Algorithms & Complexity
Lecture 5: Maximum Flow

Antoine Vigneron

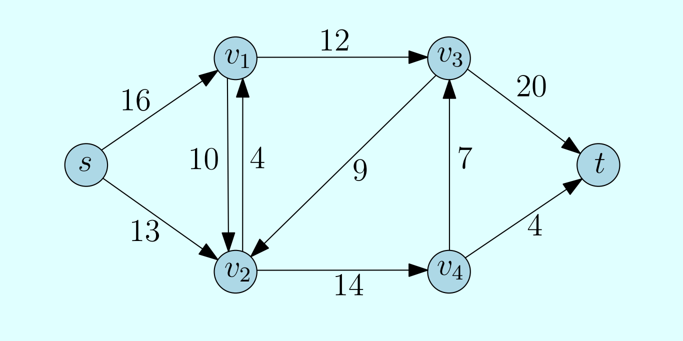
Ulsan National Institute of Science and Technology

March 18, 2018

Reference

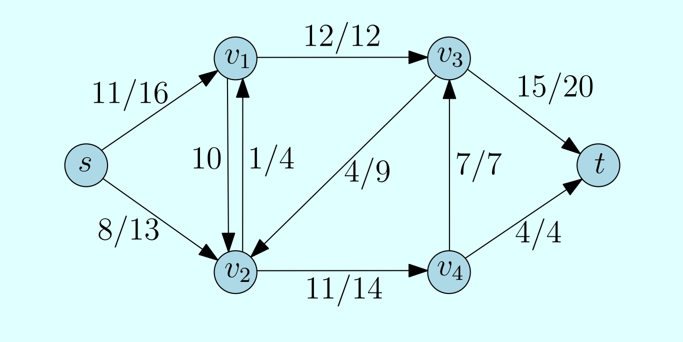
- Assignment 1 is due now.
- Chapter 26 in Introduction to Algorithms by Cormen, Leiserson, Rivest and Stein.
  - These slides are based on the *2nd edition* (2001), also available at the library.
  - The 3rd edition uses a different convention: If edge $(u, v) \in E$ then $(v, u) \notin E$, and then flow conservation is written differently and skew symmetry is irrelevant.

Flow Networks



- A *flow network* $G = (V, E)$ is a directed graph.
- Each edge (u, v) is weighted by a non-negative *capacity* $c(u, v) \geqslant 0$.
  - If $(u, v) \notin E$, then $c(u, v) = 0$.
- Two special vertices: the *source* s and the *sink* t.
- For each $v \in V$, there is a path $s \rightsquigarrow v \rightsquigarrow t$.

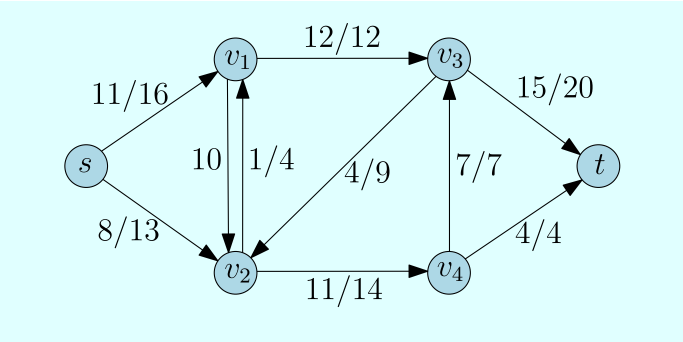
Flows



A *flow* in G is a function $f : V \times V \to \mathbb{R}$ such that:

- $\forall u, v \in V,\ f(u,v) \leqslant c(u,v).$ (*Capacity constraint*)
- $\forall u, v \in V,\ f(u,v) = -f(v,u).$ (*Skew symmetry*)
- $\forall u \in V \setminus \{s,t\},\ \sum_{v \in V} f(u,v) = 0.$ (*Flow conservation*)

Terminology

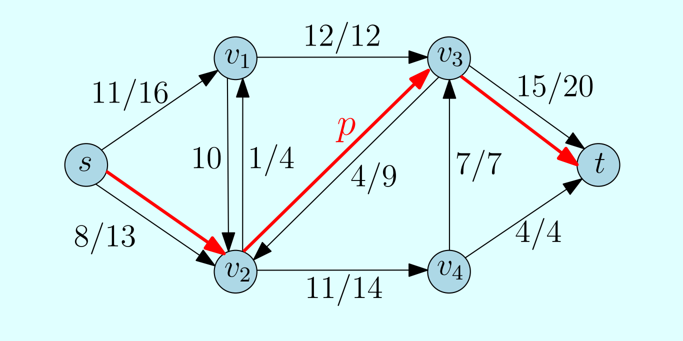


- $f(u, v)$ is called the *flow* from u to v.
- The *value* of a flow is $|f| = \sum_{v \in V} f(s, v)$.
- The *maximum-flow problem* is to find a flow of maximum value in a flow network.

Augmenting Path

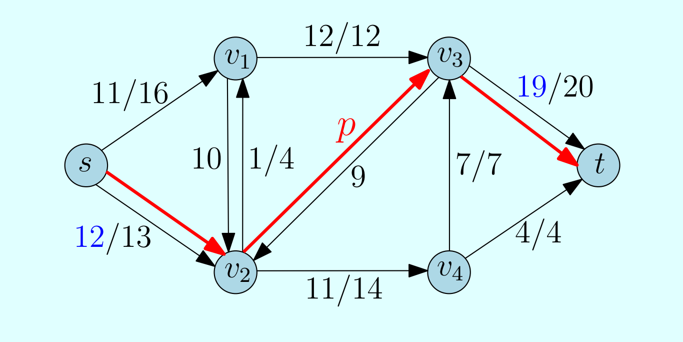
Definition

A *simple path* in a graph is a path with no repeated vertices.



An *augmenting path* is a simple path $p : s \rightsquigarrow t$ along which we can send more flow.

Augmenting Path



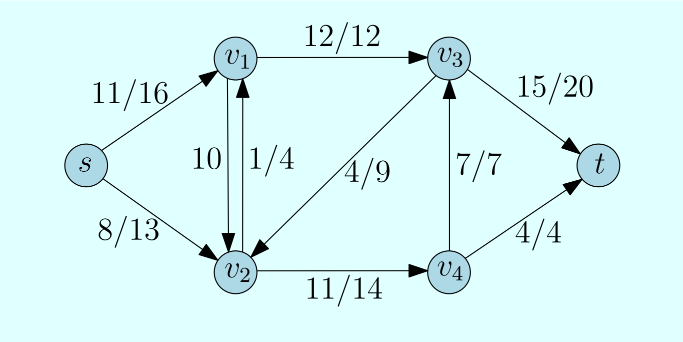
Result after sending 4 units of flow
along the augmenting path *p*.

The Ford-Fulkerson Method

Ford-Fulkerson method for maximum flow

```
Initialize flow f to 0.
while there exists an augmenting path p do
    augment flow f along p.
return f
```

Residual Network

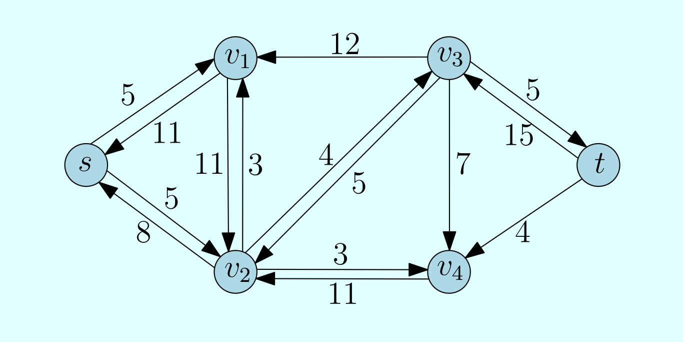


A flow network G and a flow f.

The *residual capacity* of (u, v) is $c_f(u, v) = c(u, v) - f(u, v)$.

- Here, $c_f(s, v_2) = 5$ and $c_f(v_2, v_3) = 0 - (-4) = 4$.
- Intuitively, the residual capacity $c_f(u, v)$ is the additional amount of flow we can push from u to v.

Residual Network

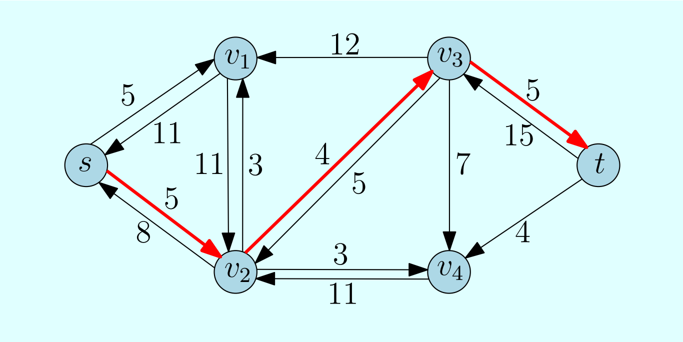


The *residual network* $G_f(V, E_f)$, with edge set

$$E_f = \{(u, v) \in V \times V \mid c_f(u, v) > 0\}.$$

Residual Network

The *residual capacity of a path* p is $c_f(p) = \min\{c_f(u,v) \mid (u,v) \text{ is on } p\}$.

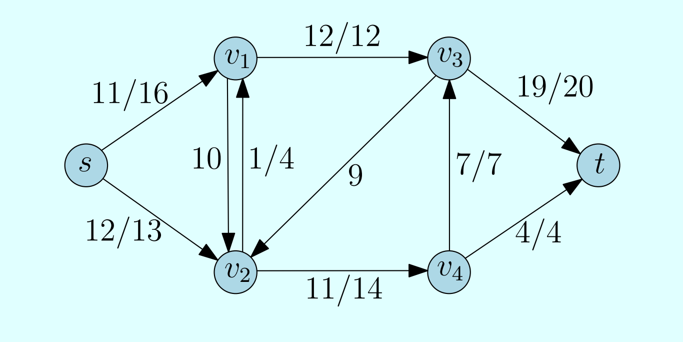


The augmenting path p, with residual capacity 4.

Property

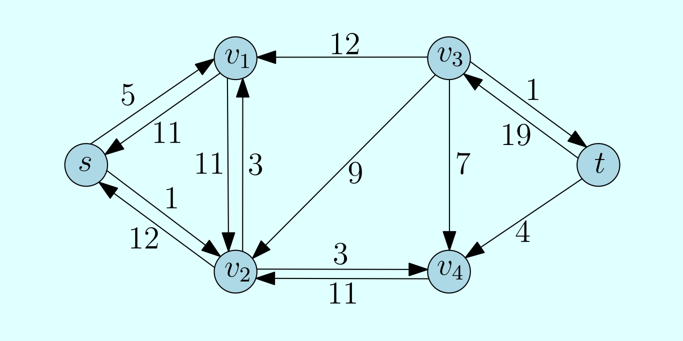
An augmenting path in G is a simple path $p : s \rightsquigarrow t$ such that $c_f(p) > 0$, or equivalently, it is a simple path $p : s \rightsquigarrow t$ in G_f.

Residual Network



The flow after augmenting p by its residual capacity 4.

Residual Network



The residual network after augmenting p by its residual capacity 4.

There is no augmenting path now, the Ford-Fulkerson method returns this flow.

Flow Sums

Definition

Let f_1 and f_2 be flows in G. Let $f_1 + f_2 : V \times V \to \mathbb{R}$ be the function such that $(f_1 + f_2)(u, v) = f_1(u, v) + f_2(u, v)$ for all $u, v \in V$. If $f_1 + f_2$ is a flow in G, then we say that $f_1 + f_2$ is the *flow sum* of f_1 and f_2.

- Which flow property can fail for $f_1 + f_2$?

Lemma

Let f be a flow in the flow network G. Let f' be a flow in the residual network G_f. Then $f + f'$ is a flow in G with value $|f + f'| = |f| + |f'|$.

Proof done in class.

Augmenting Paths

Let $G = (V, E)$ be a flow network. Let f be a flow in G, and let p be an augmenting path in G_f. Define $f_p : V \times V \to \mathbb{R}$ by

$$f_p(u, v) = \begin{cases} c_f(p) & \text{if } (u, v) \text{ is on } p, \\ -c_f(p) & \text{if } (v, u) \text{ is on } p, \\ 0 & \text{otherwise.} \end{cases}$$

Lemma

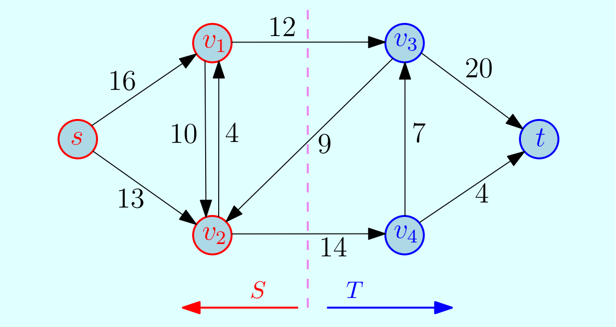
The function f_p is a flow in G_f with value $|f_p| = c_f(p) > 0$.

Proof done in class.

Corollary

Let $f' : V \times V \to \mathbb{R}$ be defined by $f' = f + f_p$. Then f' is a flow in G with value $|f'| = |f| + |f_p| > |f|$.

Cuts of Flow Networks

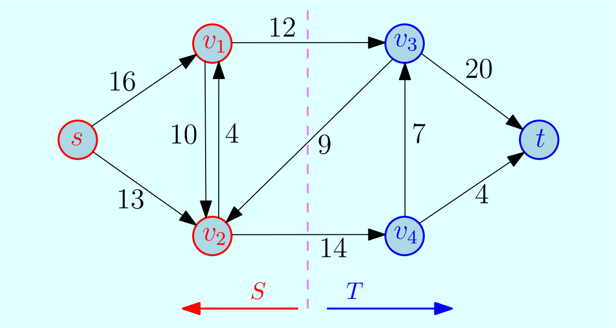


Definition

A *cut* (S, T) of a flow network $G = (V, E)$ is a partition of V into S and $T = V \setminus S$ such that $s \in S$ and $t \in T$.

Here $(S, T) = (\{s, v_1, v_2\}, \{v_3, v_4, t\})$.

Cuts of Flow Networks

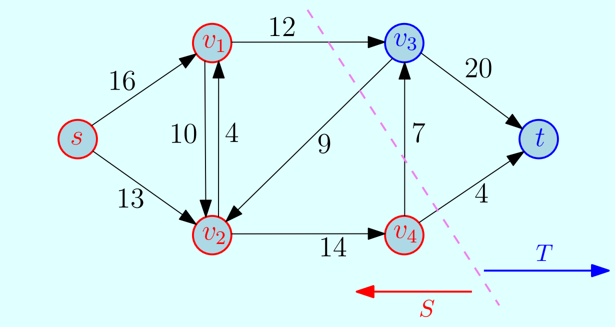


Definition

The *capacity* of a cut (S, T) is $c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v)$.

Here $c(S, T) = 12 + 14 = 26$.

Cuts of Flow Networks

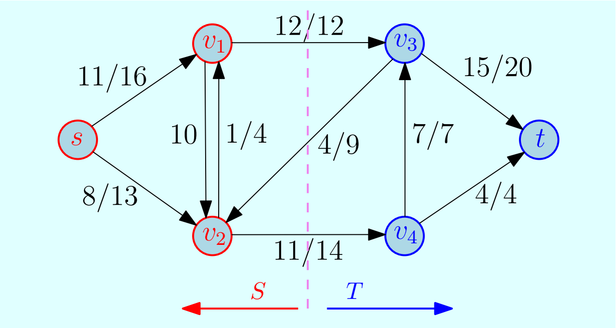


Definition

A *minimum cut* is a cut (S, T) with minimum capacity.

Here the minimum cut (S, T) has capacity $c(S, T) = 12 + 7 + 4 = 23$.

Cuts of Flow Networks



Definition

The *net flow* across a cut (S, T) is $f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v)$.

Here $f(S, T) = 12 + 11 - 4 = 19$.

Cuts of Flow Networks

Lemma

For any cut (S, T), the net flow $f(S, T)$ across (S, T) is equal to the value $|f|$ of the flow.

Proof (sketch).

For any $X, Y \subset V$, we denote $f(X, Y) = \sum_{u \in X} \sum_{v \in Y} f(u, v)$.

$$\begin{aligned} f(S, T) &= f(S, V) - f(S, S) \\ &= f(S, V) \\ &= f(\{s\}, V) + f(S \setminus \{s\}, V) \\ &= f(\{s\}, V) \\ &= |f| \end{aligned}$$

Cuts of Flow Networks

Corollary (1)

The flow $\sum_{u \in V} f(u,t)$ *into the sink is equal to* $|f|$.

Corollary (2)

The value $|f|$ *of any flow* f *is at most the capacity* $c(S,T)$ *of any cut* (S,T).

Cuts of Flow Networks

Theorem (Max-flow min-cut theorem)

In a flow network, the maximum value of a flow is equal to the minimum capacity of a cut.

Proof: follows from the Lemma below.

Lemma

If f is a flow in a network G, then the following three conditions are equivalent:

1. *f is a maximum flow in G.*
2. *The residual network G_f admits no augmenting path.*
3. *The value $|f|$ of f is equal to the capacity $c(S, T)$ of a cut (S, T).*

Proof of the Lemma: (1) $\Rightarrow$ (2) $\Rightarrow$ (3) $\Rightarrow$ (1).

The Basic Ford-Fulkerson Algorithm

Ford-Fulkerson

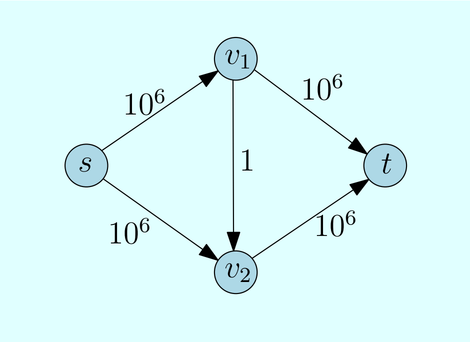
1: **for** each edge $(u, v) \in E$ **do**
2: $\quad f[u, v] \leftarrow 0$
3: $\quad f[v, u] \leftarrow 0$
4: **while** $\exists$ simple path $p : s \rightsquigarrow t$ in G_f **do**
5: $\quad c_f(p) \leftarrow \min\{c_f(u, v) \mid (u, v) \text{ is in } p\}$
6: $\quad$ **for** each edge (u, v) in p **do**
7: $\quad\quad f[u, v] \leftarrow f[u, v] + c_f(p)$
8: $\quad\quad f[v, u] \leftarrow -f[u, v]$
9: **return** f

At Line 4, the path p is found by depth-first search or breadth-first search.

Analysis

- We assume integral capacities: $c(u, v) \in \mathbb{N}$ for each u, v.
- Denote by $|f^*|$ the value of an optimal flow.
- Lines 1–3: $O(|E|)$.
- The While loop is iterated at most $|f^*|$ times.
- At each iteration:
  - Line 4: graph search (DFS or BFS) can be done in $O(|E| + |V|)$ time.
    - This is $O(|E|)$ in our case because the graph is connected, hence $|E| \geqslant |V| - 1$.
  - Lines 5–8: $O(|E|)$.
- Overall running time: $O(|E| \times |f^*|)$.

Bad Case



- $|f^*| = 2.10^6$.
- In this example, in the worst case, the while loop is iterated $|f^*|$ time.

The Edmonds-Karp Algorithm

The *Edmonds-Karp* algorithm is the basic Ford-Fulkerson method with breadth-first search.

- In particular, we take an augmenting path with as few edges as possible.

Theorem

The Edmonds-Karp algorithm computes a maximum flow in $O(|V| \cdot |E|^2)$ time.

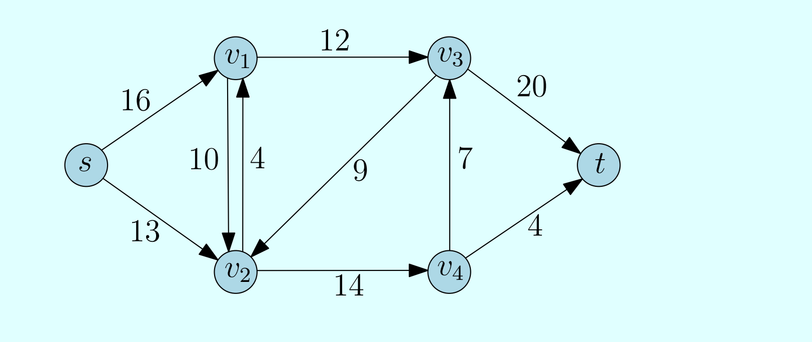
We denote by $\delta_f(u, v)$ the shortest path distance from u to v in G_f, where each edge has unit distance.

Lemma

For each vertex v, the shortest path distance $\delta_f(s, v)$ never decreases during the course of the Edmonds-Karp algorithm.

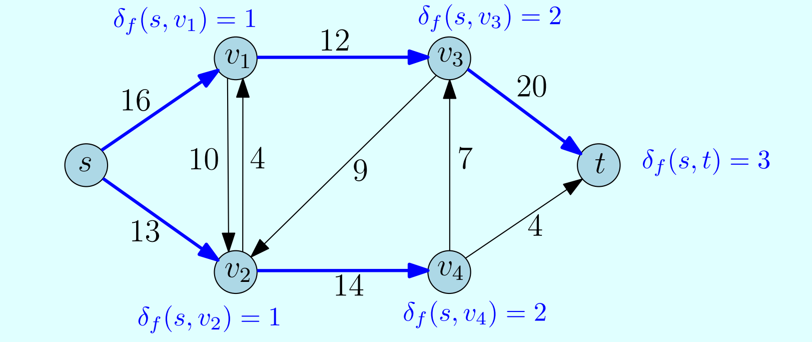
See next slides for the proofs of the lemma and the theorem.

The Edmonds-Karp Algorithm: Example



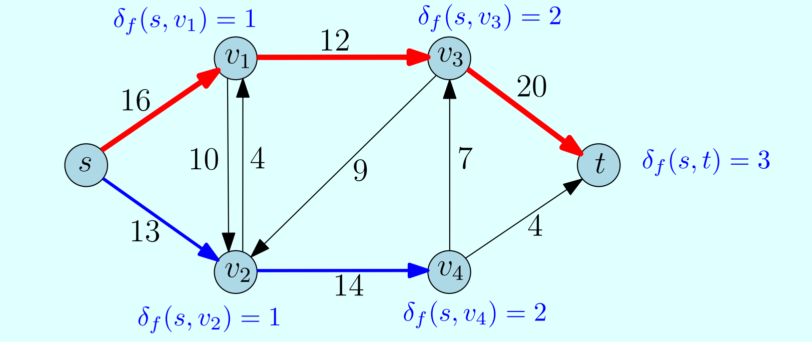
The flow network G.

The Edmonds-Karp Algorithm: Example



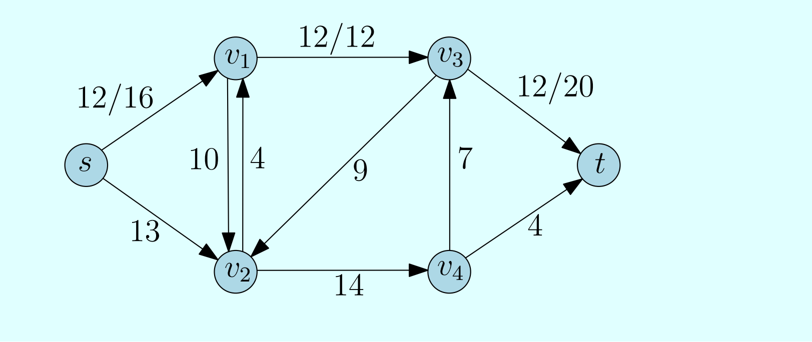
Breadth-first search in G_f for $f = 0$.

The Edmonds-Karp Algorithm: Example



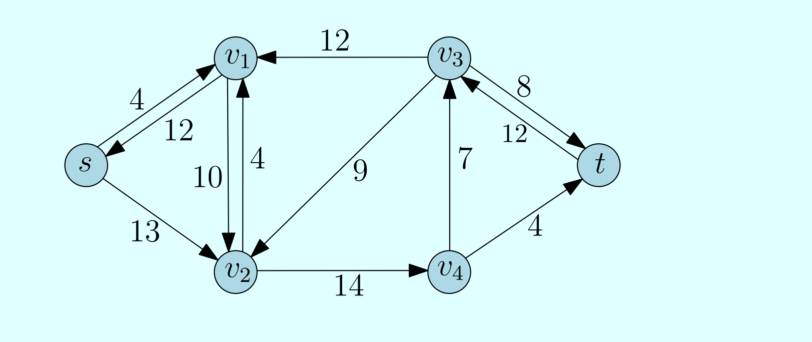
The augmenting path p, with residual capacity $c_f(p) = 12$.

The Edmonds-Karp Algorithm: Example



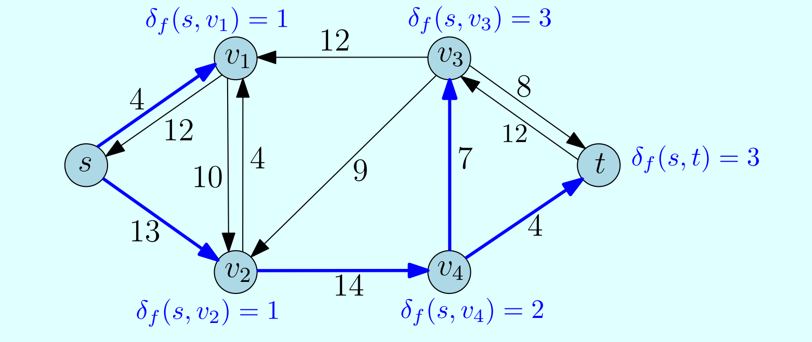
The flow f after pushing 12 units through p.

The Edmonds-Karp Algorithm: Example



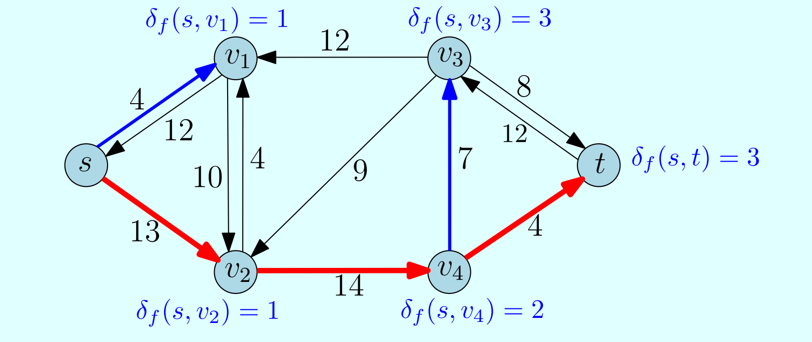
The residual network G_f.

The Edmonds-Karp Algorithm: Example



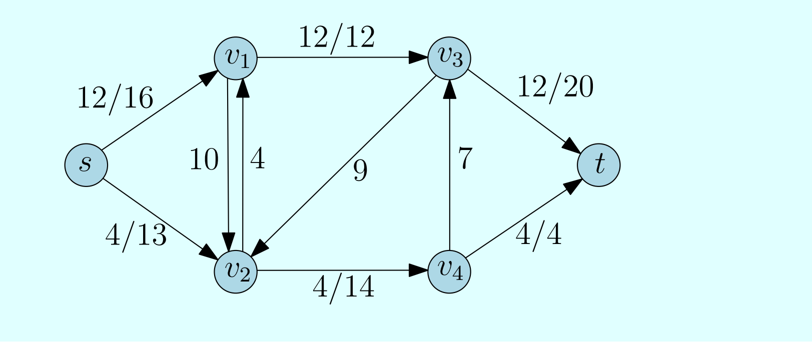
Breadth-first search in G_f.

The Edmonds-Karp Algorithm: Example



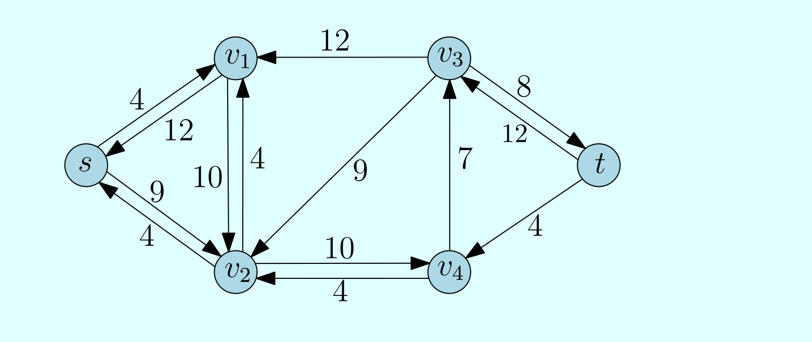
The augmenting path p, with residual capacity $c_f(p) = 4$.

The Edmonds-Karp Algorithm: Example



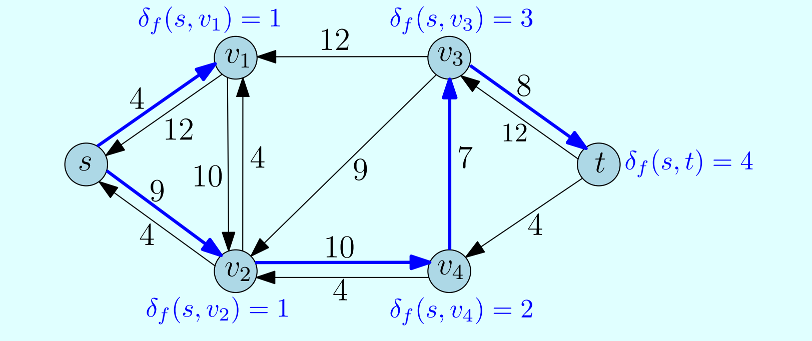
The flow f after pushing 4 units through p.

The Edmonds-Karp Algorithm: Example



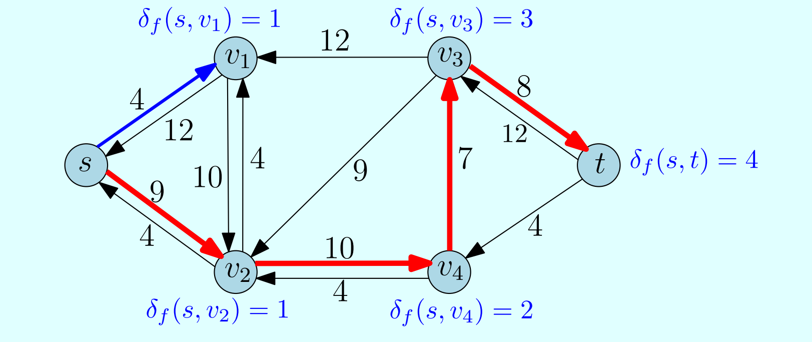
The residual network G_f.

The Edmonds-Karp Algorithm: Example



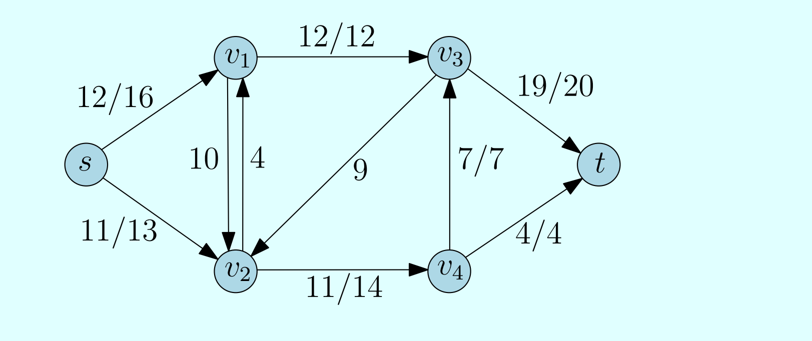
Breadth-first search in G_f.

The Edmonds-Karp Algorithm: Example



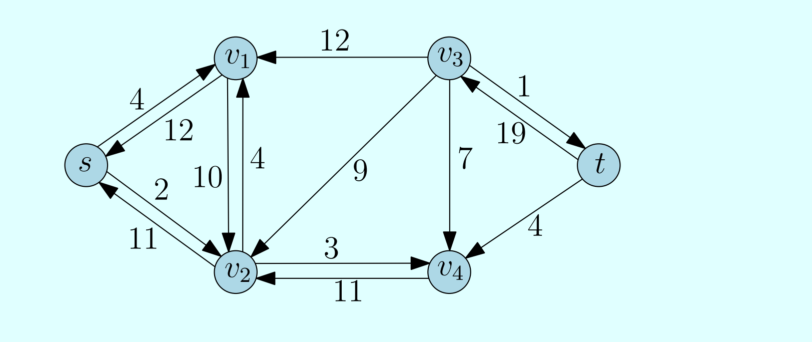
The augmenting path p, with residual capacity $c_f(p) = 7$.

The Edmonds-Karp Algorithm: Example



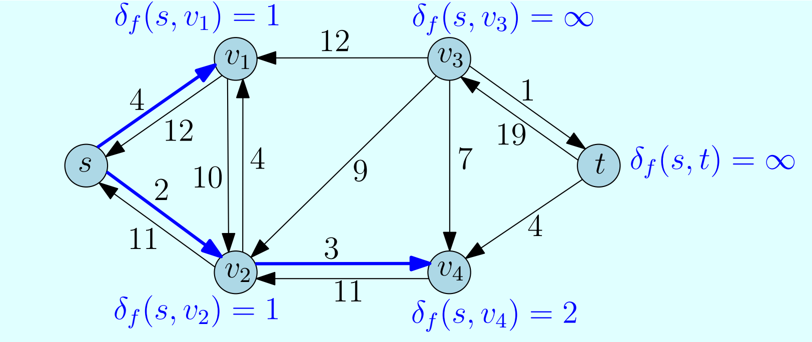
The flow f after pushing 7 units through p.

The Edmonds-Karp Algorithm: Example



The residual network G_f.

The Edmonds-Karp Algorithm: Example



Breadth-first search in G_f.

The sink t is unreachable, so the algorithm terminates.

The Edmonds-Karp Algorithm: Proof of the Lemma

We want to prove that for each v, the distance $\delta_f(s, v)$ never decreases during the course of the Edmonds-Karp algorithm.

- When the flow is augmented along p, some edges are created or deleted in G_f, which may affect δ_f.
- We will first apply the insertions, and then the deletions, and see how δ_f is affected.
- So we first consider the new edges.
  - An edge (v, u) may be created if $(u, v) \in p$.
  - But then, before the edge is introduced, $\delta_f(v) = \delta_f(u) + 1$.
  - So in the resulting graph, a shortest path to u cannot go through v.
  - Therefore, the insertion of edge (v, u) does not affect δ_f.
- So after we insert all the new edges, δ_f is unchanged.
- Then we delete some edges.
  - When we delete an edge, δ_f cannot decrease.
- So overall, $\delta_f(s, v)$ cannot decrease for any vertex v.

The Edmonds-Karp Algorithm: Proof of the Theorem

It suffices to prove that there are $O(|V| \cdot |E|)$ flow augmentations.
Proof done in class.

Integer Values

- If all the capacities $c(u, v)$ are integers, then the Ford-Fulkerson algorithm (both the basic version and the Edmonds-Karp algorithm) never introduce any number that is not an integer. It follows that:

Theorem (integrality theorem)

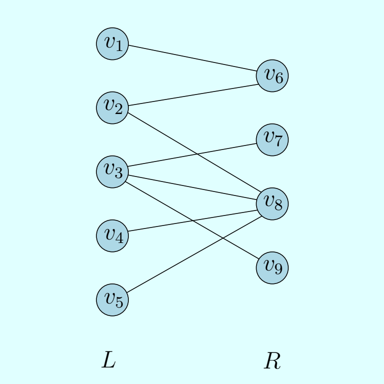
If the capacity function c takes only integral values, then the maximum flow f^* produced by the Ford-Fulkerson method is such that for all u, v, the value $f^*(u, v)$ is an integer. Thus, the value $|f^*|$ of a maximum flow is an integer.

Maximum Bipartite Matching

Definition

A graph $G = (V, E)$ is *bipartite* if its vertex set V can be partitioned into two sets L, R such that $E \subseteq L \times R$.

Example:

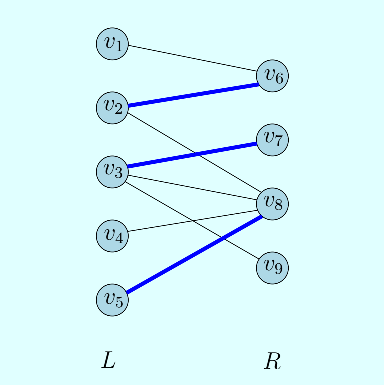


Maximum Bipartite Matching

Definition

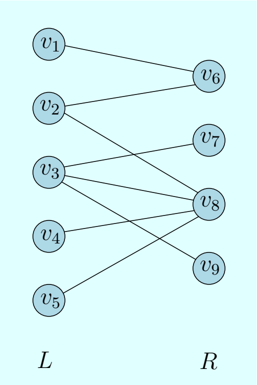
A *maximum bipartite matching* of a bipartite graph G is a matching in G with maximum cardinality.

Example:

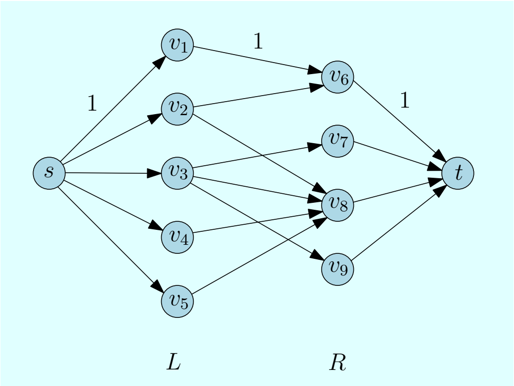


Maximum Bipartite Matching and Maximum Flow

The problem of computing a maximum bipartite matching reduces to computing a maximum flow.

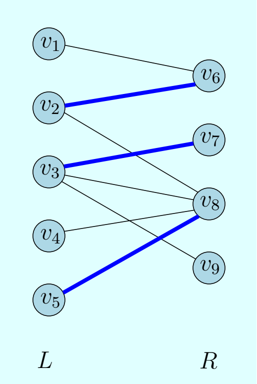


Instance G of maximum bipartite matching.

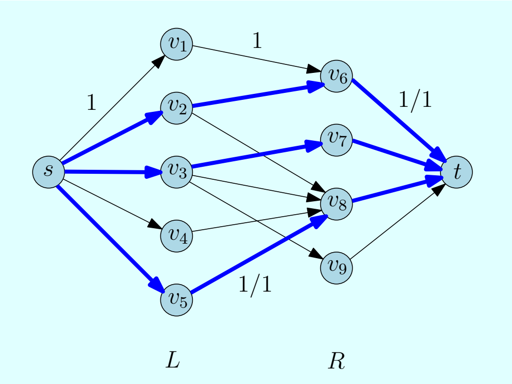


The corresponding flow network G'. All capacities $c(u, v)$ are set to 1.

Maximum Bipartite Matching and Maximum Flow



A maximum bipartite matching in G.



The corresponding maximum flow in G'.

Maximum Bipartite Matching and Maximum Flow

Let $G = (V, E)$ be an instance of maximum bipartite matching, with V partitioned into L, R, and all edges in $L \times R$.

As above, we transform it into a flow network $G'(V', E')$ such that:

- $V' = V \cup \{s, t\}$.
- $E' = E \cup (\{s\} \times L) \cup (R \times \{t\})$.
- $c(u, v) = 1$ for all $(u, v) \in E'$.

We say that a flow f is *integer-valued* if $f(u, v)$ is an integer for all (u, v).

Lemma

If M is a matching in G, then there is an integer-valued flow f in G' with value $|f| = |M|$. Conversely, if f is an integer-valued flow in G', then there is a matching M in G with cardinality $|M| = |f|$.

Proof done in class.

Maximum Bipartite Matching and Maximum Flow

So it follows from the integrality theorem that:

Corollary

The cardinality of a maximum matching M^* in a bipartite graph G is the value $|f^*|$ of a maximum flow in the corresponding flow network G'.

Thus, using the Edmonds-Karp algorithm:

Corollary

We can compute a maximum matching in a bipartite graph $G(V, E)$ in time $O(|V| \cdot |E|^2)$.