

Algorithms and Data Structures 2011/12 Solutions & Marking-scheme for Coursework 2

1. (a) We must prove that every MST T of a given weighted graph (G, W) is also an MLST of that graph.

The definition of MLST is any tree T for which the “link-value” $\ell(T) = \max\{W(e) : e \in E(T)\}$ is the-minimum possible for spanning trees of the graph.

proof: (by contradiction)

Suppose there is some spanning tree T which is an MST of (G, W) but *not* an MLST of (G, W) . Now consider one of the MLSTs \hat{T} of (G, W) for comparison alongside T .

We know, by the definition of a MLST, that for any *spanning tree* T' of (G, W) satisfies $\ell(T') \geq \ell(\hat{T})$ when \hat{T} is an MLST. In the case of T , which we assume is not an MLST, it must be the case that $\ell(T) > \ell(\hat{T})$.

Let $e = (u, v) \in E(T)$ be an edge which satisfies $W(e) > \ell(\hat{T})$ - such an edge must exist in T , otherwise T would be an MLST (**NB1**). We know e is *not* an edge of \hat{T} . Now consider the path $\hat{p}_{u,v}$ between vertex u and vertex v in \hat{T} (**NB2**). Then, because \hat{T} is an MLST, we know that for every edge $f \in \hat{p}_{u,v}$, that $W(f) < W(e)$.

Now reconsider T , and delete the edge e from this tree. This gives two subtrees T_u and T_v which span $V(G)$. The path $\hat{p}_{u,v}$ must contain *at least one edge* $g = (w, z)$ such that $w \in V(T_u)$ and $z \in V(T_v)$, because if it is a path from u to v it must “cross over” from T_u to T_v at some stage. Hence $T_u \cup T_v \cup \{(w, z)\}$ is a spanning tree of the graph G (**NB3**). Also, we know $W(g) < W(e)$. Hence the cost of $(T \setminus \{e\}) \cup \{g\}$ is $W(T) - W(e) + W(g) < W(T)$.

Hence we have a contradiction to our assumption that T was an MST (**NB4**).

So every MST T must also be an MLST.

Key points: The most important points (marked **NB** above) for 1(a) were

- NB1** Observe that if an MST T is *not* an MLST, then T contains some edge e which has $W(e)$ *strictly greater* than the “link-weight” if the MLSTs for (G, W) - and then to focus on this edge for the rest of the proof.
- NB2** Then important to consider *the unique path* $\hat{p}_{u,v}$ *between the endpoints of* e in some MLST \hat{T} - and to observe that every single edge f on $\hat{p}_{u,v}$ have $W(f) < W(e)$ (because this is true for all f in \hat{T}).
- NB3** Finally important to consider $(T \setminus \{e\}) \cup \{g\}$, while *justifying that it is a tree* - easy bit that deleting $e = (u, v)$ chops T into T_u and T_v , *but also very important* that the path $\hat{p}_{u,v}$ from \hat{T} must contain an edge (call it g) with an endpoint from each of T_u and T_v (*very very important to show this*). Hence $T_u \cup T_v \cup \{g\}$ is a tree.
- NB4** Prove that the cost of $(T \setminus \{e\}) \cup \{g\}$ is then less than that of T , and derive the contradiction.

marking of (a): (10 marks)

There will be some attempts where the student chats on about how if T is an MST it was created by “adding a fringe edge” each time, so it must have all the cheapest edges etc etc... at most 3 marks for these or similar.

For a proof which generally follows the “shape” of mine, 10 marks go as follows:

- 1 mark for starting off with the counterexample of an MST, which is NOT an MLST.
- 1 mark for **NB1**, for noting that since ℓ of MLST is best possible over all spanning trees, T must contain some edge which is larger than the MLST value ℓ .
- 1 mark for considering an actual MLST (\hat{T} in my proof) which has the optimal link-weight ℓ .
- 1 mark for looking at the particular path between u and v in \hat{T} .
- 1 mark for mentioning that $W(f) < W(e)$ holds for all edges f in the path $\hat{p}_{u,v}$.
- 2 marks for showing why there must be some specific edge g along $\hat{p}_{u,v}$ which crosses between components T_u and T_v (after e deleted from T), and that hence $(T \setminus \{e\}) \cup \{g\}$ is also a spanning tree of G (**NB3** - this is the bit they are most likely to miss, some will just “add some edge from path”).
- 2 marks for showing that $(T \setminus \{e\}) \cup \{g\}$ has cost *strictly less than* T did.
- 1 mark for stating that this is a contradiction, hence it must be the case that all MSTs are also MLSTs.

Of course if they have given a correct rigorous proof in some other way, then give them the marks for it. I actually can't see another proof myself.

Feedback: If they have done one of the discussion-type answers, please write a sentence, saying “We are looking for a rigorous proof by contradiction (see solutions for the correct approach).”

Otherwise, if they have attempted a proof by contradiction but missed certain things - write in the specific things they missed as per my things-to-look-for(marks) above.

If they have a correct proof write “good” or “nice”.

(b) There are many examples (and (c) will give them a hint), an easy example is the example of a path p, q, r, s where the three path edges have weight 2 . . . together with an extra edge (q, s) of weight 1.

marking of (b): 2 marks for a correct 4-vertex (or 3-vertex) example.

1 mark if they have a correct 5-vertex example. 0 otherwise.

(c) (i) (2 marks)

There is no MST which contains only weight-1 edges - we can see this by the fact that the grid becomes disconnected if we remove all weight-2 edges. There are 4 different Spanning trees which have a single weight-2 edge (any of the 4 vertical edges), together with all of the weight-1 horizontal edges. This is best possible (with value $2 + 6 * 1 = 8$).

marking of (c)(i): 2 marks for stating the value 4 (don't have to say reason).

Only 1 mark if they don't have the 4 right but say in writing that it must have 1 vertical edge and all horizontals.

(c) (ii) (6 marks)

For the MLSTs, observe that every spanning tree must contain a weight-2 edge. Hence *every spanning tree* of our graph will be an MLST.(NB5)

We now count the number of spanning trees (STs), conditioning on the number of vertical edges in the tree.(NB6)

- STs with one vertical edge. If we are given a single vertical edge, there is a *unique* way to complete this to an ST by adding horizontals (must add all horizontals).
So one ST for each vertical edge \Rightarrow 4 of this kind.
- If we include *all 4* vertical (weight 2) edges in our ST, then to complete to a tree, we must add exactly one of the horizontal edges $((i, 0), ((i + 1), 0))$ (bottom row) or $((i, 1), ((i + 1), 1))$ for each of $i = 0, 1, 2$. There are exactly $2^3 = 8$ ways of making these choices. So 8 STs of this kind.
- STs with exactly 2 vertical edges.
There are $\binom{4}{2} = 6$ pairs of vertical edges. One pair is separated by horizontal distance three; 2 pairs by horizontal distance two; 3 pairs by horizontal distance one.
For “horizontal distance 3” we get a ST if and only if we add one length-3 horizontal path in entirety, and add all-but-one-edge of the other horizontal path. So 6 options (which of the 6 edges to delete).
In the case of “horizontal distance 2”, we must include all horizontal edges except one-edge of the two connecting length-2 paths between our vertical pair. So $2 \times 4 = 8$ options (2 for the particular vertical pair, 4 for the edge to drop).
In the case of “horizontal distance 1” (with 3 options for the pair of vertical edges), we must include all horizontal edges except one of the 2 horizontal edges between our vertical pair. So $3 \times 2 = 6$.
Total number of STs with two vertical edges is then 20.
- Considering STs with 3 vertical edges, it is either three in a row (2 options) or three with a gap (2 options).
In the “three in a row”, we must drop at least one of the horizontal pair between each adjacent pair of vertical edges: 2 by 2 ways to do this.

In the “three with a gap” case, we must drop one of a horizontal pair of edges, and one edge from a pair of length-2 horizontal paths: 2 by 4 ways to do this. So $2 \times 2 \times 2 + 2 \times 2 \times 4 = 24$ with three vertical edges.

So overall $4 + 8 + 20 + 24 = 56$.

Key points: The most important points (marked **NB** above) for 1(a) were

NB5 They should note that the number of MLST is the number of STs for this graph.

NB6 They should do something to partition the count. My method is in terms of number of vertical edges (and some subpartitioning); they might partition a different way. (eg, 8 vertices, so ST has 7 edges. Take the 10 edges and consider all ways of dropping 3, checking if ST results)

Other solutions:

(A) They might use Kirchoff’s matrix-tree theorem for counting spanning trees of a graph. This allows them to just compute the determinant of any minor of the Laplacian matrix for our graph - this is the number of trees. (Google this if you don’t know it - its everywhere).

(B) Might have written a program to enumerate all trees and count them.

marking (c)(ii): (6 marks total)

1 mark for noticing it is the number of spanning trees.

1 mark for breaking the count up into subcases that do *do not overlap* and then add - partitioning might be my way, might be another.

4 marks each for counting correctly the subcases - (and similar if they broke it up a different way). For my first 2 cases, they should write about that much detail - could give a bit less info for the last two cases than me.

Other solutions:

Assume the 1 marks for the STs observation as above.

The other 5 marks . . .

- If they do no workings and write 56 as answer, give them 2 marks.
- If they do some workings, and come up with an approximate value, or an upper bound, give up to 3 marks depending on how close to 56, how good the details.
- If they use approach (A), with justification of the Theorem and the definition of Laplacian matrix, give them up to all 5 marks (but also write on their script that the aim was to get them to do it by first principles - wouldn’t have Kirchoff/Laplacian to hand in an exam).
- If they use approach (B), give them up to all 5 marks depending on how well they describe their program and how it did the enumeration and checking (but also write on their script that the aim was to get them to do it by first principles - wouldn’t have computer/compiler to hand in an exam).

2. Line Formatting problem.

Let x_1, \dots, x_m be the input words.

For $i \leq j \leq n$, let $\ell_i = |x_i|$, the length of the i th word. First of all, we let $e[i, j]$ be the empty space at the end of the line if x_i, \dots, x_j are printed in a line:

$$e[i, j] = L - \sum_{k=i}^j \ell_k + i - j.$$

Of course $e[i, j]$ may be negative, which means that we cannot print x_i, \dots, x_j in a single line. It is quite useful to perhaps store the $e[i, j]$ values in a $n \times n$ matrix at the very beginning of the DP algorithm - this can be done in $\Theta(n^2)$ time (*though they will be able to get away without doing it*).

Key observation: For most of the solution, we work to construct a solution in which we also count the square of the trailing-spaces *on the last line*, and then fix it up at the end. (They have been given this as a hint)

There is more than one solution to this, depending on which recurrence is used. I give the most-likely one first.

Let $f[i, j]$ denote the sum of squares of the trailing spaces (including final line) of an *optimal* line-formatting of words x_i, \dots, x_j . Then we can write the following recurrence (**NB7**):

$$f[i, j] = \begin{cases} e[i, j]^2 & \text{if } e[i, j] \geq 0 \\ \min_{i \leq k < j} (f[i, k] + f[k + 1, j]) & \text{otherwise.} \end{cases}$$

This recurrence will form the basis of our $\Theta(n^3)$ algorithm. Now the details.

After this introduction I now deal with the points that I asked for in the question:

(a) I asked for details of the tables.

The main table is the table f , where $f[i, j]$ (**NB8**) represents the cheapest formatting of words w_i, \dots, w_j , for every $1 \leq i \leq j \leq n$. It is an $n \times n$ table.

The partner table to f is the table s , with $s[i, j]$ representing some index k (say) such that we can obtain $f[i, j]$ (the cheapest formatting) as $f[i, k] + f[k + 1, j]$ (ie, some optimal/cheapest formatting of w_i, \dots, w_j has a line-break after word w_k). (**NB9**)

Finally we have a table $e[i, j]$ which represents the number of free spaces on a line containing words w_i, \dots, w_j (this will sometimes be negative). *they do not need to have this table though. This could be computed on the fly.*

All these tables have dimensions $n \times n$

(b) I asked for the algorithm to compute the entries to the tables. I wouldn't expect them to give pseudocode but here it is. The input to this algorithm is the vector ℓ of word lengths ($\ell_i = |w_i|$ for all i).

Algorithm SPLITPOINTS(ℓ)

1. $n \leftarrow \ell.length$
2. **for** $i \leftarrow 1$ **to** n **do**
3. $e[i, i] \leftarrow L - \ell_i$
4. **for** $j \leftarrow i + 1$ **to** n **do**
5. $e[i, j] \leftarrow e[i, j - 1] - \ell_j - 1$
6. **for** $d \leftarrow 1$ **to** n **do**
7. **for** $i \leftarrow 1$ **to** $n - d + 1$ **do**
8. $j \leftarrow i + d - 1$
9. **if** $e[i, j] \geq 0$ **then**
10. $f[i, j] \leftarrow e[i, j]^2$
11. $s[i, j] \leftarrow \text{NIL}$
12. **else**
13. $f[i, j] \leftarrow \infty$
14. **for** $k \leftarrow i$ **to** $j - 1$ **do**
15. $q \leftarrow f[i, k] + f[k + 1, j]$
16. **if** $q < f[i, j]$ **then**
17. $f[i, j] \leftarrow q$
18. $s[i, j] \leftarrow k$
19. **if** $s[1, n] \neq \text{NIL}$ **then**
20. $k \leftarrow n - 1$
21. **while** $s[k + 1, n] = \text{NIL}$ **do**
22. **if** $f[1, k] < f[1, n]$ **then**
23. $f[1, n] \leftarrow f[1, k]$
24. $s[1, n] \leftarrow k$
25. $k \leftarrow k - 1$
26. **return** s

The loop in lines 2–4 computes e in a straightforward way. The loop in lines 6–18 computes f and s . This is done in a way very similarly to the MATRIX-CHAIN-ORDER algorithm using my recurrence above.

Note that even if they have not pre-computed the table e , the work to be done to compute these “on the fly” is $O(n)$ work, at most n^2 times. So it’s ok for the $\Theta(n^3)$ alg.

Correctness depends on our recurrence. Clearly the best formatting for x_i, \dots, x_j will correspond to at least one decomposition for a specific k (unless x_i, \dots, x_j all fit on one line) - any k for which w_k is a last-on-line-word for some optimal formatting will do. Hence in considering all k , we will surely pick up an optimal formatting (**NB10**).

Also note that the algorithm fills in $f[i, j]$ and $s[i, j]$ in order of increasing $(j - i)$ - and since the recurrence only considers k values which keep at least one word to either side of the split, this means that all the lookups to $f[i, k]$ and $f[k + 1, j]$ are *already saved in the table* when we look them up. (**NB11**)

Lines 19-25 do the “fixing-up” to adapt to the case where we don’t include the last line. We consider all possible options for keeping trailing words on the last line.... and that will fit ($s[k + 1, n] = \text{NIL}$), and see which $s[1, k]$ is least for these options. (**NB12**)

We also update the table s to take this into account.

(c) Lines 1-5 take $O(n^2)$.

Lines 19-25 take $O(n)$ time.

Lines in between take $O(n^3)$ - we consider at most n^2 pairs of i, j , and do $O(1)$ work for every $i \leq k \leq j$. So $O(n^3)$ overall.

The $\Omega(n)^3$ comes from the nested loop in lines 6-18. We will do $\Omega(1)$ work for all i, k, j such that $1 \leq i \leq k < j \leq n$, ie $\sum_{i=1}^n \sum_{j=i+1}^n (j - i)\Omega(1)$. This is at least

$$\begin{aligned} \Omega(1) \sum_{i=1}^{n/2} \sum_{j=i}^n (j - i) &\geq \Omega(1) \sum_{i=1}^{n/2} \sum_{j=n/2}^{3n/4} (j - i) \\ &\geq \Omega(1) \sum_{i=1}^{n/2} \sum_{j=3n/4}^n (n/4) = \Omega(1) \frac{n}{2} \frac{n}{4} \frac{n}{4} = \Omega(n^3). \end{aligned}$$

(**NB13**)

(d) Here is pseudocode to print out the formatting. The input to this algorithm is the vector x of actual words in the paragraph:

Algorithm PRETTY-PRINT(x)

1. $n \leftarrow x.length$
2. **for** $i \leftarrow 1$ **to** n **do**
3. $\ell_i \leftarrow |x_i|$
4. $s \leftarrow \text{SPLITPOINTS}(\ell)$
5. REC-PRINT($x, s, 1, n$)

Algorithm REC-PRINT(x, s, i, j)

1. **if** $s[i, j] = \text{NIL}$ **then**
2. **print** $x_i \dots x_j$ **newline**
3. **else**
4. REC-PRINT($x, s, i, s[i, j]$)
5. REC-PRINT($x, s, s[i, j] + 1, j$)

Alternative solution: There is a *nicer, simpler* solution where we work with a table $f[i]$, representing the cheapest formatting for words w_1, \dots, w_i (and also a table $s[i]$). We can express the value of $f[i]$ recursively(**NB7'**)::

$$f[i] = \begin{cases} e[1, i]^2 & \text{if } e[1, i] \geq 0 \\ \min_{1 \leq k < j} (f[k] + e[k + 1, j]) & \text{otherwise.} \end{cases}$$

With this recurrence it is possible to achieve a $\Theta(n^2)$ algorithm. Reason I set the problem up as $\Theta(n^3)$ is because the structure of that recurrence is like the one for Matrix-chain multiplication.

Key points: (in regard to $\Theta(n^3)$ soln)

The most important points (marked **NB** above) for 1(a) were

NB7 So so important to have a recurrence which is *correct* - ie relates the solution for $f[i, j]$ to smaller subproblems correctly.

Note I am assuming they will be working with the include-square-of-last-line version for most of the solution. They may have a more complicated recurrence if they tried to work with the “special case for last line” version throughout.

(similarly for **NB7'** in the $\Theta(n^2)$ alg)

NB8, NB9 Must state which tables will be used and what entries mean (same for the $\Theta(n^2)$ alg)

NB10 *Justify the correctness of their recurrence.* Must show the recurrence will definitely consider an optimal/cheapest solution.

NB11 Must must design the algorithm so that the table is filled-in in an order that ensures that when computing $f[i, j]$, all the solutions to relevant subproblems wrt the rhs of recurrence *will have been previously computed and saved* and hence are just “lookups” (not recursive calls)

NB12 (assuming they mostly worked with the include-square-of-last-line version) they must do the fixing-up to get the answer to the original problem.

Marking: There will probably be some solutions which are “greedy” algorithms - ie, they fill the first line as much as possible, then the next etc etc... These get at most 8 marks because they do not solve the *asked for* problem.

For DP solutions . . . use the following scheme (but also be aware that the various pieces, justifications etc) may not be always neatly demarkated by (a), (b) etc).

(a) (5 marks overall)

Give 4 marks for defining $f[i, j]$ and saying what the entries correspond to in terms of w_1, \dots, w_n and the problem (or 4 marks for $f[i]$ for the $\Theta(n^2)$ version).

Some of them also had a “pair” of fs if they did their solution before I gave them the hint to “include square for-last-line” for the main part of Alg. Try to work out if these messier solutions are right, and award marks if so.

Give the 5th mark for $s[i, j]$ (or $s[i]$ in $O(n^2)$ alg).

$e[i, j]$ is not *really* necessary - but if they define it, and give details of its entries, and don't have all 5 for (a), then give an extra 1 for this.

(b) (5 marks overall)

I do *not* expect pseudocode like I gave; a good and correct English explanation is fine.

2 marks for a correct recurrence for f (either $O(n^3)$ version or $O(n^2)$) and its *justification* (like I have written in English in (b)). If they have a more complicated recurrence, do your best to give marks for it if *correct* (write feedback if its not correct).

1 mark for using width d as the parameter to the first **for**-loop. This “realises” (**NB11**).

1 mark for good details apart from above, including s .

1 mark for showing how to fix-up the answer to take care of the last-line.

(if they have the $O(n^2)$ solution use a analogous mark scheme)

(c) (3 marks overall)

2 marks for $O(n^3)$. 1 mark for $\Theta(n^3)$.

If they did the alternative DP alg, 2 marks for $O(n^2)$, and 1 extra for showing $\Theta(n^2)$.

(d) 2 marks allocated appropriately.

feedback:

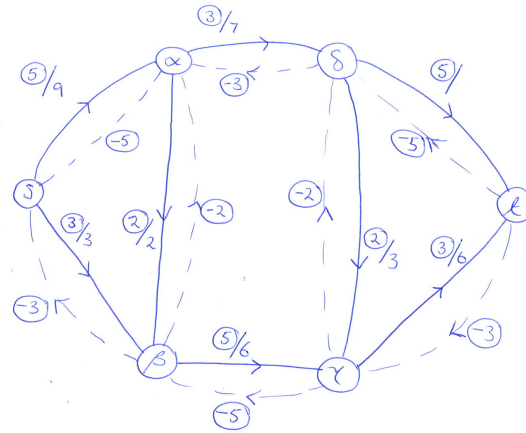
If they give a greedy algorithm , write on the sheet that it is *not* a Dynamic programming Algorithm and (is not correct. Give them the example of $L = 80, w_1 = 60, w_2 = 15, w_3 = 50, w_4 = 30, w_5 = 40$ - simple example of when greedy gives the wrong answer.

For DP solutions - write notes where anything is wrong. Also write notes indicating corrections - or, if their solution is very wrong, point them to online solutions.

If they have extra tables, please take time to work out how they are proceeding and whether their alg is correct. There may be extra tables if they have worked with the “special case for last line” variant of the problem throughout their Alg.

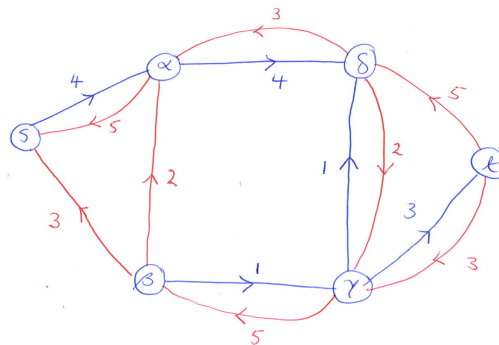
3. Network Flow question.

Given the tables as specified in the coursework solution, can be helpful (though no marks going to this) to draw capacities and original flow in pictorial form. Here it is:



- (a) The capacity of edge (u, v) in the residual network is by definition $c_f(u, v) = c(u, v) - f(u, v)$. Using this rule for all $u, v \in V \times V$, (really only relevant if at least one of (u, v) or (v, u) had non-zero capacity in the capacity table), we get the following residual network. As is the rule, we only show edges with non-zero residual capacity. Note I used red to indicate “new” arcs which appear only because they represent the option to reverse shipment of flow in the opposite direction.

(a) Residual Network



Marking of (a): (4 marks)

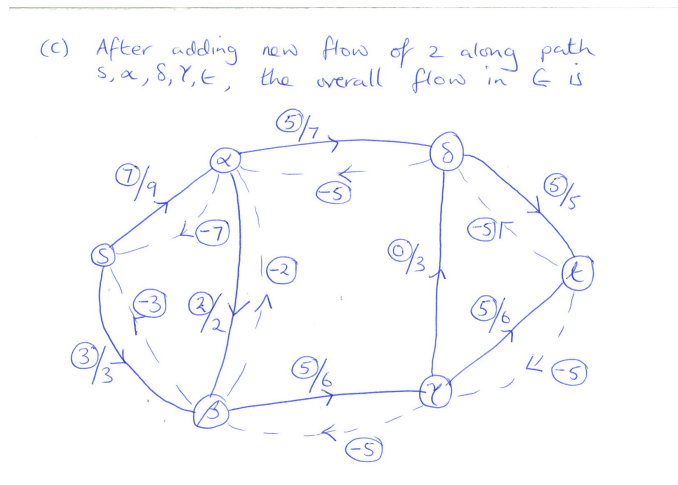
1 mark for explaining how they got the residual capacities (with my equation given above).

Other 3 marks for the details - subtract $\frac{1}{2}$ mark for each error in the residual network, add and round up. (One slip for free). Or if they have omitted all the “mirror” arcs but have all the forward arcs right, they get 1 of the 3.

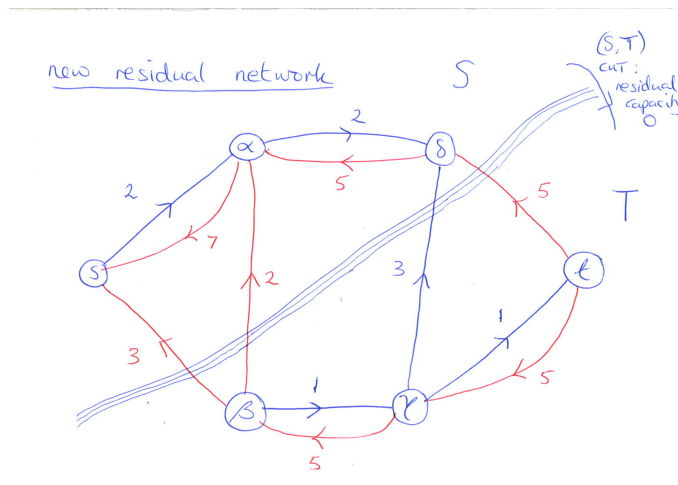
- (b) The unique augmenting path is $s \rightarrow \alpha \rightarrow \delta \rightarrow \gamma \rightarrow t$. The minimum residual capacity along this path is 2, therefore it is possible to route 2 extra units of flow along the augmenting path.

Marking of (b): (3 marks) 2 marks for the augmenting path, 1 mark for stating that the maximum extra flow is 2.

- (c) Just add 2 to the flows in edges (s, α) , (α, δ) and (γ, t) making them 7, 5 and 5, respectively. Also cancel the flow in (γ, δ) (equivalent to adding 2 to (δ, γ)). Here is the network with the updated flow after adding this augmenting path flow:



We can then recompute/update the residual network as follows:



The set of “reachable” vertices in the residual network from the source s is $\{s, \alpha, \delta\}$. Therefore the (S, T) cut $(\{s, \alpha, \delta\}, \{\beta, \gamma, t\})$ has value 0 in the residual network, and

value 10 in the original graph.

This means that by the MaxFlow-MinCut theorem, the updated flow of value 10 is max possible.

Marking of (c): (3 marks overall)

1 mark for the new flow. 2 marks for the cut and saying why it shows the updated flow (original + 2 extra) is optimal.

(they do not have to draw the updated residual network as long as their answer is right)

(d) (2 marks)

The flows in the cut edges are uniquely determined. But once these are fixed, the flows in all remaining edges happen to be determined by conservation of flow at the vertices - can see this for the “s-side” by considering δ first and noticing that it must be the case that

$$f(\delta, \alpha) + f(\delta, \gamma) + f(\delta, t) = 0,$$

that is we require

$$f(\delta, \alpha) = f(\gamma, \delta) + f(t, \delta) = 0 - 5 = -5$$

(for any residual network in which the cut from $\{s, \alpha, \delta\}$ to $\{\beta, \gamma, t\}$ is zero, which implies $f(\delta, \gamma) = 0$ and $f(t, \delta) = -f(\delta, t) = -5$). So we must always have $f(\alpha, \delta) = 5$ for a max flow. A similar argument holds for the edge (s, α) and for the two edges (β, γ) and (γ, t) on the other side of the cut.

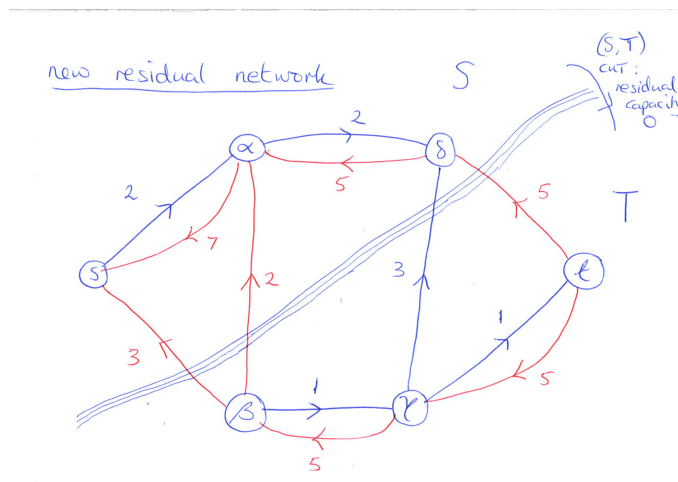
Marking of (d):

2 marks for a carefully argued solution.

Only 1 mark for some sort of argument.

(e) (3 marks overall)

As we showed in (c), the residual network is:



Note that there are no edges from $\{s, \alpha, \delta\}$ to $\{\beta, \gamma, t\}$. (It is a min. cut, after all.) Therefore any cycle would have to lie entirely within S or entirely within T .

In S , there is no arc in either direction between s and δ . So these cannot participate in a cycle together. Hence in S , no simple cycle longer than 2. No non-trivial cycle.

In T , there is no arc in either direction between β and t . So similarly, T has no non-trivial cycle.

Marking of (e):

1 mark for the residual network.

1 mark for noting any cycle must be entirely in S or entirely in T . 1 mark for the remaining details. (or if they took another approach, 2 marks for good details)

Mary Cryan