Introduction to Algorithms

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

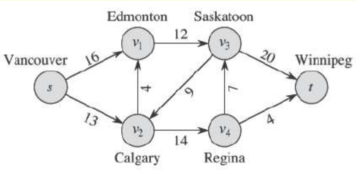
LSSU Math 400

Subsection 1

Flow Networks

Flow Networks

- A **flow network** $G = (V, E)$ is a directed graph in which each edge $(u, v) \in E$ has a nonnegative **capacity** $c(u, v) \geq 0$.
  - We require that if E contains (u, v), then there is no edge (v, u).
  - If $(u, v) \notin E$, then we define $c(u, v) = 0$, and disallow self-loops.
- We distinguish a **source** vertex s and a **sink** vertex t.
- For convenience, we assume that each vertex lies on some path from the source to the sink. That is, for each vertex $v \in V$, the flow network contains a path $s \rightsquigarrow v \rightsquigarrow t$.
- The graph is therefore connected and, since each vertex other than s has at least one entering edge, $|E| \geq |V| - 1$.



Flows and the Max Flow Problem

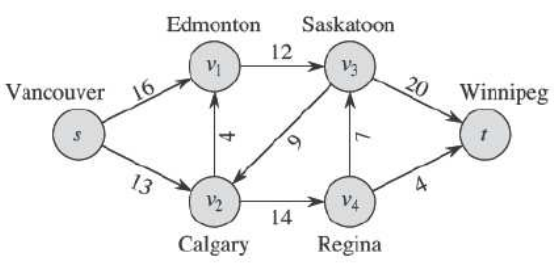
- Let $G = (V, E)$ be a flow network with a capacity function c.
- Let s be the source of the network, and let t be the sink.
- A **flow** in G is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the following two properties:

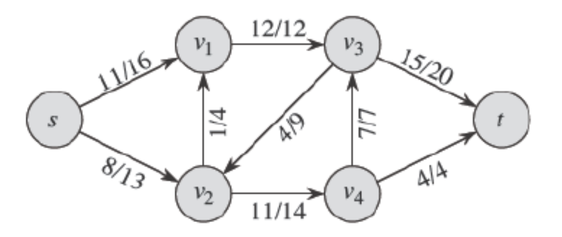
  Capacity constraint: For all $u, v \in V$, $0 \leq f(u, v) \leq c(u, v)$.

  Flow conservation: For all $u \in V - \{s, t\}$, $\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$.
- When $(u, v) \notin E$, there can be no flow from u to v, and $f(u, v) = 0$.
- We call $f(u, v)$ the **flow** from u to v.
- The **value** of f is defined as $|f| = \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s)$.
- **Maximum Flow Problem**:

  Given a flow network G with source s and sink t, find a flow of maximum value.

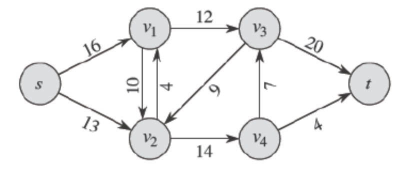
Example of Network Flow

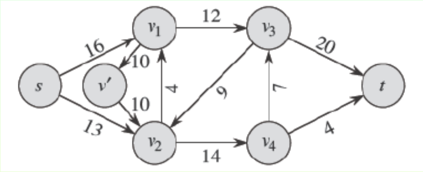




Modeling Problems with Antiparallel Edges

- We call two edges (v_1, v_2) and (v_2, v_1) **antiparallel**.

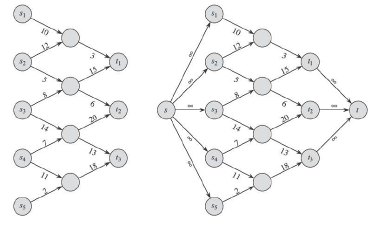




- Since we disallowed such edges in a flow network, to model such edges, we transform the network into an equivalent one not containing antiparallel edges.
  - We choose one of the two antiparallel edges, say (v_1, v_2), and split it by adding a new vertex v';
  - We replace edge (v_1, v_2) with the pair of edges (v_1, v') and (v', v_2).
  - We set the capacity of both new edges to the capacity of the original.
- The resulting network is equivalent to the original one and satisfies the flow network conditions.

Networks with Multiple Sources and Sinks

- A maximum-flow problem may have several sources $\{s_1, s_2, \ldots, s_m\}$ and sinks $\{t_1, t_2, \ldots, t_n\}$, rather than just one of each.
- We can reduce the problem of determining a maximum flow in a network with multiple sources and multiple sinks to an ordinary maximum-flow problem.
- We convert such a network to an ordinary flow network.



- We add a supersource s and add a directed edge (s, s_i) with capacity $c(s, s_i) = \infty$, for $i = 1, 2, \ldots, m$.
- We add a supersink t and a directed edge (t_i, t), with capacity $c(t_i, t) = \infty$, for each $i = 1, 2, \ldots, n$.

The single source s simply provides as much flow as desired for the multiple sources s_i, and the single sink t likewise consumes as much flow as desired for the multiple sinks t_i.

Subsection 2

The Ford-Fulkerson Method

The Ford-Fulkerson Method

- The Ford-Fulkerson method for solving the maximum flow problem iteratively increases the value of the flow.
  - Start with $f(u, v) = 0$, for all $u, v \in V$, giving an initial flow of value 0.
  - At each iteration, increase the flow value in G by finding an "augmenting path" in an associated "residual network" G_f.
    Once we know the edges of an augmenting path in G_f, we can easily identify specific edges in G for which we can change the flow so that we increase the value of the flow.
- Although each iteration of the Ford-Fulkerson method increases the value of the flow, we shall see that the flow on any particular edge of G may increase or decrease.

  Decreasing the flow on some edges may be necessary in order to enable an algorithm to send more flow from the source to the sink.
- We repeatedly augment the flow until the residual network has no more augmenting paths.
- The max-flow min-cut theorem assures a max flow at termination.

The Ford-Fulkerson Procedure

FordFulkersonMethod(G, s, t)

1. initialize flow f to 0
2. while there exists an augmenting path p in the residual network G_f
3. augment flow f along p
4. return f

- Intuitively, given a flow network G and a flow f, the residual network G_f consists of edges with capacities that represent how we can change the flow on edges of G.

Description of the Residual Networks

- An edge of the flow network can admit an amount of additional flow equal to the edge's capacity minus the flow on that edge.
- If that value is positive, we place that edge into G_f with a "residual capacity" of $c_f(u,v) = c(u,v) - f(u,v)$.
  - The only edges of G in G_f are those that can admit more flow;
  - Those edges (u,v) whose flow equals their capacity have $c_f(u,v) = 0$, and they are not in G_f.
  - G_f may also contain edges that are not in G:
    - To increase the total flow, we may need to decrease the flow $f(u,v)$ on a particular edge in G.
    - So we place an edge (v,u) into G_f with residual capacity $c_f(v,u) = f(u,v)$, i.e., an edge that can admit flow in the opposite direction to (u,v), at most canceling out the flow on (u,v).
    - These reverse edges in the residual network allow an algorithm to send back flow it has already sent along an edge amounting to decreasing the flow on the edge.

Formal Treatment of Residual Networks

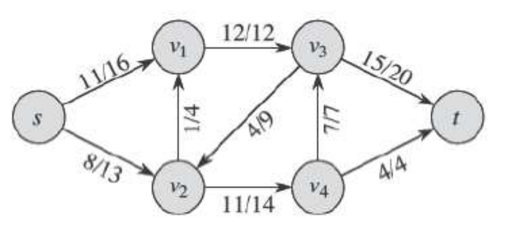
- Consider a flow network $G = (V, E)$ with source s and sink t.
- Let f be a flow in G, and consider a pair of vertices $u, v \in V$.
- We define the **residual capacity** $c_f(u, v)$ by

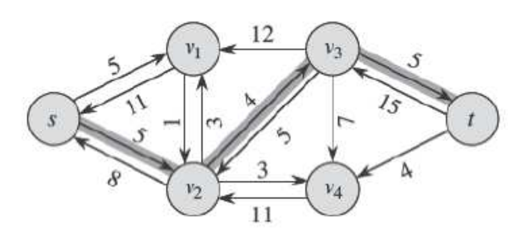
$$c_f(u,v) = \begin{cases} c(u,v) - f(u,v), & \text{if } (u,v) \in E \\ f(v,u), & \text{if } (v,u) \in E \\ 0, & \text{otherwise} \end{cases}$$

- Since $(u, v) \in E$ implies $(v, u) \notin E$, exactly one case in the preceding equation applies to each ordered pair of vertices.
- Given a flow network $G = (V, E)$ and a flow f, the **residual network of** G **induced by** f is $G_f = (V, E_f)$, where

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

Example


12/12
11/16
15/20
1/4
4/9
7/7
8/13
4/4
11/14
s
t


12
5
5
11
15
1
3
4
5
7
5
8
3
4
11
s
t

Augmentations

- Observe that the residual network G_f is similar to a flow network with capacities given by c_f.
- It does not satisfy our definition of a flow network because it may contain both an edge (u, v) and its reversal (v, u).
- Other than this difference, a residual network has the same properties as a flow network, and we can define a flow in the residual network G_f with respect to capacities c_f.
- If f is a flow in G and f' is a flow in the corresponding residual network G_f, we define $f \uparrow f'$, the **augmentation of flow f by f'**, to be a function from $V \times V$ to $\mathbb{R}$, defined by

$$(f \uparrow f')(u, v) = \begin{cases} f(u, v) + f'(u, v) - f'(v, u), & \text{if } (u, v) \in E \\ 0, & \text{otherwise} \end{cases}$$

  - The flow on (u, v) is increased by $f'(u, v)$;
  - The flow on (u, v) is decreased by $f'(v, u)$ (**cancelation**).

Flow After Augmentation (Capacities)

Lemma

Let $G = (V, E)$ be a flow network with source s and sink t and let f be a flow in G. Let G_f be the residual network of G induced by f and let f' be a flow in G_f. Then $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

Claim: $f \uparrow f'$ obeys the capacity constraint for each edge in E and flow conservation at each vertex in $V - \{s, t\}$.

For the capacity constraint, if $(u, v) \in E$, then $c_f(v, u) = f(u, v)$. Hence $f'(v, u) \leq c_f(v, u) = f(u, v)$. So we get

$$\begin{array}{lcl}
(f \uparrow f')(u, v) & = & f(u, v) + f'(u, v) - f'(v, u) \\
 & \geq & f(u, v) + f'(u, v) - f(u, v) = f'(u, v) \geq 0. \\
(f \uparrow f')(u, v) & = & f(u, v) + f'(u, v) - f'(v, u) \\
 & \leq & f(u, v) + f'(u, v) \leq f(u, v) + c_f(u, v) \\
 & = & f(u, v) + c(u, v) - f(u, v) = c(u, v).
\end{array}$$

Flow After Augmentation (Flow Conservation)

- For flow conservation, because both f and f' obey flow conservation, we have, for all $u \in V - \{s, t\}$,

$$\begin{array}{rcl}
\sum_{v \in V} (f \uparrow f')(u, v) & = & \sum_{v \in V} (f(u, v) + f'(u, v) - f'(v, u)) \\
& = & \sum_{v \in V} f(u, v) + \sum_{v \in V} f'(u, v) \\
& & \qquad - \sum_{v \in V} f'(v, u) \\
& = & \sum_{v \in V} f(v, u) + \sum_{v \in V} f'(v, u) \\
& & \qquad - \sum_{v \in V} f'(u, v) \\
& & \text{(by flow conservation)} \\
& = & \sum_{v \in V} (f(v, u) + f'(v, u) - f'(u, v)) \\
& = & \sum_{v \in V} (f \uparrow f')(v, u).
\end{array}$$

Value of Flow After Augmentation

- Finally, we compute the value of $(f \uparrow f')$.

  Recall that we disallow antiparallel edges in G (but not in G_f).

  Hence, for each vertex $v \in V$, we know that there can be an edge (s, v) or (v, s), but never both.

  We define:
  - $V_1 = \{v : (s, v) \in E\}$, the set of vertices with edges from s;
  - $V_2 = \{v : (v, s) \in E\}$, the set of vertices with edges to s.

  We have $V_1 \cup V_2 \subseteq V$ and, since we disallow antiparallel edges, $V_1 \cap V_2 = \emptyset$.

  We now compute

$$\begin{array}{rcl} |f \uparrow f'| & = & \sum_{v \in V}(f \uparrow f')(s, v) - \sum_{v \in V}(f \uparrow f')(v, s) \\ & = & \sum_{v \in V_1}(f \uparrow f')(s, v) - \sum_{v \in V_2}(f \uparrow f')(v, s). \\ & & ((f \uparrow f')(w, x) = 0, \text{ if } (w, x) \notin E.) \end{array}$$

Value of Flow After Augmentation (Cont'd)

- Next, we apply the definition of $f \uparrow f'$ to the preceding equation, and then reorder and group terms to obtain

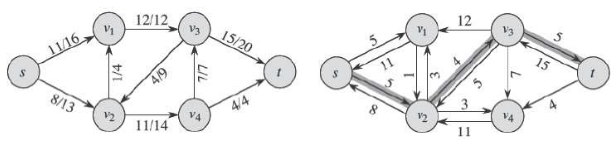
$$\begin{aligned}
|f \uparrow f'| &= \textstyle\sum_{v \in V_1} (f(s,v) + f'(s,v) - f'(v,s)) \\
&\quad - \textstyle\sum_{v \in V_2} (f(v,s) + f'(v,s) - f'(s,v)) \\
&= \textstyle\sum_{v \in V_1} f(s,v) + \sum_{v \in V_1} f'(s,v) - \sum_{v \in V_1} f'(v,s) \\
&\quad - \textstyle\sum_{v \in V_2} f(v,s) - \sum_{v \in V_2} f'(v,s) + \sum_{v \in V_2} f'(s,v) \\
&= \textstyle\sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) + \sum_{v \in V_1} f'(s,v) \\
&\quad + \textstyle\sum_{v \in V_2} f'(s,v) - \sum_{v \in V_1} f'(v,s) - \sum_{v \in V_2} f'(v,s) \\
&= \textstyle\sum_{v \in V_1} f(s,v) - \sum_{v \in V_2} f(v,s) \\
&\quad + \textstyle\sum_{v \in V_1 \cup V_2} f'(s,v) - \sum_{v \in V_1 \cup V_2} f'(v,s).
\end{aligned}$$

  We can extend all four summations to sum over V, since each additional term has value 0. We thus have

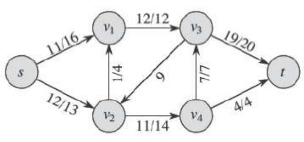
$$|f \uparrow f'| = \sum_{v \in V} f(s,v) - \sum_{v \in V} f(v,s) + \sum_{v \in V} f'(s,v) - \sum_{v \in V} f'(v,s) = |f| + |f'|.$$

Augmenting Paths and Residual Capacity

- Given a flow network $G = (V, E)$ and a flow f, an **augmenting path** p is a simple path from s to t in the residual network G_f.



- By the definition of the residual network, we may increase the flow on an edge (u, v) of an augmenting path by up to $c_f(u, v)$ without violating a capacity constraint in G.



- We call the maximum amount by which we can increase the flow on each edge in an augmenting path p the **residual capacity** of p, given by
$$c_f(p) = \min\{c_f(u, v) : (u, v) \text{ is on } p\}.$$

Flow Along an Augmenting Path

Lemma

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

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

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

Increasing a Flow via Augmenting Paths

- The following corollary shows that if we augment f by f_p, we get another flow in G whose value is closer to the maximum.

Corollary

Let $G = (V, E)$ be a flow network, f a flow in G and p an augmenting path in G_f. Let f_p be the flow along path p, and suppose that we augment f by f_p. Then the function $f \uparrow f_p$ is a flow in G with value

$$|f \uparrow f_p| = |f| + |f_p| > |f|.$$

- Immediate from preceding lemmas.

Cuts of Flow Networks

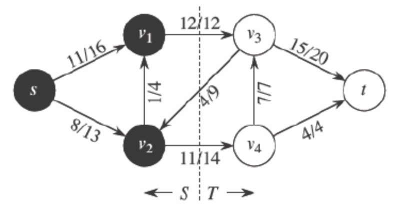
- The Ford-Fulkerson method repeatedly augments the flow along augmenting paths until it has found a maximum flow.
- When the algorithm terminates, we have actually found a maximum flow, since the max-flow min-cut theorem tells us that a flow is maximum if and only if its residual network contains no augmenting path.
- A **cut** (S, T) of flow network $G = (V, E)$ is a partition of V into S and $T = V - S$, such that $s \in S$ and $t \in T$.
- If f is a flow, then the **net flow** $f(S, T)$ **across the cut** (S, T) is defined to be

$$f(S, T) = \sum_{u \in S} \sum_{v \in T} f(u, v) - \sum_{u \in S} \sum_{v \in T} f(v, u).$$

Capacity of Cuts and Minimum Cuts

- The **capacity** of a cut (S, T) is
$$c(S, T) = \sum_{u \in S} \sum_{v \in T} c(u, v).$$
- A **minimum cut** of a network is a cut whose capacity is minimum over all cuts of the network.
- Note the following important difference:
  - For capacity we count only the edges going from S to T;
  - For flow, we consider flow from S to T minus flow from T to S.

Example: Consider the cut $(\{s, v_1, v_2\}, \{v_3, v_4, t\})$ of the figure.



The net flow across this cut is $f(v_1, v_3) + f(v_2, v_4) - f(v_3, v_2) = 12 + 11 - 4 = 19.$
The capacity of the cut is $c(v_1, v_3) + c(v_2, v_4) = 12 + 14 = 26.$

Cuts and Flows

Lemma

Let f be a flow in a flow network G with source s and sink t and let (S, T) be any cut of G. Then the net flow across (S, T) is $f(S, T) = |f|$.

- We can rewrite the flow conservation condition for any node $u \in V - \{s, t\}$ as $\sum_{v \in V} f(u, v) - \sum_{v \in V} f(v, u) = 0$.
  Taking the definition of $|f|$ and adding the left-hand side of the preceding equation, summed over all vertices in $S - \{s\}$, gives

$$\begin{aligned}
|f| &= \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) \\
&\quad + \sum_{u \in S - \{s\}} (\sum_{v \in V} f(u, v) - \sum_{v \in V} f(v, u)) \\
&= \sum_{v \in V} f(s, v) - \sum_{v \in V} f(v, s) \\
&\quad + \sum_{u \in S - \{s\}} \sum_{v \in V} f(u, v) - \sum_{u \in S - \{s\}} \sum_{v \in V} f(v, u) \\
&= \sum_{v \in V} (f(s, v) + \sum_{u \in S - \{s\}} f(u, v)) \\
&\quad - \sum_{v \in V} (f(v, s) + \sum_{u \in S - \{s\}} f(v, u)) \\
&= \sum_{v \in V} \sum_{u \in S} f(u, v) - \sum_{v \in V} \sum_{u \in S} f(v, u).
\end{aligned}$$

Cuts and Flows (Cont'd)

- Because $V = S \cup T$ and $S \cap T = \emptyset$, we can split each summation over V into summations over S and T.

$$\begin{array}{rcl} |f| & = & \sum_{v \in S} \sum_{u \in S} f(u,v) + \sum_{v \in T} \sum_{u \in S} f(u,v) \\ & & - \sum_{v \in S} \sum_{u \in S} f(v,u) - \sum_{v \in T} \sum_{u \in S} f(v,u) \\ & = & \sum_{v \in T} \sum_{u \in S} f(u,v) - \sum_{v \in T} \sum_{u \in S} f(v,u) \\ & & + (\sum_{v \in S} \sum_{u \in S} f(u,v) - \sum_{v \in S} \sum_{u \in S} f(v,u)). \end{array}$$

The two summations within the parentheses are actually the same, since for all $x, y \in V$, $f(x,y)$ appears once in each summation.

Hence, these summations cancel, and we have

$$|f| = \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) = f(S,T).$$

Cut Capacities Bound the Value of a Flow

Corollary

The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

- Let (S, T) be any cut of G and let f be any flow. By the lemma and the capacity constraint,

$$\begin{array}{rcl} |f| & = & f(S,T) \\ & = & \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{u \in S} \sum_{v \in T} f(v,u) \\ & \leq & \sum_{u \in S} \sum_{v \in T} f(u,v) \\ & \leq & \sum_{u \in S} \sum_{v \in T} c(u,v) \\ & = & c(S,T). \end{array}$$

- Therefore, the value of a maximum flow in a network is bounded from above by the capacity of a minimum cut of the network.

The Max-Flow Min-Cut Theorem

- The value of a maximum flow is in fact equal to the capacity of a minimum cut.

Theorem (The Max-Flow Min-Cut Theorem)

If f is a flow in a flow network $G = (V, E)$ with source s and sink t, then the following conditions are equivalent:

1. f is a maximum flow in G.
2. The residual network G_f contains no augmenting paths.
3. $|f| = c(S, T)$, for some cut (S, T) of G.

1⇒2: Suppose for the sake of contradiction that f is a maximum flow in G but that G_f has an augmenting path p. Then, the flow found by augmenting f by f_p is a flow in G with value strictly greater than $|f|$. This contradicts the assumption that f is a maximum flow.

The Max-Flow Min-Cut Theorem (Cont'd)

2⇒3: Suppose that G_f has no augmenting path, that is, that G_f contains no path from s to t.

Define

$$\begin{array}{rcl} S & = & \{v \in V : \text{there exists a path from } s \text{ to } v \text{ in } G_f\}; \\ T & = & V - S. \end{array}$$

The partition (S, T) is a cut: $s \in S$, and $t \notin S$ because there is no path from s to t in G_f.

Consider a pair of vertices $u \in S$, $v \in T$.

- If $(u, v) \in E$, we must have $f(u, v) = c(u, v)$, since otherwise $(u, v) \in E_f$, which would place v in set S.
- If $(v, u) \in E$, we must have $f(v, u) = 0$, because otherwise $c_f(u, v) = f(v, u)$ would be positive and we would have $(u, v) \in E_f$, which would place v in S.
- If neither (u, v) nor (v, u) is in E, then $f(u, v) = f(v, u) = 0$.

The Max-Flow Min-Cut Theorem (Conclusion)

We thus have

$$\begin{aligned} f(S,T) &= \sum_{u \in S} \sum_{v \in T} f(u,v) - \sum_{v \in T} \sum_{u \in S} f(v,u) \\ &= \sum_{u \in S} \sum_{v \in T} c(u,v) - \sum_{v \in T} \sum_{u \in S} 0 \\ &= c(S,T). \end{aligned}$$

Therefore, $|f| = f(S,T) = c(S,T)$.

3⇒1: $|f| \le c(S,T)$, for all cuts (S,T). The condition $|f| = c(S,T)$, thus, implies that f is a maximum flow.

The Basic Ford-Fulkerson Algorithm

- In each iteration of the Ford-Fulkerson method, we find some augmenting path p and use p to modify the flow f.

  We replace f by $f \uparrow f_p$, obtaining a new flow of value $|f| + |f_p|$.
- The following implementation computes the maximum flow in a flow network $G = (V, E)$ by updating the flow attribute $(u, v).f$ for each edge $(u, v) \in E$.
  - If $(u, v) \notin E$, we assume implicitly that $(u, v).f = 0$.
  - We also assume that we are given the capacities $c(u, v)$ along with the flow network, and $c(u, v) = 0$, if $(u, v) \notin E$.
- We compute the residual capacity in accordance with

$$c_f(u, v) = \begin{cases} c(u, v) - f(u, v), & \text{if } (u, v) \in E \\ f(v, u), & \text{if } (v, u) \in E \\ 0, & \text{otherwise} \end{cases}.$$

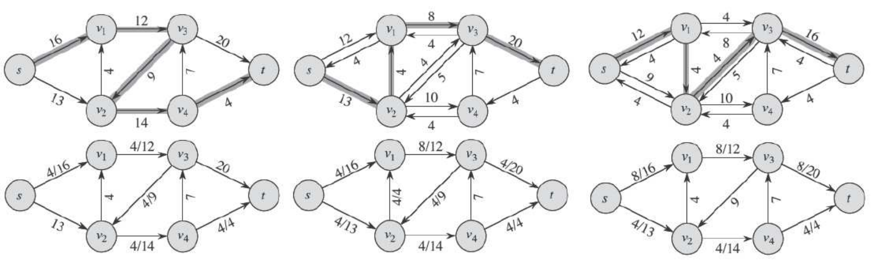
- The expression $c_f(p)$ is a temporary variable that stores the residual capacity of the path p.

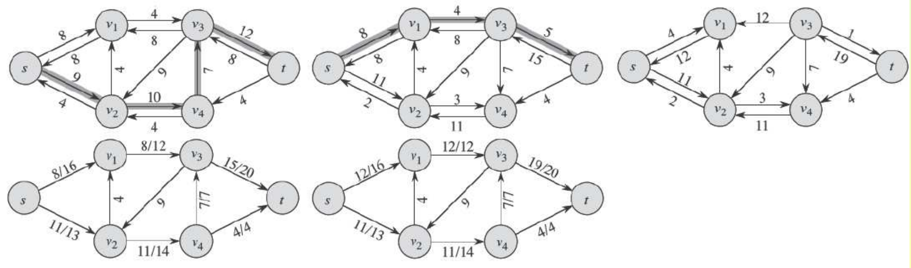
The Basic Ford-Fulkerson Procedure

FordFulkerson(G, s, t)

```
for each edge (u, v) ∈ G.E
    (u, v).f = 0
while there exists a path p from s to t in the residual network G_f
    c_f(p) = min {c_f(u, v) : (u, v) is in p}
    for each edge (u, v) in p
        if (u, v) ∈ E
            (u, v).f = (u, v).f + c_f(p)
        else (v, u).f = (v, u).f − c_f(p)
```

Illustrating the FordFulkerson





How FordFulkerson Works

- Lines 1-2 initialize the flow f to 0.
- The while loop of Lines 3-8 repeatedly finds an augmenting path p in G_f and augments flow f along p by the residual capacity $c_f(p)$.

  Each residual edge in path p is either an edge in the original network or the reversal of an edge in the original network.
  - Lines 6-8 update the flow in each case appropriately, adding flow when the residual edge is an original edge and subtracting it otherwise.
- When no augmenting paths exist, the flow f is a maximum flow.

Analysis of Ford-Fulkerson

- The running time of FORDFULKERSON depends on how we find the augmenting path p in Line 3.

  If we find the augmenting path by using a breadth-first search, the algorithm runs in polynomial time.
- Before proving this result, we obtain a simple bound for the case in which we choose the augmenting path arbitrarily and all capacities are integers.
  - If f^* denotes a maximum flow in the transformed network, then a straightforward implementation of FORDFULKERSON executes the while loop of Lines 3-8 at most $|f^*|$ times, since the flow value increases by at least one unit in each iteration.
  - We can perform the work done within the while loop efficiently if we implement the flow network $G = (V, E)$ with the right data structure and find an augmenting path by a linear-time algorithm.

Analysis of Ford-Fulkerson: Implementation Details

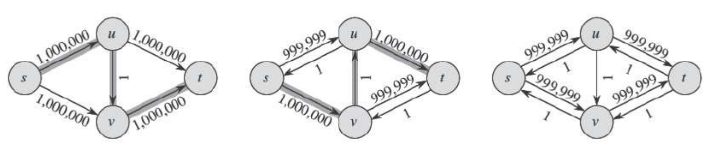
- We assume that we keep a data structure corresponding to a directed graph $G' = (V, E')$, where $E' = \{(u, v) : (u, v) \in E \text{ or } (v, u) \in E\}$.
- Edges in the network G are also edges in G', and therefore we can easily maintain capacities and flows in this data structure.
- Given a flow f on G, the edges in the residual network G_f consist of all edges (u, v) of G', such that $c_f(u, v) > 0$.
- The time to find a path in a residual network is therefore $O(|V| + |E'|) = O(|E|)$, if we use either depth-first search or breadth-first search.
- Each iteration of the while loop thus takes $O(|E|)$ time, as does the initialization in Lines 1-2.
- So the total running time of the FordFulkerson is $O(|E||f^*|)$.

Subsection 3

The Edmonds-Karp Algorithm

An Unfortunate Scenario

- In the flow network a maximum flow has value 2,000,000:



  - 1,000,000 units of flow traverse the path $s \to u \to t$;
  - Another 1,000,000 units traverse the path $s \to v \to t$.
- If the first augmenting path found by FordFulkerson is $s \to u \to v \to t$, the flow has value 1 after the first iteration.
- If the second iteration finds the augmenting path $s \to v \to u \to t$, the flow then has value 2.
- If we continue choosing:
  - the augmenting path $s \to u \to v \to t$ in the odd-numbered iterations;
  - the augmenting path $s \to v \to u \to t$ in the even-numbered iterations,

  we would perform a total of 2,000,000 augmentations, increasing the flow value by only 1 unit in each.

The Edmonds-Karp Version

- The bound on FORDFULKERSON can be improved if we implement the computation of the augmenting path p in Line 3 using breadth first search.

  We choose the augmenting path as a shortest path from s to t in the residual network, where each edge has unit distance (weight).
- We call the Ford-Fulkerson method so implemented the **Edmonds-Karp algorithm**.
- We prove that the Edmonds-Karp algorithm runs in $O\left(|V||E|^2\right)$ time.
- The analysis depends on the distances to vertices in the residual network G_f.
- We use the notation $\delta_f(u, v)$ for the shortest-path distance from u to v in G_f, where each edge has unit distance.

Monotonicity of the Shortest-Path Distance $\delta_f(s,v)$ in G_f

Lemma

If the Edmonds-Karp algorithm is run on a flow network $G=(V,E)$ with source s and sink t, then for all vertices $v \in V-\{s,t\}$, the shortest-path distance $\delta_f(s,v)$ in the residual network G_f increases monotonically with each flow augmentation.

- We will suppose that for some vertex $v \in V-\{s,t\}$, there is a flow augmentation that causes the shortest-path distance from s to v to decrease and derive a contradiction.

  Let f be the flow just before the first augmentation that decreases some shortest-path distance. Let f' be the flow just afterward. Let v be the vertex with the minimum $\delta_{f'}(s,v)$ whose distance was decreased by the augmentation, so that $\delta_{f'}(s,v) < \delta_f(s,v)$.

  Let $p = s \rightsquigarrow u \rightarrow v$ be a shortest path from s to v in $G_{f'}$.

  So $(u,v) \in E_{f'}$ and $\delta_{f'}(s,u) = \delta_{f'}(s,v) - 1$.

Monotonicity of $\delta_f(s, v)$ (Cont'd)

- Because of the choice of v, we know that the distance label of vertex u did not decrease, i.e., $\delta_{f'}(s, u) \geq \delta_f(s, u)$.

  Claim: $(u, v) \notin E_f$.

  If $(u, v) \in E_f$, then $\delta_f(s, v) \leq \delta_f(s, u) + 1 \leq \delta_{f'}(s, u) + 1 = \delta_{f'}(s, v)$. This contradicts our assumption that $\delta_{f'}(s, v) < \delta_f(s, v)$.

  To have $(u, v) \notin E_f$ and $(u, v) \in E_{f'}$, the augmentation must have increased the flow from v to u. The Edmonds-Karp algorithm always augments flow along shortest paths. Therefore the shortest path from s to u in G_f has (v, u) as its last edge. So

  $$\delta_f(s, v) = \delta_f(s, u) - 1 \leq \delta_{f'}(s, u) - 1 = \delta_{f'}(s, v) - 2.$$

  This contradicts the assumption that $\delta_{f'}(s, v) < \delta_f(s, v)$.

  So our assumption that such a vertex v exists is incorrect.

Number of Iterations of Edmonds-Karp Algorithm

Theorem

If the Edmonds-Karp algorithm is run on a flow network $G = (V, E)$ with source s and sink t, then the total number of flow augmentations performed by the algorithm is $O(|V||E|)$.

- We say that an edge (u, v) in a residual network G_f is **critical** on an augmenting path p if the residual capacity of p is the residual capacity of (u, v), i.e., if $c_f(p) = c_f(u, v)$.
  - After we have augmented flow along an augmenting path, any critical edge on the path disappears from the residual network.
  - Moreover, at least one edge on any augmenting path must be critical.

Claim: Each of the $|E|$ edges can become critical at most $\frac{|V|}{2}$ times.

Let u and v be vertices in V that are connected by an edge in E. Since augmenting paths are shortest paths, when (u, v) is critical for the first time, we have $\delta_f(s, v) = \delta_f(s, u) + 1$. Once the flow is augmented, the edge (u, v) disappears from the residual network.

Number of Iterations of Edmonds-Karp (Cont'd)

- Edge (u, v) cannot reappear until after the flow from u to v is decreased, which occurs only if (v, u) appears on an augmenting path. If f' is the flow in G when this event occurs, then we have $\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1$. Since $\delta_f(s, v) \leq \delta_{f'}(s, v)$, we have $\delta_{f'}(s, u) = \delta_{f'}(s, v) + 1 \geq \delta_f(s, v) + 1 = \delta_f(s, u) + 2$.
  Thus, between two critical appearances of (u, v), the distance of u from s increases by at least 2. The intermediate vertices on a shortest path from s to u cannot contain s, u or t. Therefore, until u becomes unreachable from the source, if ever, its distance is at most $|V| - 2$. Thus, after the first time that (u, v) becomes critical, it can become critical at most $\frac{|V|-2}{2} = \frac{|V|}{2} - 1$ times more, for a total of $\leq \frac{|V|}{2}$ times. Hence, the total number of critical edges during the entire execution is $O(|V||E|)$. Since each augmenting path has at least one critical edge, the theorem follows.
- We now get $O(|V||E|^2)$ time for the Edmonds-Karp algorithm.

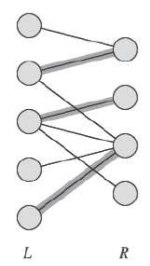
Subsection 4

Maximum Bipartite Matching

The Maximum-Bipartite-Matching Problem

- Given an undirected graph $G=(V,E)$, a **matching** is a subset of edges $M \subseteq E$, such that for all vertices $v \in V$, at most one edge of M is incident on v.

  We say that a vertex $v \in V$ is **matched** by the matching M if some edge in M is incident on v; otherwise, v is **unmatched**.

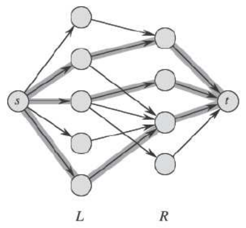


- A **maximum matching** is a matching of maximum cardinality, that is, a matching M such that for any matching M', we have $|M| \geq |M'|$.
- We restrict our attention to finding maximum matchings in **bipartite graphs**, i.e., graphs in which the vertex set can be partitioned into $V = L \cup R$, where L and R are disjoint and all edges in E go between L and R.
- We assume that every vertex in V has at least one incident edge.

Finding a Maximum Bipartite Matching

- We use Ford-Fulkerson to find a maximum matching in an undirected bipartite graph $G = (V, E)$ in time polynomial in $|V|$ and $|E|$.
- We construct a flow network in which flows correspond to matchings.

  We define the corresponding flow network $G' = (V', E')$ for the bipartite graph G as follows:

  - We let the source s and sink t be new vertices not in V.
  - We let $V' = V \cup \{s, t\}$.
  - If the vertex partition of G is $V = L \cup R$, the directed edges of G' are the edges of E, directed from L to R, along with V new edges:

  $$E' = \{(s, u) : u \in L\} \cup \{(u, v) : (u, v) \in E\} \cup \{(v, t) : v \in R\}.$$

  - We assign unit capacity to each edge in E'.
- Since each vertex in V has at least one incident edge, $|E| \geq \frac{|V|}{2}$. Thus, $|E| \leq |E'| = |E| + |V| \leq 3|E|$. So $|E'| = \Theta(E)$.

Matchings in G Correspond to Flows in G'

- A flow f on a flow network $G = (V, E)$ is **integer-valued** if $f(u, v)$ is an integer, for all $(u, v) \in V \times V$.

Lemma

Let $G = (V, E)$ be a bipartite graph with vertex partition $V = L \cup R$, and let $G' = (V', E')$ be its corresponding flow network. 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 $|M| = |f|$.

- We first show that a matching M in G corresponds to an integer-valued flow in G'. Define f as follows:
  - If $(u, v) \in M$, then $f(s, u) = f(u, v) = f(v, t) = 1$.
  - For all other edges $(u, v) \in E'$, we define $f(u, v) = 0$.

  One can easily verify that f satisfies the capacity constraints and flow conservation.

Matchings in G Correspond to Flows in G' (Cont'd)

- Intuitively, each edge $(u, v) \in M$ corresponds to one unit of flow in G' that traverses the path $s \to u \to v \to t$. Moreover, the paths induced by edges in M are vertex-disjoint, except for s and t. The net flow across cut $(L \cup \{s\}, R \cup \{t\})$ is equal to $|M|$, whence the value of the flow is $|f| = |M|$.

Matchings in G Correspond to Flows in G' (Converse)

- To prove the converse, let f be an integer-valued flow in G', and let $M = \{(u,v) : u \in L, v \in R \text{ and } f(u,v) > 0\}$. Each vertex $u \in L$ has only one entering edge, namely (s,u) and its capacity is 1. Thus, each $u \in L$ has at most one unit of flow entering it. If one unit of flow does enter, by flow conservation, one unit of flow must leave. Furthermore, since f is integer-valued, for each $u \in L$, the one unit of flow can enter on at most one edge and can leave on at most one edge. Thus, one unit of flow enters u if and only if there is exactly one vertex $v \in R$, such that $f(u,v) = 1$, and at most one edge leaving each $u \in L$ carries positive flow. A symmetric argument applies to each $v \in R$. The set M is therefore a matching.
  Next, observe that:
  - for every matched vertex $u \in L$, we have $f(s,u) = 1$;
  - for every edge $(u,v) \in E - M$, we have $f(u,v) = 0$.

  Consequently, $f(L \cup \{s\}, R \cup \{t\}) = |M|$. Hence, by a preceding lemma, $|f| = f(L \cup \{s\}, R \cup \{t\}) = |M|$.

The Integrality Theorem

- Finally, by showing that if $|f|$ is an integer, the Ford-Fulkerson returns a flow in G' for which every $f(u, v)$ is an integer, we conclude that a maximum matching in a bipartite graph G corresponds to a maximum flow in its corresponding flow network G'.

Theorem (Integrality Theorem)

If the capacity function c takes on only integral values, then the maximum flow f produced by the Ford-Fulkerson method has the property that $|f|$ is an integer. Moreover, for all vertices u and v, the value of $f(u, v)$ is an integer.

- The proof is by induction on the number of iterations and is omitted.

Maximum Matchings and Maximum Flows

Corollary

The cardinality of a maximum matching M in a bipartite graph G equals the value of a maximum flow f in its corresponding flow network G'.

- Suppose that M is a maximum matching in G and that the corresponding flow f in G' is not maximum. Then there is a maximum flow f' in G', such that $|f'| > |f|$. Since the capacities in G' are integer-valued, by the Integrality Theorem, we can assume that f' is integer-valued. Thus, f' corresponds to a matching M' in G with cardinality $|M'| = |f'| > |f| = |M|$. This contradicts our assumption that M is a maximum matching.

  In a similar manner, we can show that if f is a maximum flow in G', its corresponding matching is a maximum matching on G.

Maximum Matchings via Ford-Fulkerson

- Thus, given a bipartite undirected graph G, we can find a maximum matching by:
  - Creating the flow network G';
  - Running the Ford-Fulkerson method;
  - Obtaining a maximum matching M from the integer-valued maximum flow f found.
- Since any matching in a bipartite graph has cardinality at most $\min(L, R) = O(|V|)$, the value of the maximum flow in G' is $O(|V|)$.
- We can therefore find a maximum matching in a bipartite graph in time $O(|V||E'|) = O(|V||E|)$, since $|E'| = \Theta(|E|)$.