Examn notes for Advanced Algorithms and Datastructures 2014

Martin Jørgensen

June 7, 2014

Dispositions

Max-Flow

1. (Define a Flow Network)
   - (Capacity Constraint)
   - (Flow Conservation)
2. Define a Max Flow
3. (How to have multible source/sink networks)
4. (Introduce Residual Networks)
5. (Introduce Augmenting Paths)
6. Cuts - In particular the min cut max flow
7. Introduce Ford-Fulkerson / Edmonds-Karp

Fibonacci Heaps

1. Mergeable heaps
2. Structure
3. Operations
    - Make-Heap
    - Insert
    - ExtractMin
    - Union/Merge
    - DecreaseKey
    - Delete

Notes

Max-Flow

Flow Network

A flow network $G = (V, E)$ is a directed graph where each edge $(u, v) \in E$ have a nonnegative capacity $c(u, v) \geq 0$. In addition, for any edge (u, v) there can be no antiparallel edge (v, u).

Two vertices in the network have special characteristics the source s and sink t. We assume each vertex $v \in V$ lies on some path from s to t, that is, for each vertex $v \in V$, the flow network contains a path $s \rightsquigarrow v \rightsquigarrow t$.

Flow Definition

We have a flow network $G = (V, E)$ with a source s and a sink t, the network has a capacity function $c(u, v)$. A flow is a real-valued function $f : V \times V \to \mathbb{R}$ that satisfies the two following properties:

- **Capacity Constraint**:
  For all $u, v \in V$, we require $0 \leq f(u, v) \leq c(u, v)$
- **Flow Conservation**:
  For all $u \in V - \{s, t\}$ we require

  $$\sum_{v \in V} f(v, u) = \sum_{v \in V} f(u, v)$$

  When $(u, v) \notin E$, there can be no flow from u to v, and $f(u, v) = 0$. We call the nonnegative quantity $f(u, v)$ the flow from vertex u to vertex v.

The value $|f|$ of a flow f is defined as

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

that is the total flow out of the source minus the flow into the source. For an example of a flow see Figure 1

Antiparallel Edges and Multiple Sources/Sinks

Since a flow network cannot contain anti-parallel edges, but we want to be able to represent them in our graph, we need a way to do so. This is done by inserting an additional node v' and let one of the edges go through this node instead, see Figure 2 for an example.

If a network have multiple sources or sinks, we can convert it to a single source/sink network by adding a supersource and supersink. An example of such conversion can be seen in Figure 3.

Flow Examples

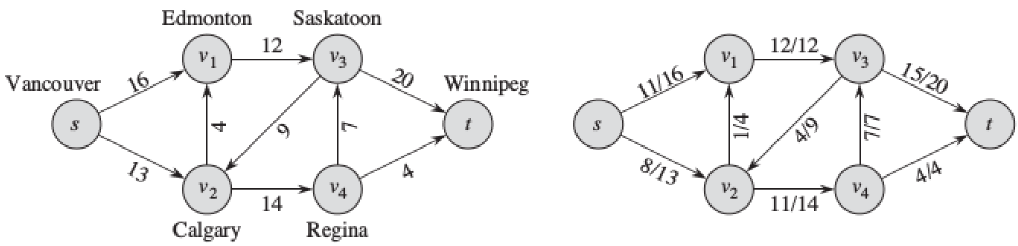

Figure 1: Example flow.

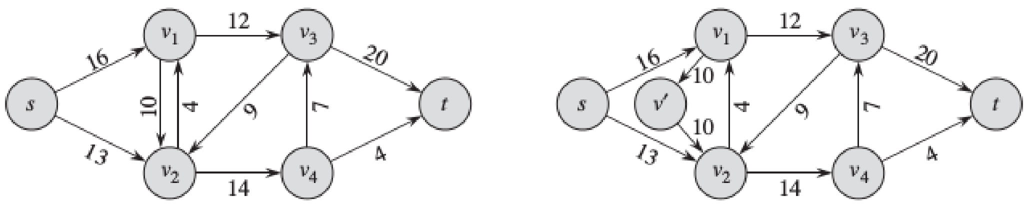

Figure 2: Conversion from antiparallel edges to proper flow.

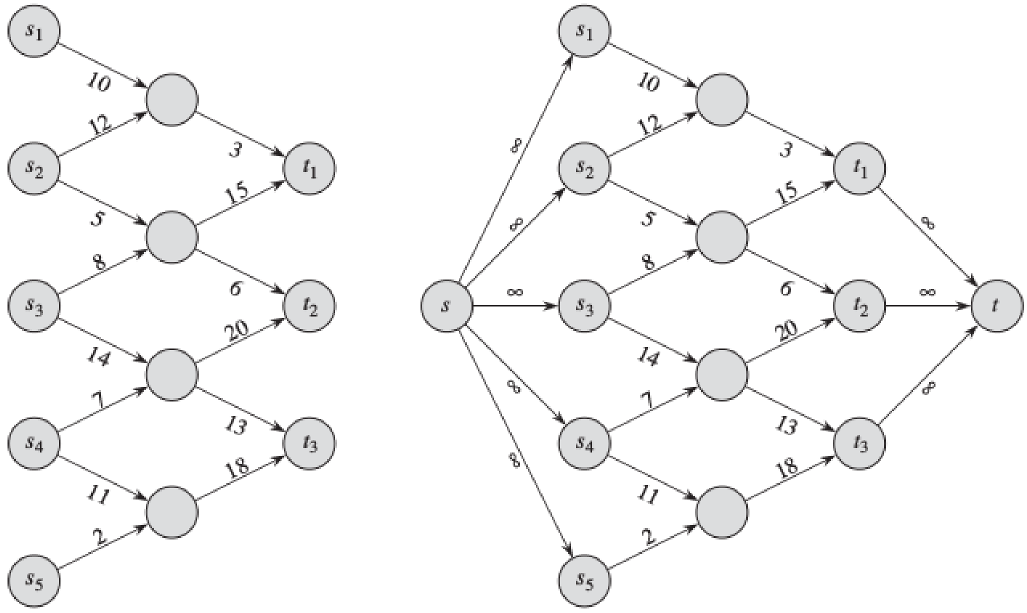

Figure 3: Example of a graph with multiple sources and sink, combined using a supersource and supersink.

Residual Networks

Given a flow network G and a flow f the residual network G_f consists of edges and capacities that represent how we can change the flow on edges of G. Suppose we have 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 then define the residual capacity $c_f(u, v)$ like this:

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

Given a flow network $G = (V, E)$ and a flow f, the residual network og G induced by f is $G_f(V, E_f)$, where

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

An example of a residual network can be seen in Figure 4.

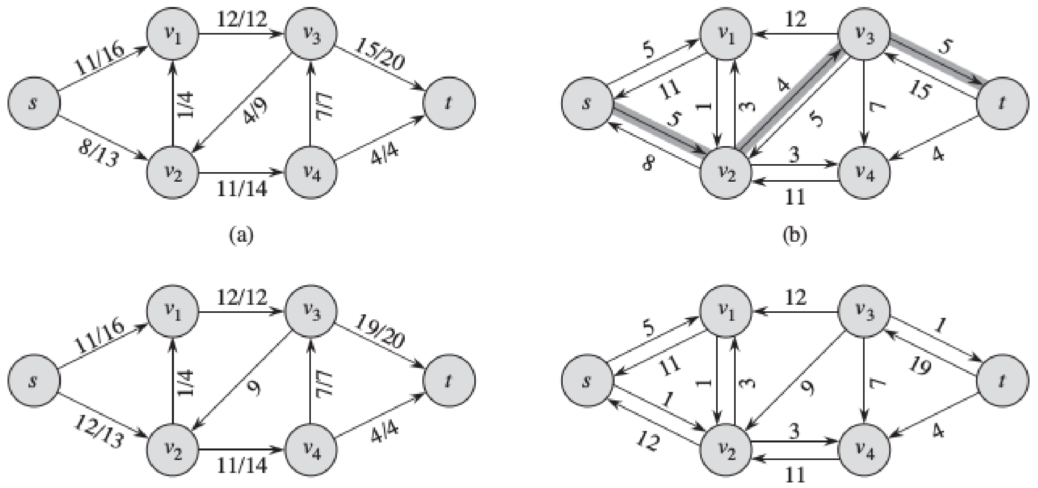

Figure 4: An example of a flow being augmented and showing the residual graph.

Augmenting Flows

Augmenting paths are simply flows that can be added to other flows in order to increase the flow value through the network. Augmenting flows are described using the $\uparrow$ operator like so:

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

an example of an applied augmenting path can bee seen in Figure 4.

Lemma 26.1 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 be induced by f, and let f' be a flow in G_f. Then the function $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f'|$.

Augmenting Paths

Given a network $G = (V, E)$ and a flow f, an augmenting path is a simple path from s to t in the residual network G_f. The shaded path in Figure 4(b) is an augmenting path. We can increase the flow on each edge in the augmenting path p by an amount equal to the residual capacity of p given by

$$c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is on } p\}$$

That is, the smallest amount of spare capacity on any edge in p.

Lemma 26.2 let $G = (V,E)$ be a flownetwork, let f be a flow in G, and let p be 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 flow in G_f with value $|f_p| = c_f(p) > 0$.

Corollary 26.3 let $G = (V,E)$ be a flownetwork, let f be a flow in G, and let p be an augmenting path in G_f. Let f_p be defined as in Lemma 26.2, and suppose that we augment f by f_p. Then the function $f \uparrow f'$ is a flow in G with value $|f \uparrow f'| = |f| + |f_p| > |f|$.

Cuts in flow networks

A cut (S,T) of a 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 isa a flow, then the net flow $f(S,T)$ across the cut (S,T) is defined to be

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

The capacity of the 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 that for capacity we count only edges going from S to T while for flow we count edges going both directions. Lemma 26.4 shows that for a given flow f, the net flow across any cut is the same and it equals $|f|$, the value of the flow.

Lemma 26.4 Let f be a flow in a flow network $G = (V, E)$ with source s and sink t, and let (S, T) be any cut of G. Then the net flow across (S, T) is $f(S, T) = |f|$

Corollary 26.5 The value of any flow f in a flow network G is bounded from above by the capacity of any cut of G.

Max-flow min-cut theorem:

Theorem 26.6 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 mximum 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.

Proof (1) $\implies$ (2): If we assume f is a maximum flow but there still is an augmenting path p in the residual graph G_f, then by Corollary 26.3, the flow found by $f \uparrow f_p$ is a flow with value strictly greater than |f|, contradicting the assumption that f is a max flow.

(2) $\implies$ (3): Suppose G_f has no augmenting paths, that is, that G_f contains no paths from s to t. We define

$$S = \{v \in V : \text{there exists a path from } s \text{ to } v \text{ in } G_f\}$$

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

We now consider a pair of vertices $u \in S$ and $v \in T$. If $(u, v) \in E$, we must have $f(u, v) = c(u, v)$ otherwise $(u, v) \in E_f$ wich would place v in 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 $(u, v) \in E_f$, whouch would place v in S. If neither (u, v) or (v, u) is in E, then $f(u, v) = f(v, u) = 0$. We thus have

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

By Lemma 26.4, we then have $|f| = f(S,T) = c(S,T)$.

(3) $\implies$ (1): By corollary 26.5, $|f| \leq c(S,T)$ for all cuts (S,T). The condition $|f| = c(S,T)$ thus implies that f is a maximum flow. $\square$

Ford-Fulkerson

The general algorithm:

FORD-FULKERSON-METHOD(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

Implementation:

FORD-FULKERSON(G, s, t)

1 **for** each edge $(u, v) \in G.E$
2     $(u, v).f = 0$
3 **while** there exists a path p from s to t in the residual network G_f
4     $c_f(p) = \min\{c_f(u,v) : (u,v) \text{ is in } p\}$
5   **for** each edge (u, v) in p
6   **if** $(u, v) \in E$
7             $(u, v).f = (u, v).f + c_f(p)$
8   **else** $(v, u).f = (v, u).f - c_f(p)$

Assume we can pick the path p in linear time, the loop header is running in $O(E)$. If f^* denote a maximum flow, then the while loop is executed at most $|f^*|$ times, since each augmentation must increase the flow value with at least one. The for loop inside the loop can be done in $O(E)$ since the longest p can be no longer than $|E|$. Giving a running time of $O(|f^*|\dot{(}E+E)) = O(E|f^*|)$.

Edmonds-Karp

Is a Ford-Fulkerson implementation that uses Shortest-Path to find the path p in line 3 of the Ford-Fulkerson algorithm. Each edge is given unit-weight and the algorithm will then pick the shortest path each time. It then has a running-time of $O(VE^2)$.

Lemma 26.7 If Edmonds-Karp 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.

Proof-ish Intuitively, if we chose a path p from $s \to v$ that is a shortest path, and then assume there is a path p' from $s \to v$ which is shorter, we contradict our initial statement that p is a shortest path. This proves that the path length do not decrease. p' might have the same length as p or might be longer. $\square$

Theorem 26.8 If Edmonds-Karp 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 $O(VE)$.

Write down and understand the proof...

Fibonacci Heaps

Write something about mergable heaps

Structure

Operations

Make-Heap

Insert

Minimum

Extract-Min

Union

DecreaseKey

Delete