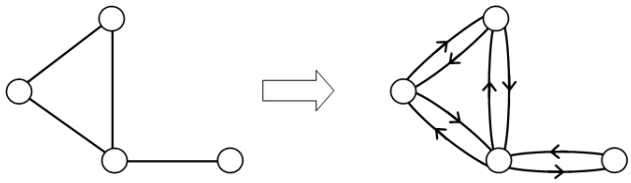
Chapter 5. Section 1

Digraphs and Networks

Our aim is to investigate how much traffic or **flow** can be pushed through a network (roads, gas pipes, Internet, etc). We are going to adopt the terminology **network = digraph with distinguish source vertex and target vertex**, so that our traffic is one-way on each edge and flows from source to target.

We will write N for networks to distinguish them, and refer to the edge set as $A(N)$. Our source will be called s and our target t. So a network is specified as $N = (V(N), A(N), s, t)$.

Note: we can still deal with undirected graphs by turning 1 two-way edge into 2 one-way edges, as illustrated below:



Our method for maximising flow will be based on two ideas: **maximal subtrees** and **directed edge cuts**.

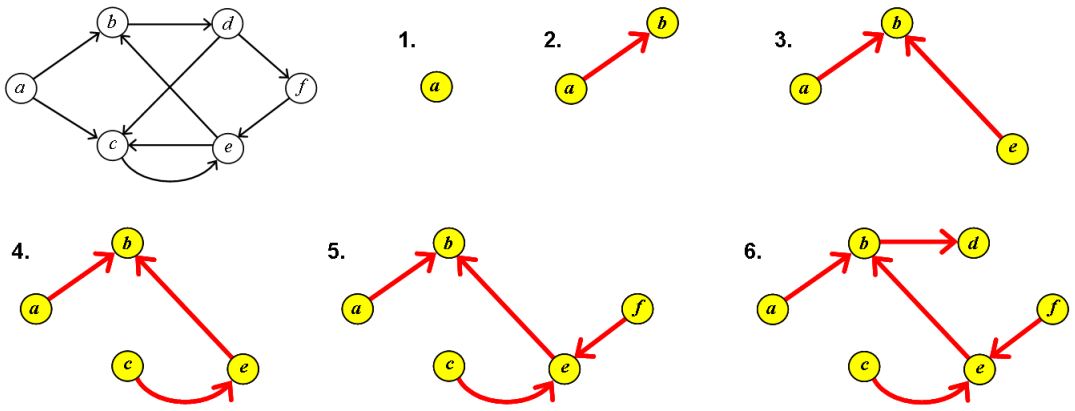
Maximal subtrees in networks

We build trees rooted at s, using an algorithm adapted from Chapter 2, Section 2. It ignores t, what is more, it doesn't care about edge direction!

Algorithm **Maximal_Network_Subtree**
Input: network $N = (V(N), A(N), s, t)$
Output: subtree T *of* N *containing* s

```
T := ({s}, ∅)     # T is a tree with one vertex s and no edges
while there is an edge xy leaving T or entering T do
    if xy leaves T then V(T) := V(T) ∪ {y} else V(T) := V(T) ∪ {x}
    A(T) := A(T) ∪ {xy}
return(T)
end
```

Here is an example of in action:



Note: (1) Maximal_Network_Subtree does **not** produce in-trees (directed towards s) or out-trees (directed away from s); an edge xy is added to T regardless of whether it enters or leaves the T. Note, thought, that xy is **not** the same edge as yx in a digraph!

(2) There is an efficiency trade off: do we store all *incoming* as well as outgoing edges for every vertex? If not then how does our **while** loop test for edges entering T? We can store more data or we can do more processing — it is our choice.

Directed edge cuts in networks

Given $N = (V(N), A(N), s, t)$ we want to know how easy it is to separate s from t.

A set of edges $X \subseteq A(N)$ whose deletion leaves no directed path from s to t is called a **directed edge** st**-cut**, or just an st**-cut**.

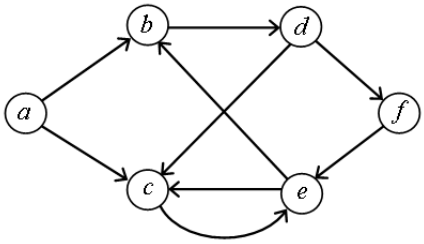
Notation: if v is a vertex of N we write $d^-(v)$ to denote the **indegree** of v, that is, the number of edges arriving at v; we write $d^+(v)$ to denote the **outdegree** of v, that is, the number of edges arriving at v. If $U \subseteq V(N)$ is a subset of network vertices then $A^-(U)$ denotes the set of all edges entering U; and $A^+(U)$ denotes the set of all edges leaving U. Note the $-$ for 'incoming' and $+$ for 'outgoing' consistently with $d^-(x)$ and $d^+(x)$.

Examples: (1) We could rewrite the **while** test and the **if** test in Maximal_Network_Subtree as, respectively,

$$\textbf{while} A^-(T) \cup A^+(T) \neq \emptyset, \quad \text{and} \quad \textbf{if} xy \in A^+(T).$$

(2) If $U \subseteq V(N)$ contains s and does not contain t then $A^+(U)$ is an st-cut; if U contains t and does not contain s then $A^-(U)$ is an st-cut.

Some st-cuts for different choices of s and t in the network on page 1 are shown in the table below (the network is redrawn for ease of reference). Ignore the last column for the moment...



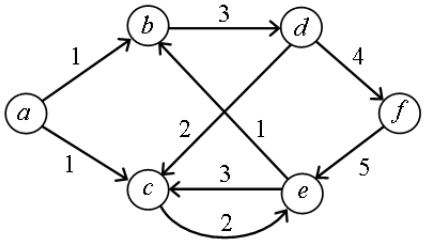
	source, target	cut	minimum capacity cut
1.	$s = a, t = f$	$\{bd\}$ or $\{df\}$	$\{ab, ac\}$, cost = 2
2.	$s = f, t = a$	$\emptyset$	$\emptyset$, cost = 0
3.	$s = a, t = a$	none possible	none possible
4.	$s = c, t = e$	$\{ce\}$ (unique)	$\{ce\}$, cost = 2
5.	$s = f, t = c$	$\{ec, bd\}$ or $\{fe\}$	$\{ec, eb\}$, cost = 4

Notice the usual distinction of **minimal** and **minimum** applies: in 5. $\{ec, bd\}$ is a minimal cut because $\{ec\}$ and $\{bd\}$ are not cuts — you cannot 'reduce' $\{ec, bd\}$; but $\{fe\}$ is a minimum size cut (which is also minimal of course).

If we add weights we get a different meaning for 'minimum'.

In networks we refer to edge weights as **capacities**indexedge!capacity since their role is to limit the amount of flow (traffic) through edges. Although we could imagine a use for negative capacity we will assume the capacity of an edge e is a nonnegative real number and denote it by $c(e)$. Think of c as a function $c : A(N) \to \mathbb{R}^{\geq 0}$.

In the drawing below, our example network has been given capacities:



If you look at the table above right you will see that minimum-capacity cuts may have more elements than minimum-size cuts.