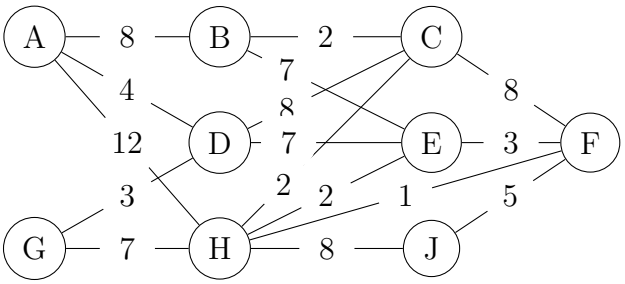
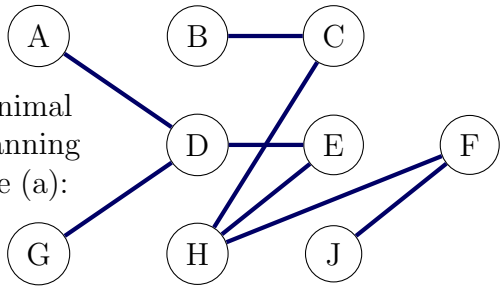
With all question sets, but *especially* with this one, you will gain much more by attempting the questions *before* looking at the solutions.

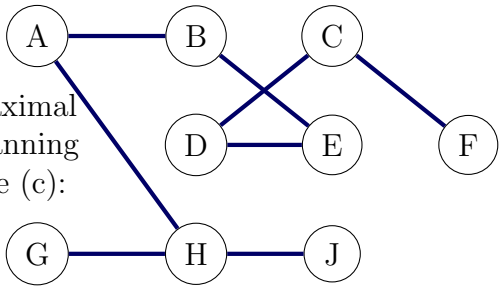
1. Let G be the weighted graph at right:
(a) Find a minimal spanning tree for G.
(b) How many minimal spanning trees does G have? Justify your answer.
(c) Find a maximal spanning tree for G.
(d) How many maximal spanning trees does G have? Justify your anser.



Minimal spanning tree (a):



Maximal spanning tree (c):

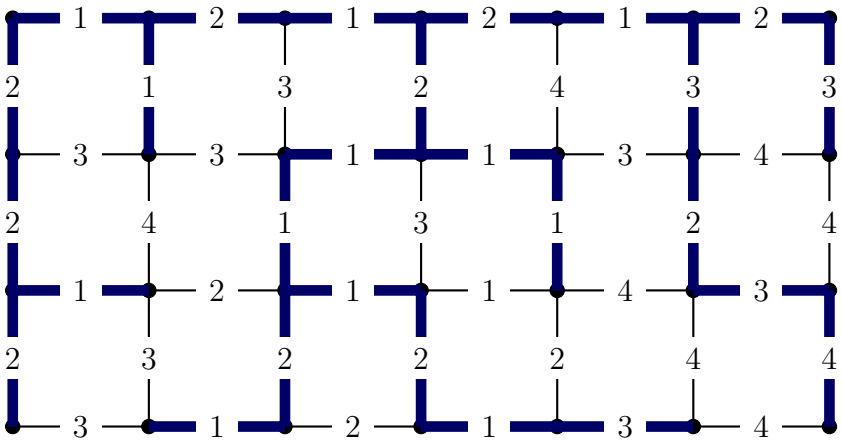


(b) There are 2 minimal spanning trees. Using Kruskal's algorithm, the only opportunity for alternative trees is the choice of final edge. It needs to be chosen from the three edges of weight 7, but one of these, BE, would create a circuit.

(d) The above maximal spanning tree is unique. Using Kruskal's algorithm, all edges of weight down to, and including, 7, can be used. Since there are exactly $8 = 9-1$ of these, no alternative tree can be found.

2. Use Kruskal's algorithm to find a minimal spanning tree for the network below.
Of course there are are lots of minimal spanning trees. To get the same answer as me (so you can verify your answer) proceed as follows, in all steps working from left to right:
1. Mark all edges of weight 1 in the top line.
2. Mark all vertical edges of weight 1 between the top line and line 2 below.
3. Mark each edge of weight 1 in line 2 unless including the edge creates a circuit.
4. Continue like this, working across and down, until you reach the bottom right corner.
5. Now repeat all the above for weight 2, then again for weight 3 and weight 4.

At all times refrain from marking an edge if that edge would create a circuit.



3. A 'ring circuit' is to be installed with six power outlets $A, \ldots, F$. (As its name implies, a ring circuit with 6 outlets connects each outlet to two 'neighbours', to produce a circuit of 6 links.) The builder supplies the electrician with cost estimates for each of the 15 potential links in the circuit. These are shown in the chart at right.

A	17	24	26	12	10
B	7	12	14	20	
C	10	20	26		
D	17	24			
E	7				
F					

(a) Apply the nearest neighbour algorithm starting at A to find a low cost circuit.

$A\ {}_{10}\ F\ {}_{7}\ E\ {}_{14}\ B\ {}_{7}\ C\ {}_{10}\ D\ {}_{26}\ A$ Total cost = 10+7+14+7+10+26 = 74.

(b) Does the algorithm yield a cheaper circuit if you start at a different outlet?

Yes. Starting at B or E gives a tour of total cost 68.
(Starting at C or F is more expensive — 77. Starting at D yields 68; same cost as A.)

(c) Is any circuit produced by the algorithm the cheapest possible circuit?

Yes. The circuits starting at B and E are the cheapest possible, but this is not at all easy to confirm. With 6 vertices there are apparently $6! = 720$ circuits, but allowing for different starting points and directions of travel, only $5!/2 = 60$ are genuinely different. However, checking the total cost of 60 circuits is a big task. At present there is no known way to quickly find the best tour for any TSP (Travelling Salesperson Problem).

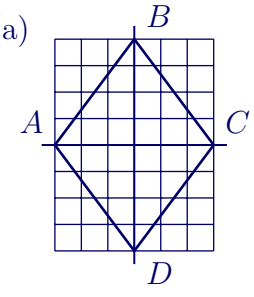
4. The nearest neighbour algorithm for TSP can fail in just about the simplest of situations, including when the weights are physical distances. As a demonstration, apply the algorithm to two complete graphs K_4, where the edge weights are the distances between vertices when placed at points with coordinates as follows:

(a) $A(-3,0)$, $B(0,4)$, $C(3,0)$, $D(0,-4)$. ($|AB| = 5$ by Pythagoras.)
(b) $A(-5,0)$, $B(0,12)$, $C(5,0)$, $D(0,-12)$. ($|AB| = 13$ by Pythagoras.)

For each weighted graph answer the following:

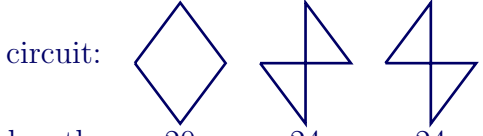
(i) What is the tour length produced by the algorithm starting from A?
(ii) What is the tour length produced by the algorithm starting from B?
(iii) What is the shortest *possible* tour?

(a)

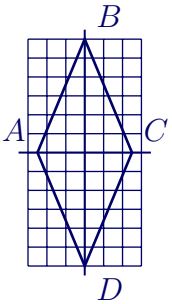


A	5	6	5
B	5	8	
C	5		
D			

(i) $ABCDA$ (or $ADCBA$); length 20.
(ii) $BCDAB$ (or $BADCB$); length 20.
(iii) Only 3 genuinely different circuits:

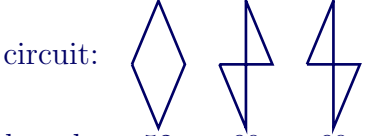


(b)



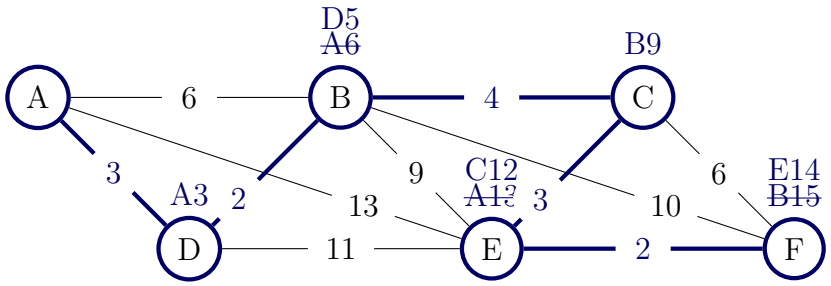
A	13	10	13
B	13	24	
C	13		
D			

(i) $ACBDA$ (or $ACDBA$); length 60.
(ii) $BACDB$ (or $BCADB$); length 60.
(iii) Only 3 genuinely different circuits:



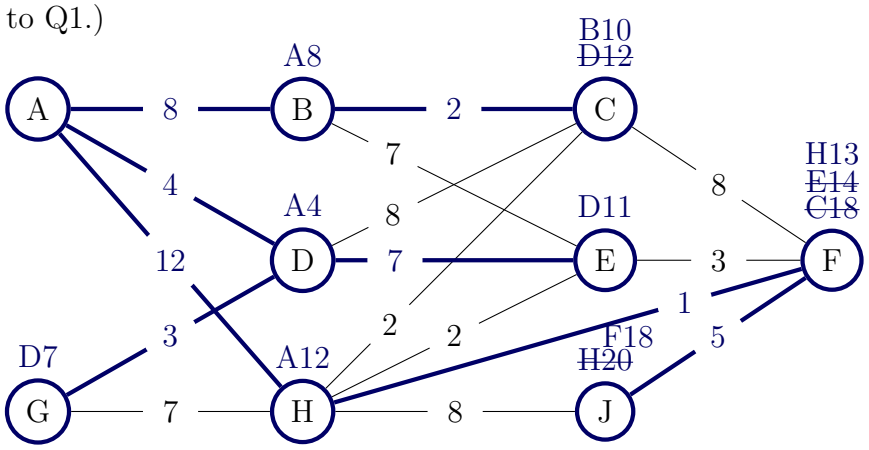
In the first case the algorithm always finds the shortest circuit, in the second, never.

5. Use Dijkstra's algorithm to find the shortest path from A to F in the network below. Show all, and only, annotations generated by the algorithm. Write annotations above vertices. Put a line through any annotation that needs updating and write the new annotation above it. Using heavy lines, mark out the spanning tree generated by the algorithm. This tree is a minimal spanning tree for the graph; is that inevitable?



NB: For Dijkstra's algorithm, the annotations must be **exactly** as above.
It is **not** inevitable that the spanning tree obtained by Dijkstra's algorithm is a minimal spanning tree. See Q6 below for example.

6. Use Dijkstra's algorithm to find the shortest path from A to J in the network of Q1, repeated below. Show all, and only, annotations generated by the algorithm. Write annotations above vertices. Put a line through any annotation that needs updating and write the new annotation above it. Using heavy lines, mark out the spanning tree generated by the algorithm. Is this tree a minimal spanning tree for the graph? (Refer to your answer to Q1.)



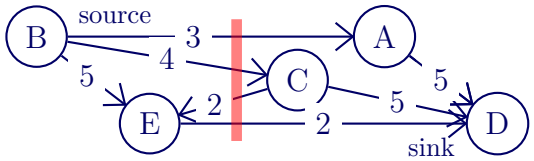
Even without referring to the answer to Q1, it is clear that the above tree is not a minimal spanning tree. For example AH could be replaced by CF to obtain a spanning tree of 4 units less total weight.

7. The matrix below shows the capacities for a transport network with vertices A,B,C,D,E.

$$\begin{bmatrix} 0 & 0 & 0 & 5 & 0 \\ 3 & 0 & 4 & 0 & 5 \\ 0 & 0 & 0 & 5 & 2 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix}$$

(a) Draw the network. Identify source and sink.

(b) Specify a partition of the vertices that creates a minimum cut.



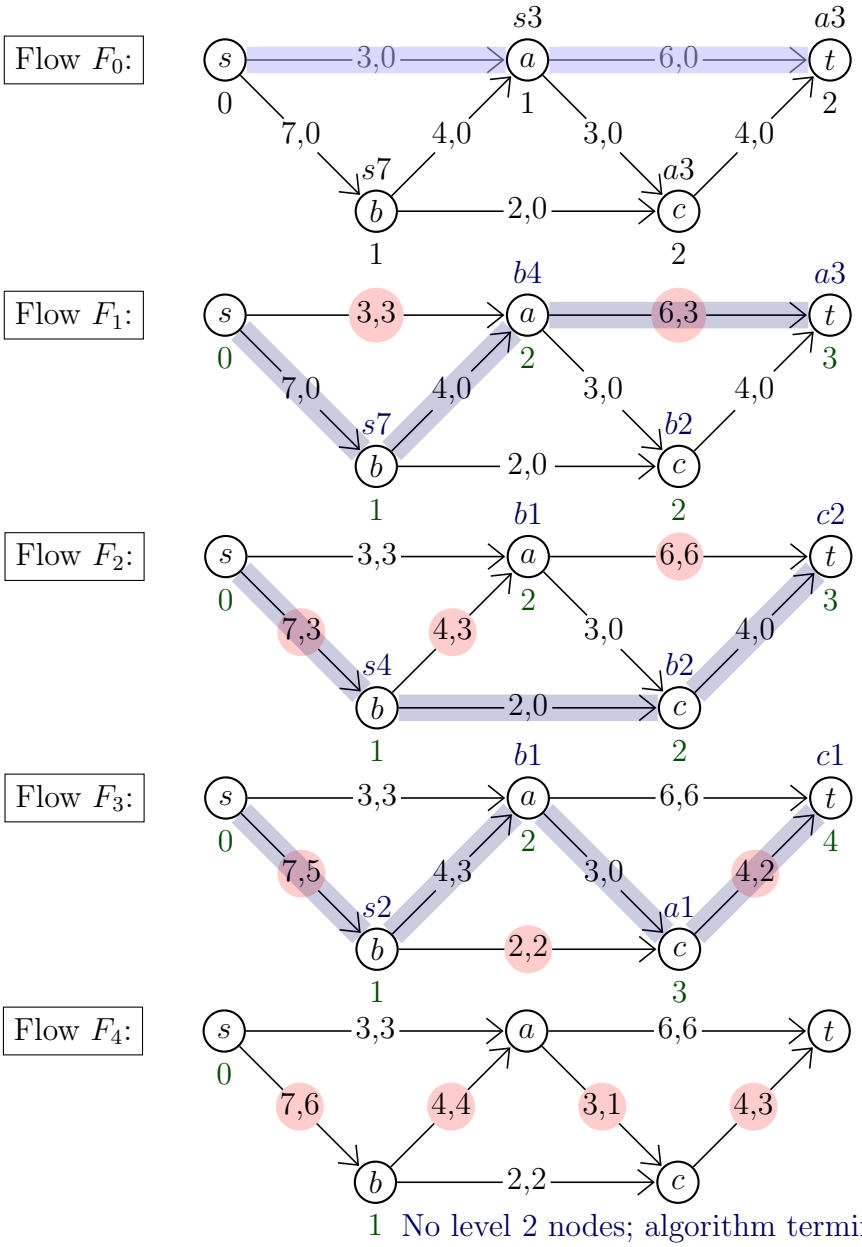
Partition = {{B,E},{A,C,D}}
(min cut value = ~~3+4+2~~ = 9)

(c) Find a maximum flow that proves your cut to be minimum. Specify the flow by a flow matrix, using the same ordering of the vertices as used in the capacities matrix. There is no need to use the labelling algorithm; find the flow 'by eye'.

Use full capacity of cut edges. Conserve flow at nodes. One solution:

$$\begin{bmatrix} 0 & 0 & 0 & 3 & 0 \\ 3 & 0 & 4 & 0 & 2 \\ 0 & 0 & 0 & 4 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \end{bmatrix}$$

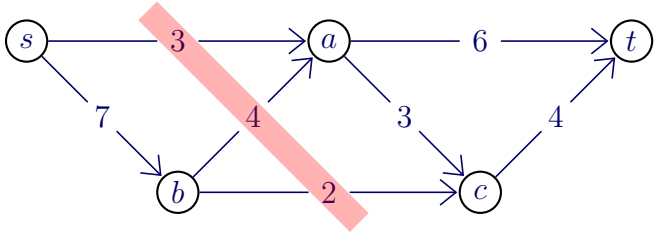
8. Demonstrate the application of the vertex labelling algorithm by applying it to the transport network below. Flow F_0 has been marked up for you, and also Flow F_2 as a check on your progress. On each diagram first write level numbers below the nodes then write labels above the nodes. Finally highlight the incremental flow that will be added to produce the flow shown in the next diagram. Levels, labels and incremental flow have been indicated on the first diagram as an example.



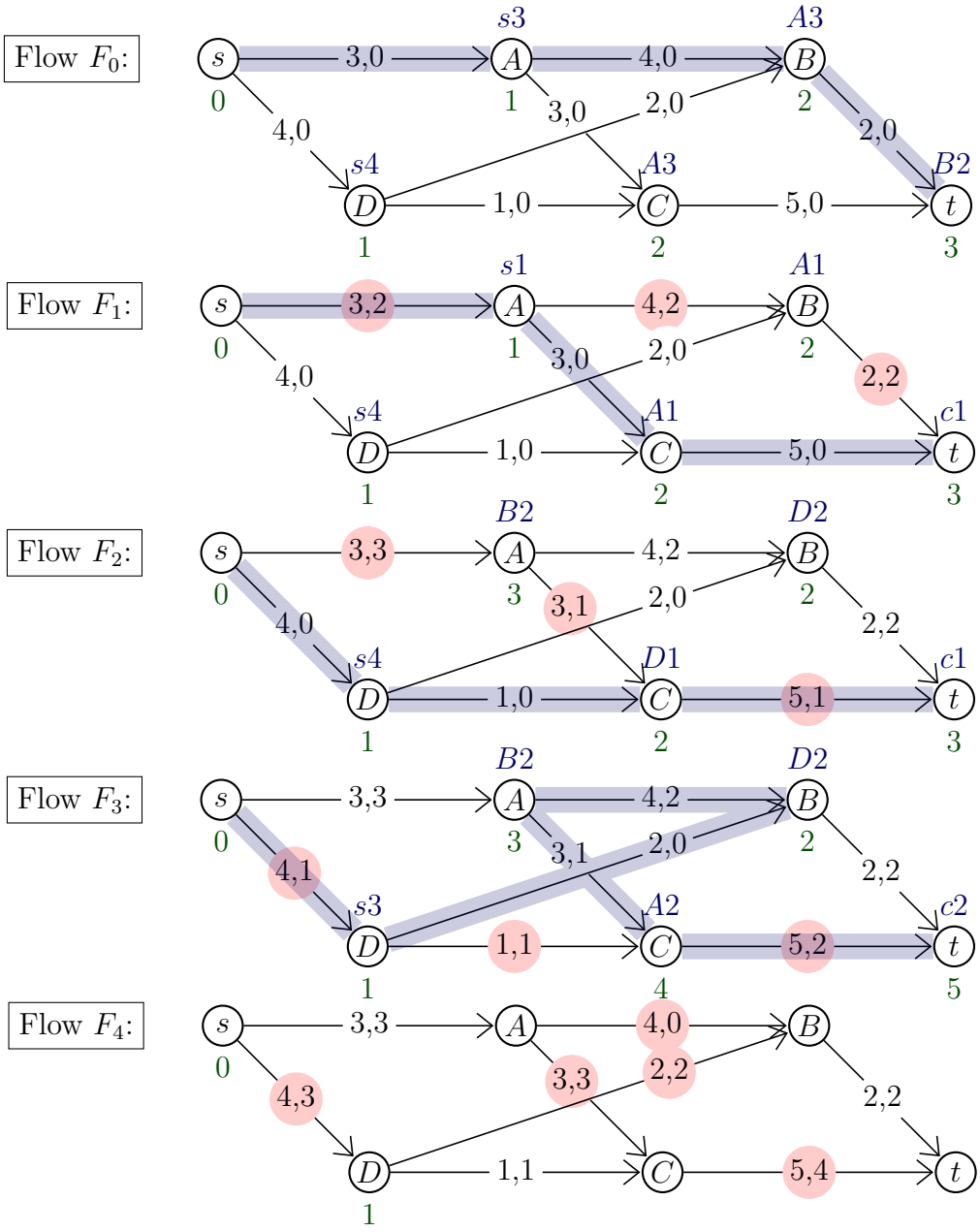
1 No level 2 nodes; algorithm terminates.

Use your max flow to help find a min cut which supports your answer.

From flow F_4 we see that the total flow into t is $6+3=9$, so this is the maximum flow value. Thus by the min-cut-max-flow theorem we seek a cut of this total capacity. It is not hard to find:



9. Here is another transport network to give you practice with the labelling algorithm. As for Q8, mark each diagram with levels, labels and a highlighted incremental flow. Stage 4, finding the incremental flow f_4 so that $F_4 = F_3 + f_4$, requires the use of a virtual flow. Check that your flow F_3 agrees with mine below before you attempt stage 4.



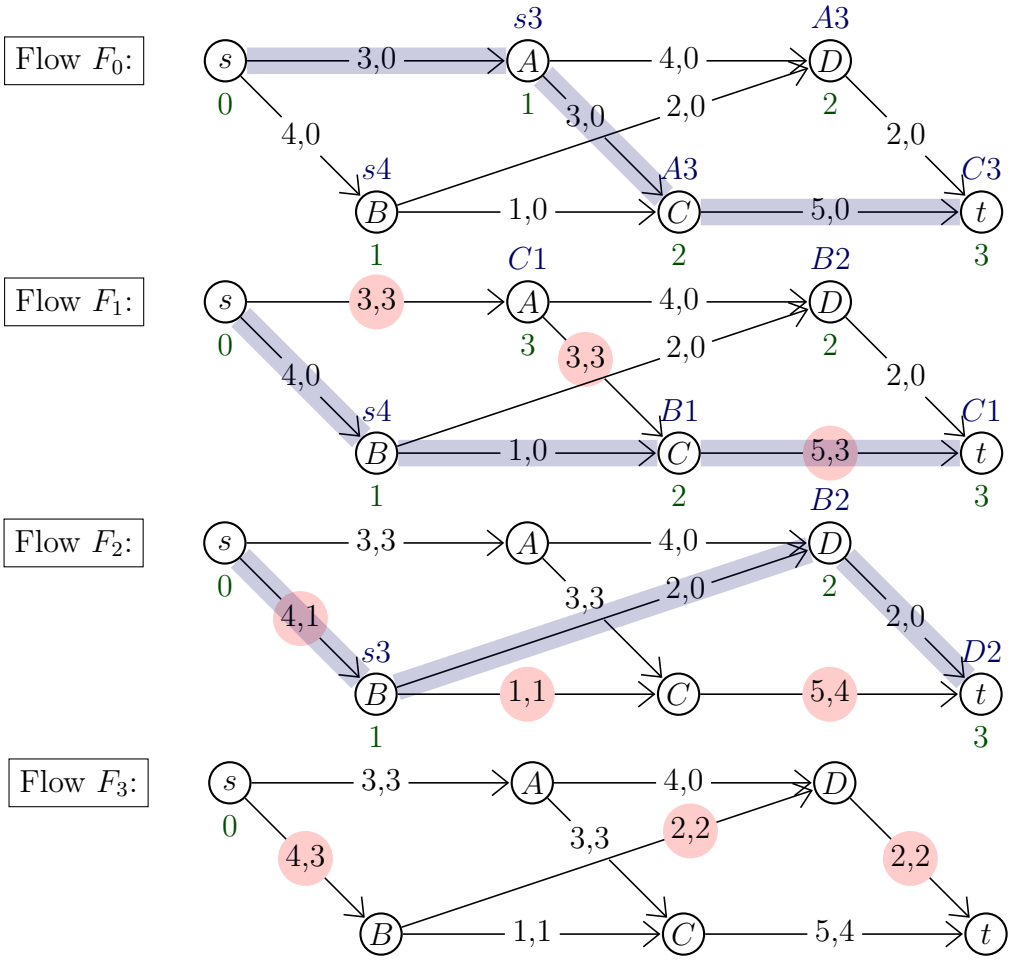
What step in the algorithm precipitates termination? Specify the node(s) involved.

From level 0 node s only the edge to D has spare capacity, so D is the only level 1 node. But no edges leading from D have any spare capacity, so there are no level 2 or higher nodes. In particular, the target t does not have a level. This terminates the algorithm.

Verify your max flow by specifying an appropriate cut.

From the inflow to t the max flow value is $2+4 = 6$. A cut with capacity 6 is through the edges sA, DB and DC. The partition is $\{\{s, D\}, \{A, B, C, t\}\}$.

10. Because alphabetical order is used at several steps, the progress of the labelling algorithm is affected by the choice of names for the nodes. To see this, apply the algorithm to the transport network below, which is the same as for Q9 except that two nodes have had their names swapped. You should find that only three stages are required to arrive at the same maximum flow, and that virtual flows are not required to get there.



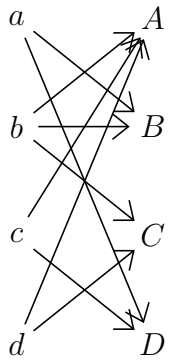
11. Briefly describe how a matching problem can be converted to a max flow problem for a transport network. In particular, explain the terms 'supersource' and 'supertarget' and say how capacities are assigned to the network.

To the arrow diagram for the relation add an extra node s, the 'supersource', and a a link from from s to each member of the domain of the relation. Then add another extra node t, the 'supersink' (super target), and a link to t from every member of the codomain of the relation. Now give all links a capacity of 1. A maximum flow now gives a maximal matching by picking all arrows from domain to codomain that have non-zero flow.

Remarks (*not necessary for the answer to this question but relevant to the presentation of the solutions to the remaining questions*):

1. Since all links have capacity 1, it is not necessary to display capacities.
2. Since a link with capacity 1 is either fully used or not used at all, rather than show spare capacity and flow we can show used/unused.
3. Similarly, there is no need to show potential flow *values* on vertex labels.
4. My convention will be show used links by heavy lines.
5. When a link AZ from domain to codomain is used, the links sA and Zt must also be used, so there is no logical need to mark them. However marking them helps to avoid making mistakes in annotating vertices, so I will still mark them, but less thickly.

12. For $S = \{a, b, c, d\}$ and $T = \{A, B, C, D\}$ a relation $R \subseteq S \times T$ is represented by the the arrow diagram at right.

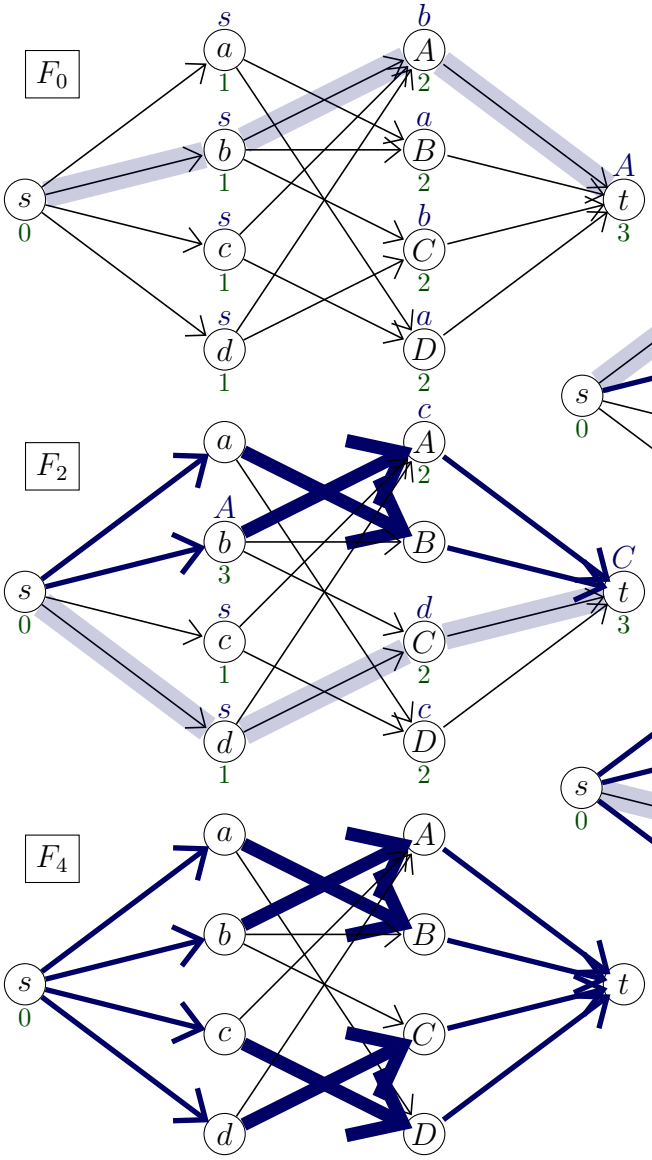


(a) List all possible full matchings $m : S \to T$. Do this without the aid of any particular algorithm; just use logical reasoning.
[*e.g.* If $(a, B) \in m$ then $(c, D) \in m$ since otherwise nothing will be matched with D.]

$$\begin{aligned} m_1 &= \{(a, B), (c, D), (b, A), (d, C)\} \\ m_2 &= \{(a, B), (c, D), (b, C), (d, A)\} \\ m_3 &= \{(a, D), (c, A), (b, B), (d, C)\} \end{aligned}$$

(b) Which matching is found by the vertex labelling algorithm?

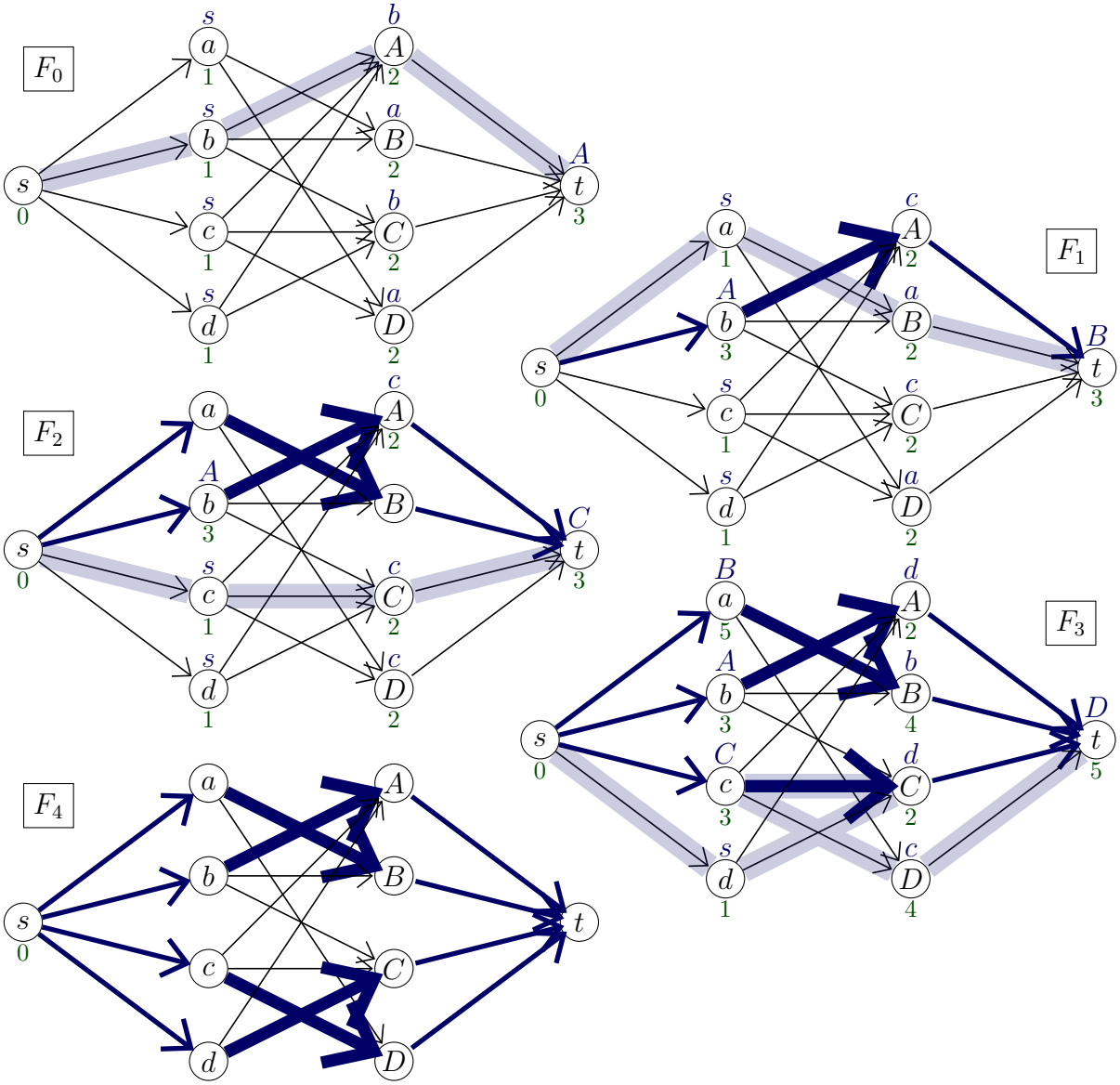
Justify your answer by marking up the flow diagrams below with levels, labels and incremental flows. List the resulting matching.



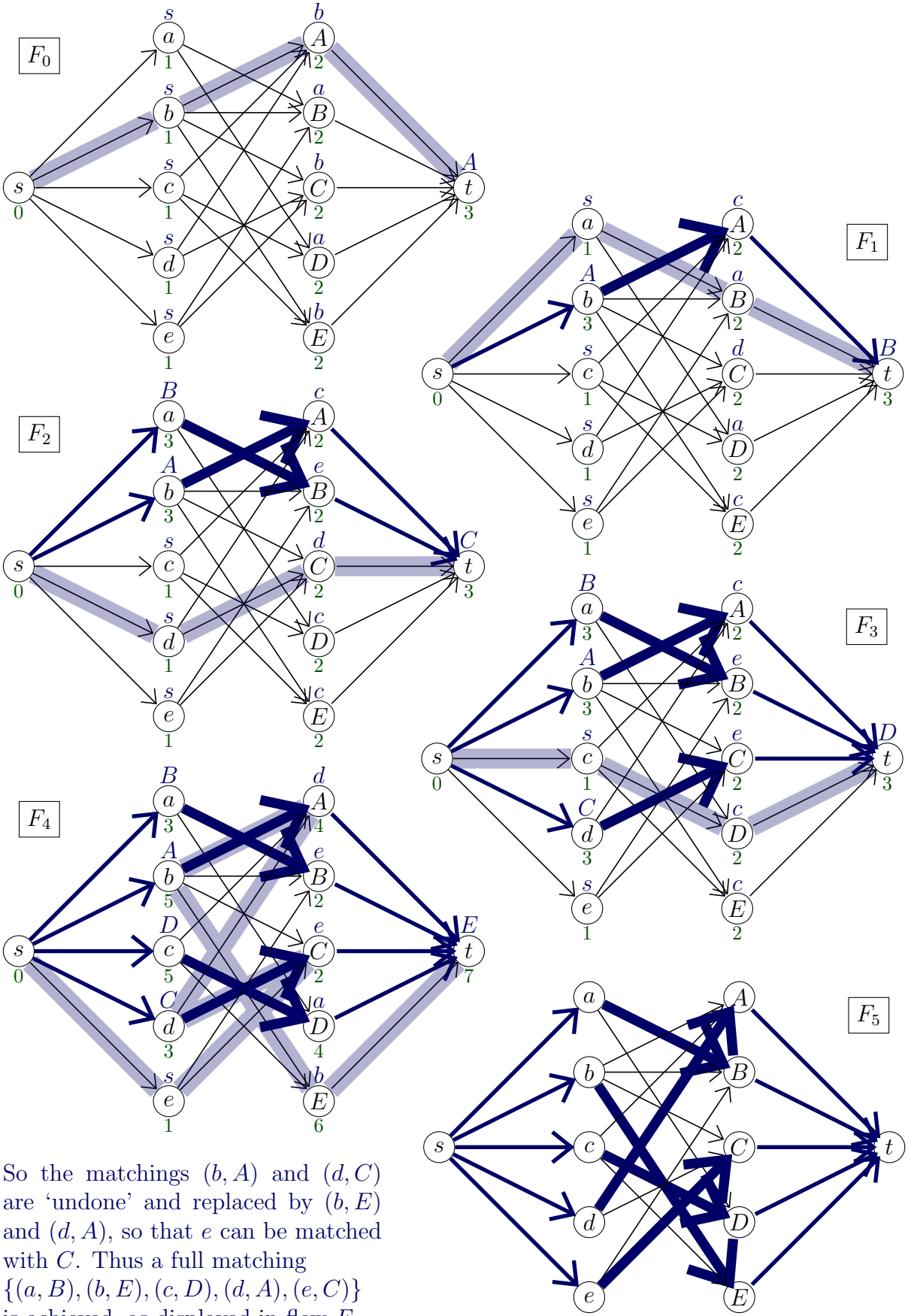
x	a	b	c	d
$m(x)$	B	A	D	C

So the matching is m_1.

13. Repeat Q12 with the addition of one extra pair (c, C) to R. You should end up with the same matching, but the extra choice actually makes the last stage a bit harder, with a virtual flow being needed.



14. Now repeat Q12 again with S augmented by an extra element e, T augmented by E and R augmented by the pairs $(e, B), (e, C), (b, E), (c, E)$. Does the algorithm lead to a full matching? If so list it, if not, say how the algorithm terminates.



So the matchings (b, A) and (d, C) are 'undone' and replaced by (b, E) and (d, A), so that e can be matched with C. Thus a full matching $\{(a, B), (b, E), (c, D), (d, A), (e, C)\}$ is achieved, as displayed in flow F_5.