Feb 13, 2008

Lecture 12

Last time we talked about spanning trees. A subgraph T of a a graph G is a spanning tree of G if T is a tree and T contains all the vertices of G. Spanning trees are not unique: the graph could have more than one spanning tree.

There were two algorithms described for finding spanning trees — the depth-first search, and the breadth-first search. Both begin with an arbitrarily chosen root.

G has a spanning tree if and only if G is connected!

The breadth-first search algorithm gives the shortest path in G from the chosen root to a given vertex.

Please read examples 1, 2, 3, 4 in Section 3.2 (pp. 104-109) in the textbook. Examples 1 and 2 can also be found in Lecture 11 notes posted on Blackboard. Example 1 was also done in class.

Chapter 4: Network Algorithms

Suppose $G=(V,E)$ is a graph, and for each edge $e \in E$ we have a non-negative integer $k(e)$. The integer $k(e)$ is called the capacity of e, and usually represents the "length" or "cost" of an edge, or the "volume" of water/fluid that can "flow" through that edge.

A network is a triple $N = (V, E, k)$ where (V, E) is a graph (often a directed graph, but not always) and

$$k : E \longrightarrow \mathbb{Z}_{\geq 0}$$

a function, where $\mathbb{Z}_{\geq 0}$ is the set of non-negative integers.

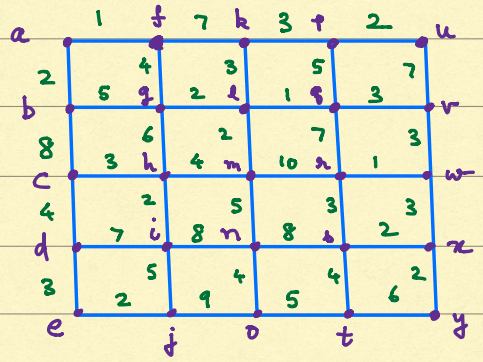
Definition (Minimum Spanning Tree): A minimum spanning tree in a network N is a spanning tree whose sum of edge capacities $k(e)$ is as small as possible.

Here are two algorithms for finding minimal spanning tree.

In what follows $N = (V, E, k)$ is a network and $n = \#$ of vertices in N.

- Kruskal's algorithm: We build T step by step as follows. In the initial step T is empty (T has no vertices, and hence no edges). At each stage add to T the lowest capacity edge which does not form a circuit with edges already in T. (In the case of a tie for the lowest capacity edge, pick any of the tied edges.). Repeat process till T has $n-1$ edges.
- Prim's Algorithm: Initially pick any edge with lowest capacity. At each stage add to T the edge with lowest capacity between a vertex in T and a vertex not in T (in case of a tie, pick any tied edge). Repeat the process till T has $n-1$ edges.

Example:

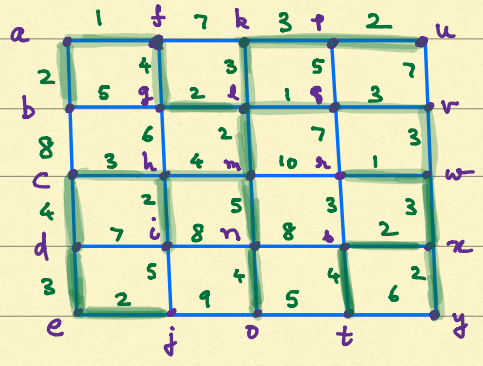


Consider the network N on the left. The vertices are a, b, c, d, ..., y and the capacities are either above the edge (for horizontal edges) or to the left of the edge (for vertical edges).

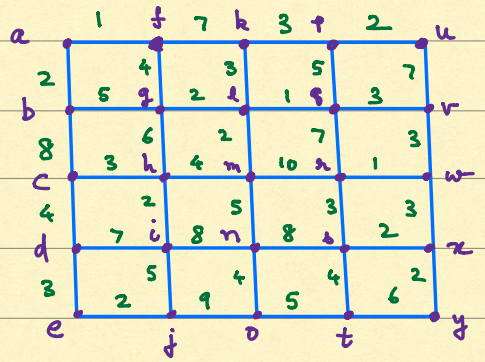
Both algorithms start with a lowest capacity edge. There are three edges with capacity 1:

- (a, f)
- (l, f)
- (r, w)

In Prim's algorithm, pick any of the above. Suppose we pick (a, f). Next add (a, b), then (f, g), then (g, l), then (l, m), ... The last but one addition would be either (m, n) or (o, t) both of capacity 5, and we can pick either. The final one would be (n, o)



The shaded portion is a minimum spanning tree for N.



For Kruskal's algorithm we would have to (a,f), (h,q), and (r,w) in the initial step (all three!!). Next we would add edges of capacity 2:

(a,b), (e,j), (g,l), (h,i), (l,m), (p,u), (s,x), (x,y).

At the next step we have to make a choice. We add almost all edges of capacity 3:

(c,h), (d,o), (k,l), (k,p), (q,v), (r,s), (v,w),

but not (w,x) unless (r,s) were omitted in our above choice (if both were present we would get a circuit with (r,w), (w,x), (s,x), (r,s)). Next we would add all edges of capacity 4, and finally either (m,n) or (o,t) to obtain the same minimum spanning tree(s) produced by Prim's algorithm.

The proofs that these algorithms give minimum spanning trees is omitted. Read pp. 132-133 of textbook for a proof that Prim's algorithm works.

Definition (Directed network): A directed network N is a triple (V, E, k) with (V, E) a directed graph and $k: E \to \mathbb{Z}_{\geq 0}$ a function. ($\mathbb{Z}_{\geq 0}$ = set of non-negative integers and $k(e)$ = capacity of e.)

4.3 Network Flows: Suppose $N = (V, E, k)$ is a directed network. Suppose a, z are vertices. An **a-z flow** on N is a function $f : E \longrightarrow \mathbb{Z}_{\geq 0}$ (set of non-neg integers) satisfying the following conditions (a), (b), and (c):

(a) $0 \leq f(e) \leq k(e) \quad \forall e \in E$;

(b) for $x \neq a$ or z, $\sum_{e \in In(x)} f(e) = \sum_{e \in Out(x)} f(e)$,

where $In(x)$ and $Out(x)$ denote the sets of edges directed into and out from vertex x;

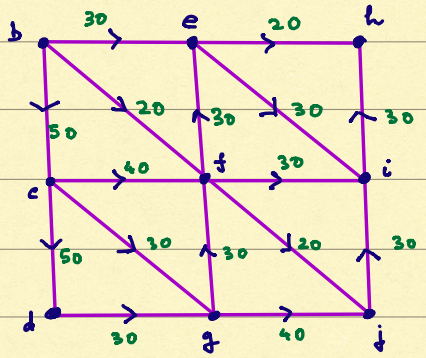
(c) $f(e) = 0$ if $e \in In(a)$ or $e \in Out(z)$.

The vertex a is called the **source** of the a-z flow and the vertex z is called the **sink** of the a-z flow.

One can visualise $f(e)$ is the amount of fluid flowing along e. In this way of thinking $k(e)$ is the maximum fluid that can flow along e, and hence the term "capacity". Condition (b) says that if x is a vertex which is not a or z, then fluid can neither be created or destroyed at x or accumulate at x. The last condition, namely (c), says that the flow goes from a to z, and never the other way around.

One can have a more general model in which the network N allows multiple sources (not just a) and multiple sinks (not just z), but in such a case we can always recast the problem into one with a single-source and single-sink network. The following example illustrates this technique.

Example 1: (Flow networks with supplies and demands)

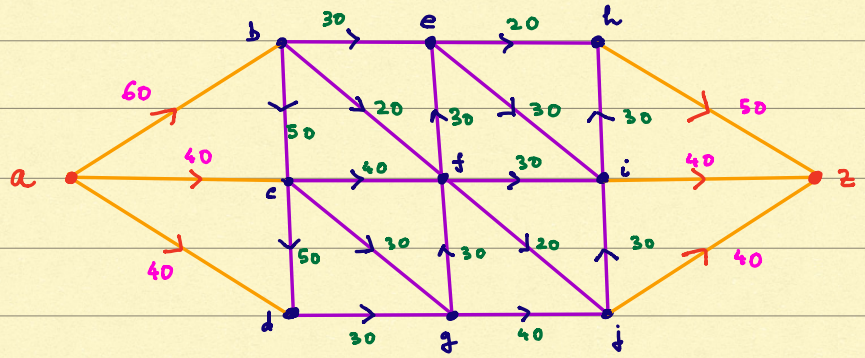


Suppose N is the directed network on the left. The numbers are the capacities of the corresponding directed edges. Thus $k(\vec{f,i}) = 30$, $k(\vec{c,d}) = 50$, $k(\vec{g,j}) = 40$, etc.

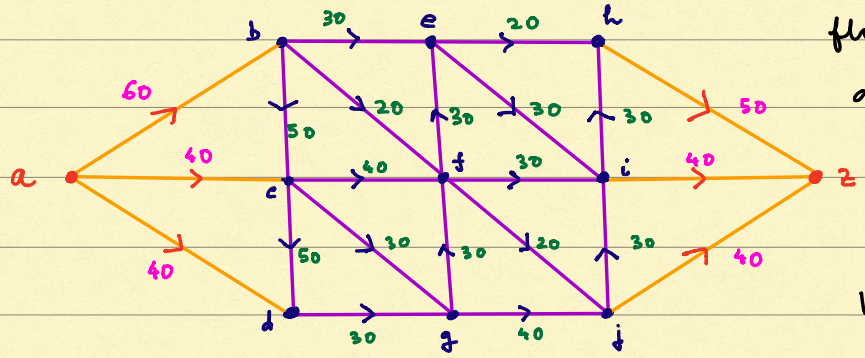
Now suppose we allow multiple sources and sinks. Suppose specifically that vertices b, c, d are sources, h, i, j are sinks, such that: b can supply up to 60 units of flow, c and d up to 40 units each, and h, i, and j have flow demands of 50, 40, and 40 units, respectively.

Here is how one can expand the network so that we have a single source (instead of b, c, and d) and a single sink (instead of h, i, and j).

Introduce two new vertices: a and z, and six new directed edges namely $(\vec{a,b})$, $(\vec{a,c})$, $(\vec{a,d})$, $(\vec{h,z})$, $(\vec{i,z})$, $(\vec{j,z})$. Give them capacities as shown in the picture:



In the picture a is a source that can supply 140 units of flow and z is a sink with a demand for up to 130 units of flow. None of the other vertices are sources are sinks.

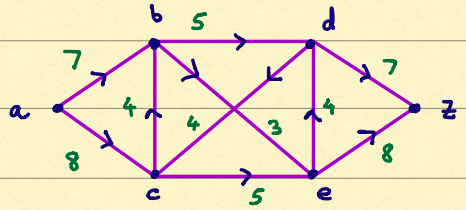


A flow satisfying the original demands is equivalent to a flow in the new network.

Important Theorems about Network Flows

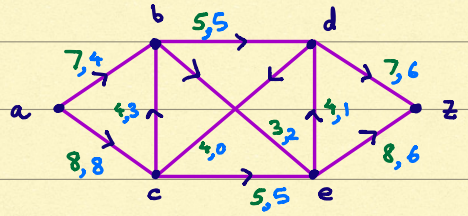
The important theorems for this course are the Max flow - Min cut Theorem and Hall's Marriage Theorem. We won't be proving them in today's lecture.

Here is a series of discussions and results (often without proofs) leading to these two theorems. It is useful to have concrete examples in front of us as we discuss. The picture below is one.



Here is a flow for the above directed network. The flow in the picture below is given by the second number along each

edge.



In this picture, for example $k(\overrightarrow{d,c}) = 4$ but $f(\overrightarrow{d,c}) = 0$. Similarly $k(\overrightarrow{c,b}) = 4$ but $f(\overrightarrow{c,b}) = 3$, etc. One can check conditions (a), (b), and (c) are true for the flow f. For example, if we look at vertex b, the in-flow is $f(\overrightarrow{a,b}) + f(\overrightarrow{c,b}) = 4+3 = 7$, whereas the out-flow at b is $f(\overrightarrow{b,e}) + f(\overrightarrow{b,d}) = 2+5 = 7$ verifying condition (b) at vertex b.

The first important theorem is:

Theorem 1: For any a-z flow in a network N, the flow out of a equals the flow into z. //

The theorem is intuitively obvious. The idea is that there is no net accumulation of fluid in any vertex not equal to a or z by condition (b), and no fluid flows into a, and no fluid flows out of z. In our example above, the flow out of a is $f(\overrightarrow{a,b}) + f(\overrightarrow{a,c}) = 4+8 = 12$, whereas the flow into z is $f(\overrightarrow{d,z}) + f(\overrightarrow{e,z}) = 6+6 = 12$. See the end of these notes for a proof of the Theorem.

To proceed further we need the concept of a cut.

Definition (of a cut): Let $N = (V, E, k)$ be a directed network, P a subset of E, and $\bar{P}$ its complement in E. The set

$$(P,\bar{P}) = \left\{ e \in E \;\middle|\; \begin{array}{l} e \text{ is an edge from a vertex in } P \\ \text{to a vertex in } \bar{P} \end{array} \right\}$$

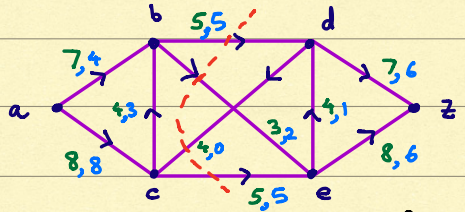
is called a cut. We call $(P,\bar{P})$ an a-z cut if $a \in P$ and $z \in \bar{P}$.

Important formula:

> (b') For each vertex subset P not containing a or z,
>
> $$\sum_{e \in (\bar{P},P)} f(e) = \sum_{e \in (P,\bar{P})} f(e)$$

The idea is again simple. The ultimate source of all flow is a, and in the end everything flows into z. A proof is given at the end of these notes (see also p. 136 of the textbook).

Consider again our example:



The dotted line shows a cut.

Let $P = \{b, c\}$. Then $\bar{P} = \{a, d, e, z\}$. The dotted line indicates the cut $(P,\bar{P})$. The edges in $(P,\bar{P})$ are the ones which cross the dotted line exactly once and from left to right.

Definition (The value of an a-z flow): Let f be an a-z flow in a directed network $N = (V, E, k)$. The value of the a-z flow f, denoted $|f|$, is the sum of the flows in edges coming out of a.

Note that by Theorem 1, $|f|$ is also equal to the sum of the flows in edges coming into z.

Definition (the capacity of a cut): Let $(P, \bar{P})$ be a cut in a directed network $N = (V, E, k)$. The **capacity** $k(P, \bar{P})$ of $(P, \bar{P})$ is defined to be

$$k(P, \bar{P}) = \sum_{e \in (P, \bar{P})} k(e).$$

Theorem 2: For any a-z flow f and any a-z cut $(P, \bar{P})$ in a directed network N, $|f| \leq k(P, \bar{P})$.

Proof at the end of these notes.

In this theorem it is extremely important that $(P, \bar{P})$ be an a-z cut, i.e., $a \in P$, $z \in \bar{P}$.

Corollary 2a: For any a-z flow f and any a-z cut $(P, \bar{P})$ in a directed network $N = (V, E, k)$, $|f| = k(P, \bar{P})$ if and only if

(i) For each edge $e \in (\bar{P}, P)$, $f(e) = 0$

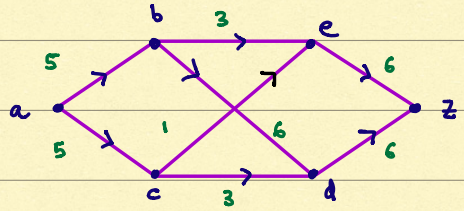
(ii) For each edge $e \in (P, \bar{P})$, $f(e) = k(e)$.

Further, when $|f| = k(P, \bar{P})$, f is a maximum flow and $(P, \bar{P})$ is an a-z cut of minimum capacity.

Proof at end of these notes.

It is clear from Thm 2 that if $|f| = k(P, \bar{P})$ then f is a maximum flow and $k(P, \bar{P})$ has minimum capacity. As it turns out, the converse is also true (Max flow - Min cut Thm to be stated later).

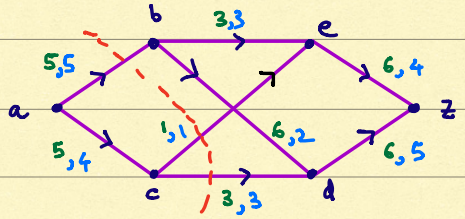
Example: Consider the following directed network.



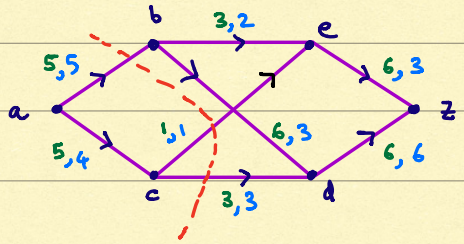
We will see later how to find a minimum cut (equivalently a maximum flow).

It turns out that $(P, \bar{P})$ with $P = \{a, c\}$ is a minimal cut.

A maximal flow is shown below. Note that $(P, \bar{P})$ consists of the three edges $(a,\vec{b})$, $(c,\vec{e})$, $(c,\vec{d})$ and $k(P,\bar{P}) = 5+1+3 = 9$.



The value of f on the edges of $(P, \bar{P})$ has to equal the capacities there by Corollary 2a. Thus $f(\vec{a,b}) = k(\vec{a,b}) = 5$, $f(\vec{c,e}) = k(\vec{c,e}) = 1$, $f(\vec{c,d}) = k(\vec{c,d}) = 3$, and this forces $f(\vec{a,c}) = 4$. Here is another maximum flow:



In both cases $|f| = 9 = k(P, \bar{P})$.

Max Flow - Min Cut Theorem: In any directed flow network, the value of a maximum a-z flow is equal to the capacity of a minimum a-z cut.

Some proofs:

First let us prove formula (b') for a flow f in a a-z network.

> (b') For each vertex subset P not containing a or z,
>
> $$\sum_{e \in (\bar{P}, P)} f(e) = \sum_{e \in (P, \bar{P})} f(e)$$

We have for any $x \neq a$ or z, $\sum_{e \in In(x)} f(e) = \sum_{e \in Out(x)} f(e)$.

Since P does not contain a or z, we get

$$\sum_{x \in P} \sum_{e \in In(x)} f(e) = \sum_{x \in P} \sum_{e \in Out(x)} f(e) \qquad \text{——— } (*).$$

Let S be the set of edges from vertices in $\bar{P}$ to vertices in P.

Then,

Left side of $(*) = \sum_{e \in S} f(e) + \sum_{e \in (\bar{P}, P)} f(e)$ ——— (1)

whereas

Right side of $(*) = \sum_{e \in S} f(e) + \sum_{e \in (\bar{P}, P)} f(e)$. ——— (2)

From (1), (2) and (*), after cancelling $\sum_{e \in S} f(e)$ from both sides, we get the formula (b').

Proof of Theorem 1: We have to show that for any a-z flow in a network N, the flow out of a equals the flow into z. In (b') take $P = V - \{a, z\}$. Then $\bar{P} = \{a, z\}$. Theorem 1 follows

from (b') and condition (c) for a flow.

Proof of Theorem 2:

Let P be a vertex subset such that $(P, \bar{P})$ is an a-z cut. This means $a \in P$ and $z \in \bar{P}$.

Let $Q = P - \{a\}$.

Then $\bar{Q} = \bar{P} \cup \{a\}$.

Now Q is a vertex subset not containing a or z, and so formula (b') applies and we have

$$\sum_{e \in (\bar{Q}, Q)} f(e) = \sum_{e \in (Q, \bar{Q})} f(e) \qquad \text{——— (I)}$$

Let R be the set of edges starting at a and ending at a vertex in $\bar{P}$. Then, as $f(e) = 0$ for $e \in In(a)$

$$\left.\begin{aligned} \sum_{e \in (P, \bar{P})} f(e) &= \sum_{e \in R} f(e) + \sum_{e \in (Q, \bar{Q})} f(e) \\ &= \sum_{e \in R} f(e) + \sum_{e \in (\bar{Q}, Q)} f(e) \end{aligned}\right\} \text{(II)}$$

Let R' be the set of edges starting at a and ending at a vertex in P. Then

$$\sum_{e \in (\bar{Q}, Q)} f(e) = \sum_{e \in R'} f(e) + \sum_{e \in (\bar{P}, P)} f(e) \qquad \text{—— (III)}$$

Substituting (III) in (II) we get:

$$\left.\begin{aligned} \sum_{e \in (P, \bar{P})} f(e) &= \sum_{e \in R} f(e) + \sum_{e \in R'} f(e) + \sum_{e \in (\bar{P}, P)} f(e) \\ &= \sum_{e \in Out(a)} f(e) + \sum_{e \in (\bar{P}, P)} f(e) = |f| + \sum_{e \in (\bar{P}, P)} f(e). \end{aligned}\right\} \text{(IV)}$$

Since $f(e) \le k(e) \ \forall e$, the above shows that

$$k(P,\bar{P}) = \sum_{e \in (P,\bar{P})} k(e) \ge \sum_{e \in (P,\bar{P})} f(e) = |f| + \sum_{e \in (\bar{P},P)} f(e). \qquad \text{(A)}$$

We thus have

$$k(P,\bar{P}) \ge |f| + \sum_{e \in (\bar{P},P)} f(e) \qquad \text{(B)}$$

In particular $k(P,\bar{P}) \ge |f|$. This proves Theorem 2.

Proof of Corollary 2a:

If $f(e) = k(e)$ for all $e \in (P,\bar{P})$, then by inequality (A), we have $k(P,\bar{P}) = |f| + \sum_{e \in (\bar{P},P)} f(e)$, and if $f(e) = 0$ for all $e \in (P,\bar{P})$, then $k(P,\bar{P}) = |f|$.

Conversely if $k(P,\bar{P}) = |f|$, then by (B) we must have $\sum_{e \in (\bar{P},P)} f(e) = 0$ which means $f(e) = 0$ for all $e \in (\bar{P},P)$.

Since by (IV)

$$\sum_{e \in (P,\bar{P})} f(e) = |f| + \sum_{e \in (\bar{P},P)} f(e)$$

and the second term on the right is zero, we get $|f| = \sum_{e \in (P,\bar{P})} f(e)$. The condition that $k(P,\bar{P}) = |f|$ therefore translates to

$$\sum_{e \in (P,\bar{P})} k(e) = \sum_{e \in (P,\bar{P})} f(e).$$

Since $k(e) \ge f(e) \ \forall e \in E$, this can only happen if $k(e) = f(e)$ for $e \in (P,\bar{P})$.

Q.e.d.