Solutions to question set for Chapter 6

1. Prove the following lemma, which is an important detail in a complete proof of correctness of Ford-Fulkerson:

   Lemma Suppose f is an st-flow in a network N and P is an f-unsaturated st-path in N. Suppose that flow g is constructed from f by increasing the value by the slack value $\mathrm{sl}((P))$ on forward arcs of P and decreasing the value by $\mathrm{sl}((P))$ on backward arcs of P. Then g is an st-flow with value $|g| = |f| + \mathrm{sl}((P))$.

   SOLUTION: We must prove

   (a) g is an xy-flow;

   (b) $|g| = |f| + s(P)$.

   We will assume that x has indegree zero and y has outdegre zero.

   (a) We must show that, for any vertex $v \neq x, y$, $g^+(v) = g^-(v)$, where $g^+(v)$ = total g on edges out of v and $g^-(v)$ = total g on edges into v. There are four possible ways that P can pass v:

   i. P does not pass v. Then g is the same as f on edges incident with v so $g^+(v) = f^+(v) = f^-(v) = g^-(v)$;

   ii. P enters v via edge uv and leaves v via edge vu'. Then uv and vu' are both forward edges on P so $g^+(v) = f^+(v) + s(P) = f^-(v) + s(P) = g^-(v)$

   iii. P enters v via edge uv and leaves v via edge $u'v$. Then uv is a forward edge on P and $u'v$ is a backward edge, so $g^+(v) = f^+(v) = f^-(v) + s(P) - s(P) = g^-(v)$

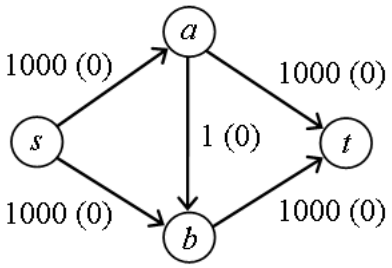
   iv. P enters v via edge vu and leaves v via edge vu'. Then vu is a backward edge on P and vu' is a forward edge, so $g^+(v) = f^+(v) - s(P) + s(P) = f^-(v) = g^-(v)$.

   In each case the flow condition on f ensures the flow condition on g, so g is a flow.

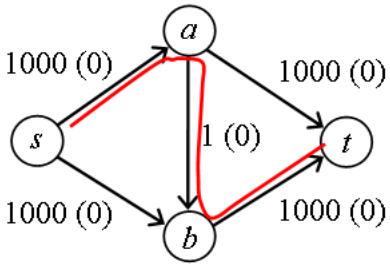
   (b) The value of the flow g is the total on edges leaving x. Now only one edge of P can leave x and g increases f by $s(P)$ on this single edge, so $|g| = |f| + s(P)$ as required.

2. The network shown below right is from our recommended further reading: the book by Papadimitiou and Steiglitz.

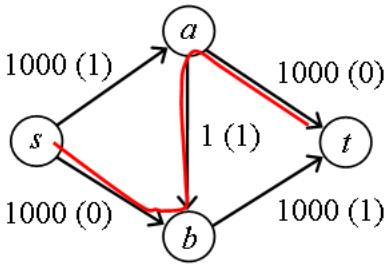
   Starting from the initial zero-value flow (shown in brackets), run the Ford-Fulkerson algorithm for two iterations, using the **longest** possible flow augmenting path in each iteration. Hence predict the number of iterations which will be required, in the **worst case**, to maximise the flow. What conclusion to you draw about the complexity of the Ford-Fulkerson algorithm?



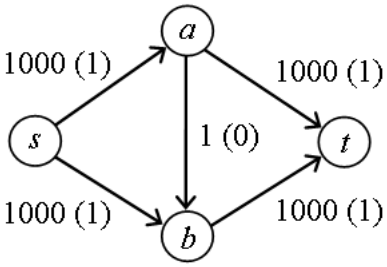
   SOLUTION: the idea of this question is to investigate the *worst case scenario*: in image (a) below, an f-unsaturated tree is given which 'chooses' the middle edge of capacity 1. The s-t path in this tree (which is the whole tree) has slack value 1 and so increasing the flow along this path increases the flow value by 1.



(a)



(b)



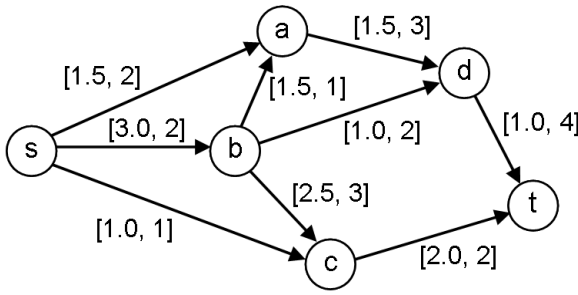
(c)

   In image (b) a 2nd iteration of Ford-Fulkerson finds a new f-unsaturated tree; again the worst case is identified: the edge of capacity 1 is chosen as a 'backward' edge. Again, this gives a path of slack value 1 which allows an increase in flow value of only 1. This new flow is shown in image (c).

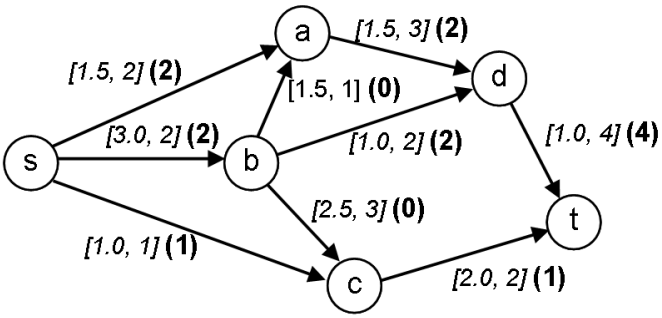
   We see that, in the worst case, the running time of Ford-Fulkerson is determined not by the size of the network but by the size of the capacity values. (A few small modifications to the design of the algorithm remove this problem however).

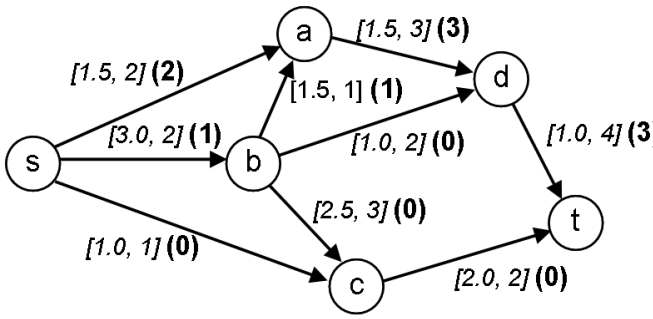
3. In the network below right each edge e is labelled with a pair of weights $[u(e), c(e)]$. The first entry is the *cost* of using the edge, the second entry is the *capacity* of the edge.

Use trial and error to find an st-flow f which maximises $|f|$, subject to the edge capacities, while minimising cost, where the cost of a flow is the sum of the costs of the edges with non-zero flow, i.e. cost= $\sum_{e.f(e)\neq 0} u(e)$. Apply the Ford-Fulkerson algorithm, starting with a zero-flow, to find a maximum st-flow. Does this minimise cost? If so, do you think Ford-Fulkerson is guaranteed to *always* minimise cost as well as maximising flow?



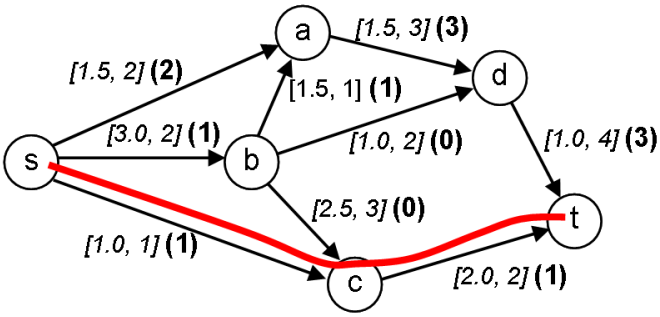
SOLUTION: A maximum flow is shown below-left. The flow values are given, as usual, in brackets (but printed in bold type to make them stand out). The flow value is 5 (the sum of the edge flows leaving s) and the flow has total cost 1.5 + 1.5 + 1.0 + 3.0 + 1.0 + 1.0 + 2.0 = 11.0, the total cost of the chosen edges.

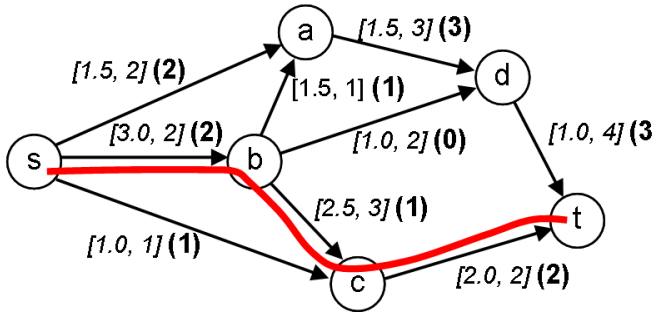




It is possible that Ford-Fulkerson will not chose this particular flow. Suppose we start with a zero-flow: every edge has zero flow on it. Two iterations of Ford-Fulkerson might produce the flow shown above-right: first 2 units of flow are sent along edges sa, ad, dt; then 1 unit of additional flow is sent along sb, ba, ad, dt.

Now we show two further increases to the flow: the first sends 1 unit along path sc, ct. The second sends a further 1 unit along sb, bc, ct. Again flow is now maximised. But this time the total cost of the edges chosen is 13.0





So we need a more sophisticated approach to maximise flow and minimise cost at the same time.

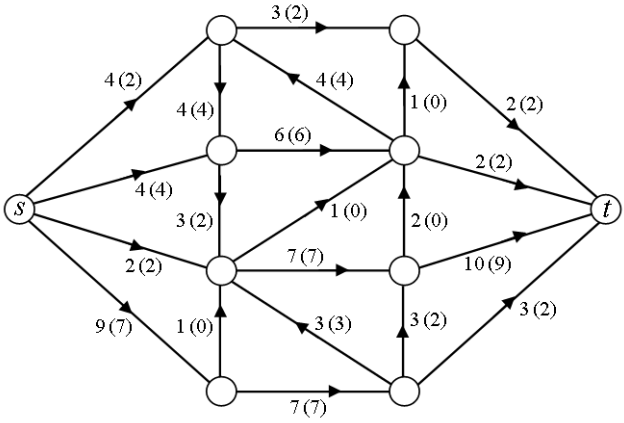
A more realistic definition of cost would charge on each edge *per unit of flow*. In this case the problem may be solved by linear programming and the original flow chosen on the previous page again solves the problem (with a total cost now of $1.5 \times 2 + 1.5 \times 2 + 1.0 \times 4 + 3.0 \times 2 + 1.0 \times 2 + 1.0 \times 1 + 2.0 \times 1 = 21.0$.

4. The network below right has a flow f from s to t with initial values of $f(e)$ given in brackets on each edge e, the unbracketed number being the capacity $c(e)$ of the edge. *There is a 1-page pdf file containing a large version of the network image available on the Chapter 6 web page: you may like to print this page and write your answers on in order to save you redrawing the network by hand when answering this question.*

   Run the Ford-Fulkerson Algorithm for **three** iterations on the network in order to find a maximum flow and a minimum capacity cut.

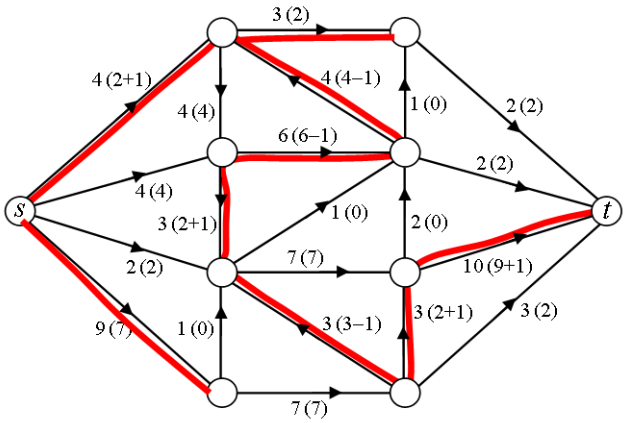
   You should construct a maximal f-unsaturated tree during each iteration and identify, if possible, a flow augmenting path in each tree. This path should then be used to increase the value of the flow; while if no path can be found then, instead, a minimum capacity cut should be identified.

   You may present your work in any way you wish provided that you make clear how the Ford-Fulkerson Algorithm is working during each iteration. *Some further advice on this is given on the accompanying 1-page image document.*

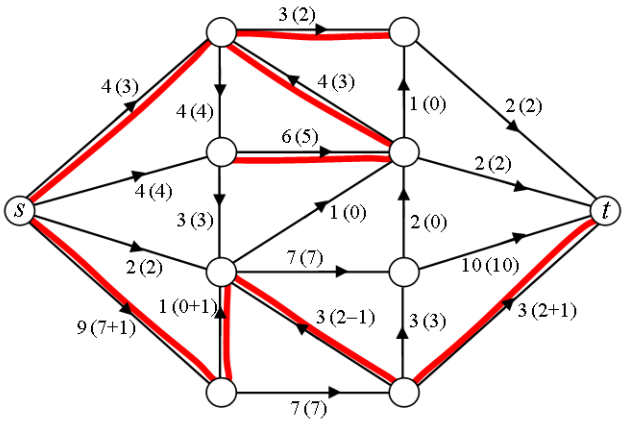


SOLUTION:

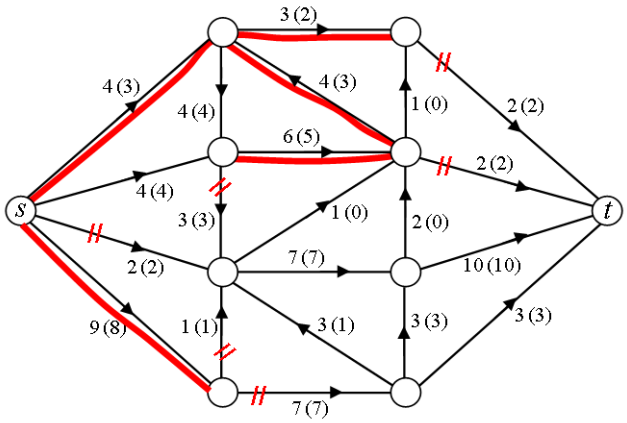
Iteration 1



Iteration 2



Iteration 3



Notes: Maximal f-unsaturated tree shown in each iteration as bold (red) edges. Augmentation of flow along a flow augmenting st-path is shown by increases/decreases in (bracketed) flow values.

The trees are not unique: for example, the flow-augmenting path in Iteration 2 may be constructed using a slightly different tree during Iteration 1. Any valid tree and path is acceptable.

The tree in Iteration 3 is maximal but does not include vertex t. The edges leading out of this tree are indicated thus: //. The sum of these edge capacities is 17 which is the same as the net flow out of s and the net flow into t.