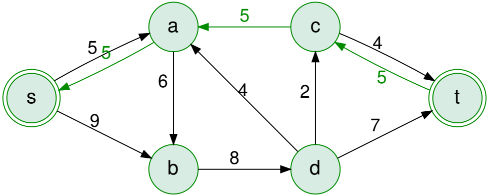
Algorytm Edmondsa-Karpa – przykład działania

2. iteracja algorytmu Edmondsa-Karpa

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



Sieć G_f, oznaczenia: $c_f(u,v)$

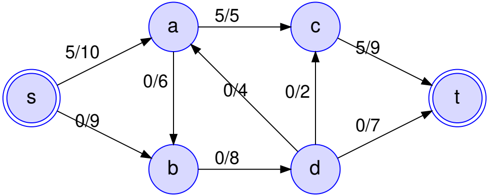


- Aktualizacja sieci rezydualnej.

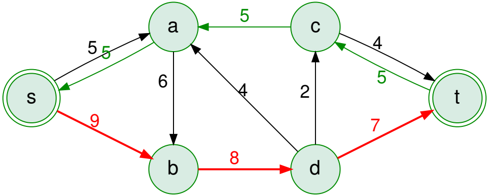
Algorytm Edmondsa-Karpa – przykład działania

2. iteracja algorytmu Edmondsa-Karpa

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



Sieć G_f, oznaczenia: $c_f(u,v)$

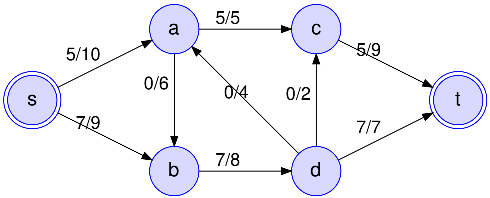


- Aktualizacja sieci rezydualnej.
- BFS: znaleziona najkrótsza **ścieżka powiększająca** p: $s - b - d - t$, $c_f(p) = 7$.

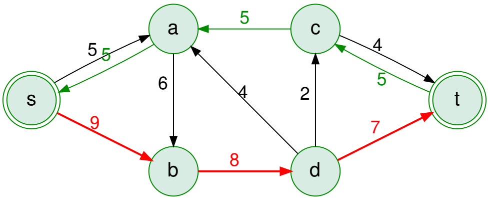
Algorytm Edmondsa-Karpa – przykład działania

2. iteracja algorytmu Edmondsa-Karpa

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



Sieć G_f, oznaczenia: $c_f(u,v)$



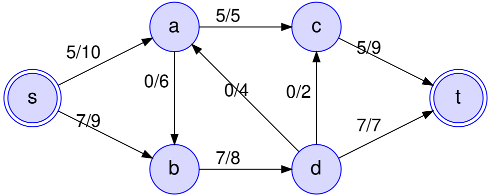
- Aktualizacja przepływów wzdłuż ścieżki p o $c_f(p) = 7$.

- Aktualizacja sieci rezydualnej.
- BFS: znaleziona najkrótsza **ścieżka powiększająca** p: $s - b - d - t$, $c_f(p) = 7$.

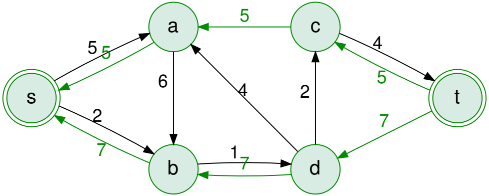
Algorytm Edmondsa-Karpa – przykład działania

3. iteracja algorytmu Edmondsa-Karpa

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



Sieć G_f, oznaczenia: $c_f(u,v)$

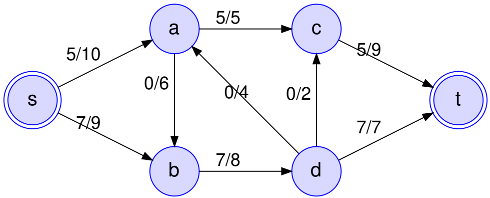


- Aktualizacja sieci rezydualnej.

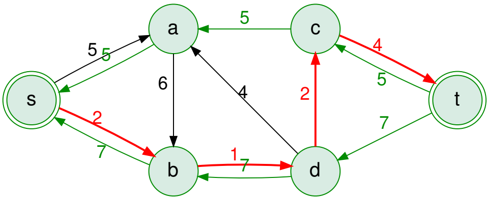
Algorytm Edmondsa-Karpa – przykład działania

3. iteracja algorytmu Edmondsa-Karpa

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



Sieć G_f, oznaczenia: $c_f(u,v)$

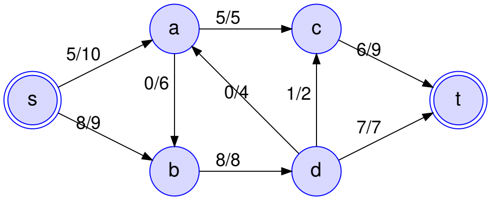


- Aktualizacja sieci rezydualnej.
- BFS: znaleziona najkrótsza **ścieżka powiększająca** p: $s - b - d - c - t$, $c_f(p) = 1$.

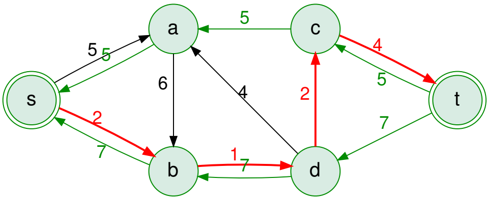
Algorytm Edmondsa-Karpa – przykład działania

3. iteracja algorytmu Edmondsa-Karpa

Sieć G, oznaczenia: $f(u, v)/c(u, v)$



Sieć G_f, oznaczenia: $c_f(u, v)$



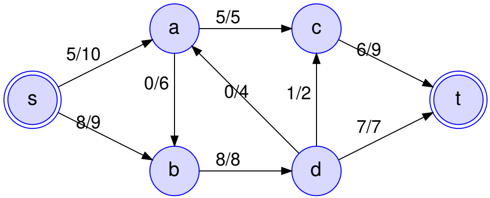
- Aktualizacja przepływów wzdłuż ścieżki p o $c_f(p) = 1$.

- Aktualizacja sieci rezydualnej.
- BFS: znaleziona najkrótsza **ścieżka powiększająca** p: $s - b - d - c - t$, $c_f(p) = 1$.

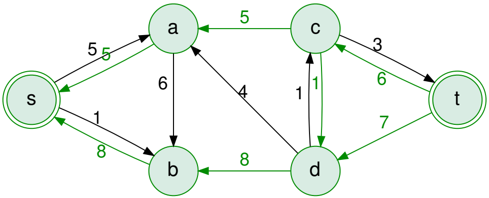
Algorytm Edmondsa-Karpa – przykład działania

4. (?) iteracja algorytmu Edmondsa-Karpa

Sieć G, oznaczenia: $f(u, v)/c(u, v)$



Sieć G_f, oznaczenia: $c_f(u, v)$

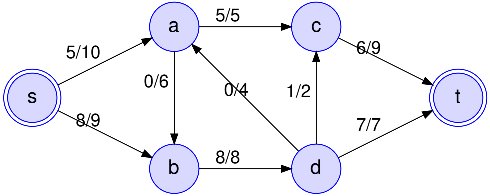


- Aktualizacja sieci rezydualnej.

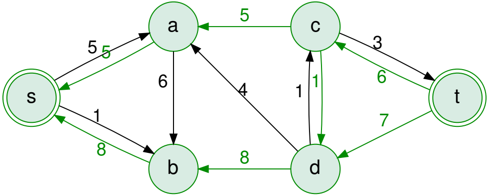
Algorytm Edmondsa-Karpa – przykład działania

~~4. (?) iteracja algorytmu Edmondsa-Karpa~~

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



Sieć G_f, oznaczenia: $c_f(u,v)$

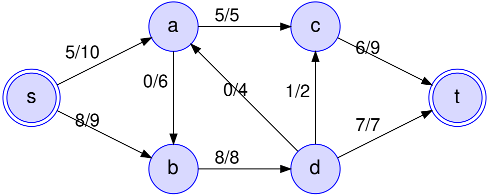


- Aktualizacja sieci rezydualnej.
- BFS: **brak ścieżki powiększającej w** G_f.

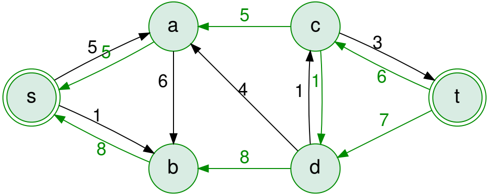
Algorytm Edmondsa-Karpa – przykład działania

~~4. (?) iteracja algorytmu Edmondsa-Karpa~~

Sieć G, oznaczenia: $f(u,v)/c(u,v)$



Sieć G_f, oznaczenia: $c_f(u,v)$



- **Koniec** działania algorytmu.
- **Wynik**: $|f_{\max}| = 13$.

- Aktualizacja sieci rezydualnej.
- BFS: **brak ścieżki powiększającej w** G_f.

Algorytm Forda-Fulkersona – cofanie przepływu

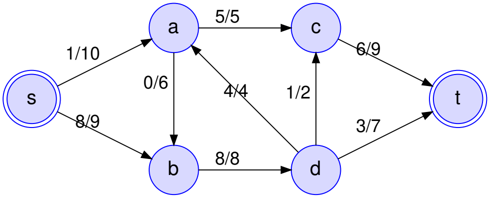
Uwaga: Czy kasowanie przepływu jest potrzebne?

. . .

for każda krawędź $(u, v) \in p$ do
    if krawędź (u, v) należy do grafu G then
        $f(u, v) \leftarrow f(u, v) + c_f(p)$
    else
        $f(v, u) \leftarrow f(v, u) - c_f(p)$
    end if
end for

. . .

Sieć G



- **Aktualna wartość** $|f| = 9$.

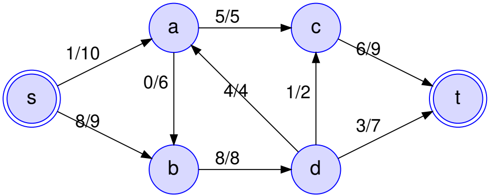
Algorytm Forda-Fulkersona – cofanie przepływu

Uwaga: Czy kasowanie przepływu jest potrzebne?

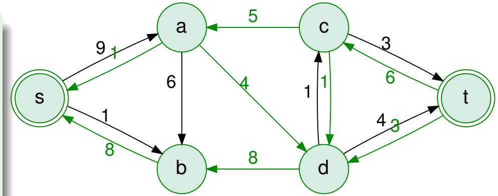
```
...
    for każda krawędź (u, v) ∈ p do
        if krawędź (u, v) należy do grafu G then
            f(u, v) ← f(u, v) + c_f(p)
        else
            f(v, u) ← f(v, u) − c_f(p)
        end if
    end for
...
```

- Aktualna wartość $|f| = 9$.

Sieć G



Sieć G_f

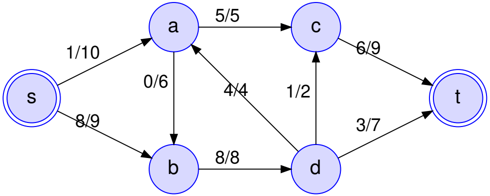


Algorytm Forda-Fulkersona – cofanie przepływu

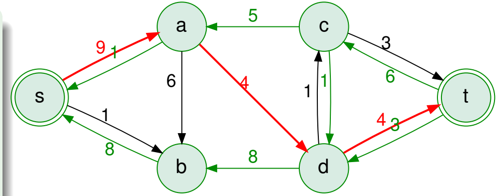
Uwaga: Czy kasowanie przepływu jest potrzebne?

```
...
    for każda krawędź (u, v) ∈ p do
        if krawędź (u, v) należy do grafu G then
            f(u, v) ← f(u, v) + c_f(p)
        else
            f(v, u) ← f(v, u) − c_f(p)
        end if
    end for
...
```

Sieć G



Sieć G_f, $c_f(p) = 4$



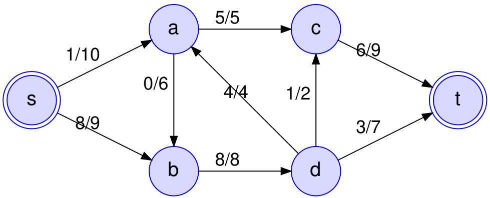
- Aktualna wartość $|f| = 9$.
- **Ścieżka powiększająca w** G_f: $s - a - d - t$, $c_f(p) = 4$.

Algorytm Forda-Fulkersona – cofanie przepływu

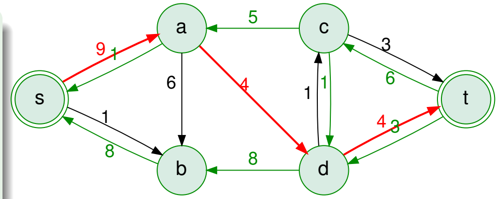
Uwaga: Czy kasowanie przepływu jest potrzebne?

```
...
for każda krawędź (u, v) ∈ p do
    if krawędź (u, v) należy do grafu G then
        f(u, v) ← f(u, v) + c_f(p)
    else
        f(v, u) ← f(v, u) − c_f(p)
    end if
end for
...
```

Sieć G



Sieć G_f, $c_f(p) = 4$



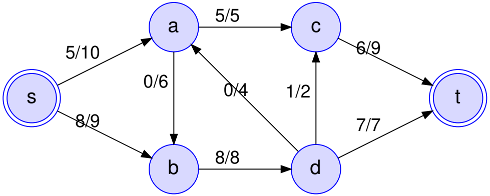
- Aktualna wartość $|f| = 9$.
- Ścieżka powiększająca w G_f: $s - a - d - t$, $c_f(p) = 4$.
- **Krawędź (a, d) nie istnieje w G!**

Algorytm Forda-Fulkersona – cofanie przepływu

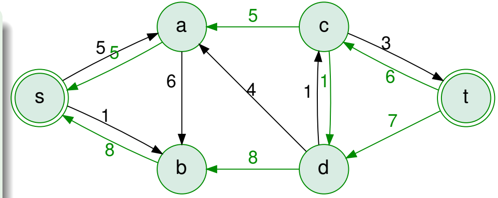
Uwaga: Czy kasowanie przepływu jest potrzebne?

```
...
    for każda krawędź (u, v) ∈ p do
        if krawędź (u, v) należy do grafu G then
            f(u, v) ← f(u, v) + c_f(p)
        else
            f(v, u) ← f(v, u) − c_f(p)
        end if
    end for
...
```

Sieć G



Sieć G_f, $c_f(p) = 4$



- Aktualna wartość $|f| = 9$.
- Ścieżka powiększająca w G_f: $s - a - d - t$, $c_f(p) = 4$.
- **Krawędź** (a, d) **nie istnieje w** G**!** $\Rightarrow$ **cofanie przepływu** $f(d, a)$.

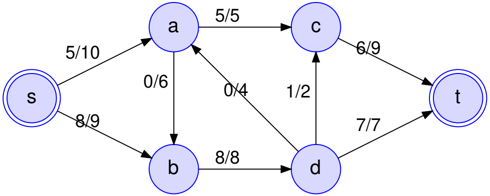
Algorytm Forda-Fulkersona – cofanie przepływu

Uwaga: Czy kasowanie przepływu jest potrzebne?

```
...
  for każda krawędź (u, v) ∈ p do
    if krawędź (u, v) należy do grafu G then
      f(u, v) ← f(u, v) + c_f(p)
    else
      f(v, u) ← f(v, u) − c_f(p)
    end if
  end for
...
```

- Aktualna wartość $|f| = 9$.
- Ścieżka powiększająca w G_f: $s - a - d - t$, $c_f(p) = 4$.
- Krawędź (a, d) nie istnieje w G! $\Rightarrow$ cofanie przepływu $f(d, a)$.
- **Aktualna wartość** $|f| = 13 = |f_{\max}|$.

Sieć G



Sieć G_f, $c_f(p) = 4$