Queen Mary
University of London

B. Sc. Examination by course unit 2014

MTH6105 Algorithmic Graph Theory MARKING SCHEME

Duration: 2 hours

Date and time: 12 May 2014, 1430h

Apart from this page, you are not permitted to read the contents of this question paper until instructed to do so by an invigilator.

You should attempt all questions. Marks awarded are shown next to the questions.

Calculators ARE permitted in this examination. The unauthorized use of material stored in pre-programmable memory constitutes an examination offence. Please state on your answer book the name and type of machine used.

Complete all rough workings in the answer book and cross through any work which is not to be assessed.

Important note: the Academic Regulations state that possession of unauthorized material at any time by a student who is under examination conditions is an assessment offence and can lead to expulsion from QMUL.

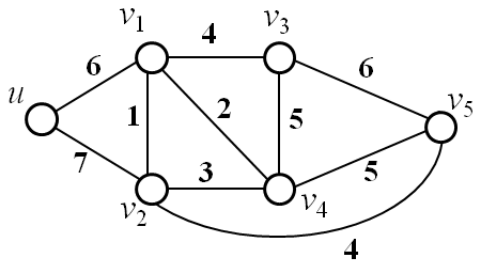
Please check now to ensure you do not have any notes, mobile phones or unauthorised electronic devices on your person. If you have any, then please raise your hand and give them to an invigilator immediately. Please be aware that if you are found to have hidden unauthorised material elsewhere, including toilets and cloakrooms, it will be treated as being found in your possession. Unauthorised material found on your mobile phone or other electronic device will be considered the same as being in possession of paper notes. Disruption caused by mobile phones is also an examination offence.

Exam papers must not be removed from the examination room.

Examiner(s): R.W. Whitty, L.H. Soicher

© Queen Mary, University of London (2014)

Question 1 Consider the graph G, having undirected weighted edges, as shown below:



(a) Apply Kruskal's Algorithm to the graph G to construct a minimum-weight spanning tree. List the iterations by which the algorithm constructs the spanning tree and write down the final output of the algorithm. [10]

(b) Repeat part (a) using Prim's Algorithm in place of Kruskal's Algorithm. State one advantage which Prim's Algorithm has over Kruskal's Algorithm. [10]

SOLUTION:
Question is unseen but similar to coursework

(a) Kruskal repeatedly adds min weight edges subject to not creating a cycle:

v_1v_2, v_1v_4	added
v_2v_4	rejected — cycle with v_1v_2, v_1v_4
v_2v_5, v_1v_3	added
v_3v_4	rejected — cycle with v_1v_3, v_1v_4
v_4v_5	rejected — cycle with v_1v_2, v_1v_4, v_2v_5
uv_1	added

[7]

[A series of drawings of trees is also acceptable but it must be indicated why some edges are rejected for inclusion in the tree, i.e. because they created cycles.]

Final output of algorithm:

$$v_1v_2,\ v_1v_4,\ v_2v_5,\ v_1v_3,\ uv_1$$

with total weight $1 + 2 + 4 + 4 + 6 = 17$. [3]

(b) Prim maintains a table of cheapest links to the growing tree. It should look something like:

© Queen Mary, University of London (2014)

	Vertices:	u	v_1	v_2	v_3	v_4	v_5	U	$V(T)$	$E(T)$
	Initialise:		(u, ∞)	(u, ∞)	(u, ∞)	(u, ∞)	(u, ∞)	u	$\{u\}$	$\emptyset$
Iteration										
1	Update labels:		$(u, 6)$	$(u, 7)$	(u, ∞)	(u, ∞)	(u, ∞)			
	Extend tree:		✔					v_1	$\{u, v_1\}$	$\{uv_1\}$
2	Update labels:			$(v_1, 1)$	$(v_1, 4)$	$(v_1, 2)$	(u, ∞)			
	Extend tree:			✔				v_2	$\{u, v_1, v_2\}$	$\{uv_1, v_1v_2\}$
3	Update labels:				$(v_1, 4)$	$(v_1, 2)$	$(v_2, 4)$			
	Extend tree:					✔		v_4	$\{u, v_1, v_2, v_4\}$	$\{uv_1, v_1v_2, v_1v_4\}$
4	Update labels:				$(v_1, 4)$		$(v_2, 4)$			
	Extend tree:						✔	v_5	$\{u, v_1, v_2, v_4, v_5\}$	$\{uv_1, v_1v_2, v_1v_4, v_2v_5\}$
5	Update labels:				$(v_1, 4)$					
	Extend tree:				✔			v_3	$\{u, v_1, v_2, v_4, v_5, v_3\}$	$\{uv_1, v_1v_2, v_1v_4, v_2v_5, v_1v_3\}$

[Not all the columns are essential: full marks provided the vertex columns and the final tree column are shown. A series of graph drawings is equally acceptable provided the label pairs are marked on all vertices (so each drawing must show all vertices at each step of the algorithm).]

Final output of algorithm:

$$uv_2,\ v_1v_2,\ v_1v_4,\ v_2v_5,\ v_1v_3$$

with total weight $6 + 1 + 2 + 4 + 4 = 17$. [2]

An advantage for Prim's algorithm is that it is always able to make an automatic choice of next edge, just by checking the current table row; Kruskal, on the other hand, must continually check through the graph to ensure no cycles are being created. [3]

Question 2 A graph G is specified as a list of 8 vertices

$$u,\ a,\ b,\ c,\ d,\ e,\ f,\ g;$$

together with a list of weighted edges (the first entry in each pair is the edge, the second entry is its weight):

$$(ua, 6),\ (ub, 3),\ (uc, 1),\ (ac, 4),\ (ad, 2),\ (af, 1),\ (bc, 1),$$
$$(bg, 5),\ (cd, 1),\ (ce, 1),\ (df, 4),\ (eg, 6),\ (fg, 1).$$

(a) Apply Dijkstra's Algorithm to find, for each vertex v, the length of a shortest path from u to v. You should describe clearly, either by drawings or by means of a table, the iterations which the algorithm performs. [12]

(b) The spanning tree you identified in part (a) is a minimum-weight spanning tree of G. By choosing a different weight for edge bg construct a weighted graph in which the output of Dijkstra's algorithm will **not** be a minimum-weight spanning tree. You are not required to re-run Dijkstra's algorithm but you should explain how the output will differ for the new graph. [8]

© Queen Mary, University of London (2014)

SOLUTION:

Question is unseen but similar to coursework. Part (b) is unseen and requires some original thought.

(a) Assuming a table is used to show the iterations of Dijkstra's algorithm we should have the following (omitting the column for vertex u):

	a	b	c	d	e	f	g	U	tree edges
Initialise:	(u, ∞)	(u, ∞)	(u, ∞)	(u, ∞)	(u, ∞)	(u, ∞)	(a, ∞)	u	$\emptyset$
Iteration									
1 Update labels:	$(u, 6)$	$(u, 3)$	$(u, 1)$	(u, ∞)	(u, ∞)	(u, ∞)	(u, ∞)		
Extend tree:			**1**					c	$\{uc\}$
2 Update labels:	$(c, 1+4)$	$(c, 1+1)$		$(c, 1+1)$	$(c, 1+1)$	(u, ∞)	(u, ∞)		
Extend tree:		**2**						b	$\{uc, bc\}$
3 Update labels:	$(c, 5)$			$(c, 2)$	$(c, 2)$	(u, ∞)	$(b, 2+5)$		
Extend tree:				**2**				d	$\{uc, bc, cd\}$
4 Update labels:	$(d, 2+2)$				$(c, 2)$	$(d, 2+4)$	$(b, 7)$		
Extend tree:					**2**			e	$\{uc, bc, cd, ce\}$
5 Update labels:	$(d, 4)$					$(d, 6)$	$(b, 7)$		
Extend tree:	**4**							a	$\{uc, bc, cd, ce, ad\}$
6 Update labels:						$(a, 5)$	$(b, 7)$		
Extend tree:						**5**		f	$\{uc, bc, cd, ce, ad, af\}$
7 Update labels:							$(f, 5+1)$		
Extend tree:							**6**	g	$\{uc, bc, cd, ce, ad, af, fg\}$

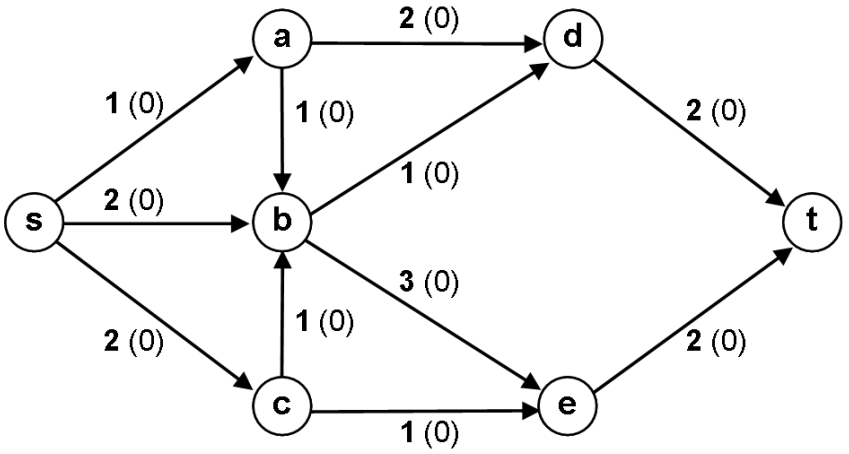
[12]

[As in Q1(c) not all columns are essential but the vertex columns and final tree column must be shown for full marks. The series of drawings which is acceptable instead of the table is omitted.]

(b) Suppose that the weight bg is changed to 3. Then the shortest path from u to g becomes uc, cb, bg of weight 5. So now the edge fg of weight 1 will no longer form part of the output spanning tree but will be replaced by the edge bg. Then the total weight of the spanning tree is increased and it is no longer minimum-weight. [8]

Question 3 The following picture shows a capacitated st-network (N, s, t) with the bold edge-weights being edge capacities c and the weights in parentheses being initial values for an st-flow f (that is, each edge e has the pair $\boldsymbol{c(e)}$ $(f(e))$ as its label).

© Queen Mary, University of London (2014)



(a) Explain why we would say the two st-paths consisting of the edges

$$sa,\ ad,\ db,\ be,\ et$$

and

$$sc,\ ce,\ ed,\ dt,$$

respectively, are f-*unsaturated*, in the context of the Ford-Fulkerson Algorithm for finding a maximum st-flow.

Re-draw the above picture to show the new flow which results if the flow f is augmented on the two paths specified. [6]

(b) Suppose the flow f is augmented along the two paths given in part (a). Show how the Ford-Fulkerson Algorithm, applied to the resulting graph, will find a maximum st-flow in this graph.

(You may indicate the steps of the algorithm by adding to a drawing of the network like the one above, but be careful to describe clearly what your additions mean.) [9]

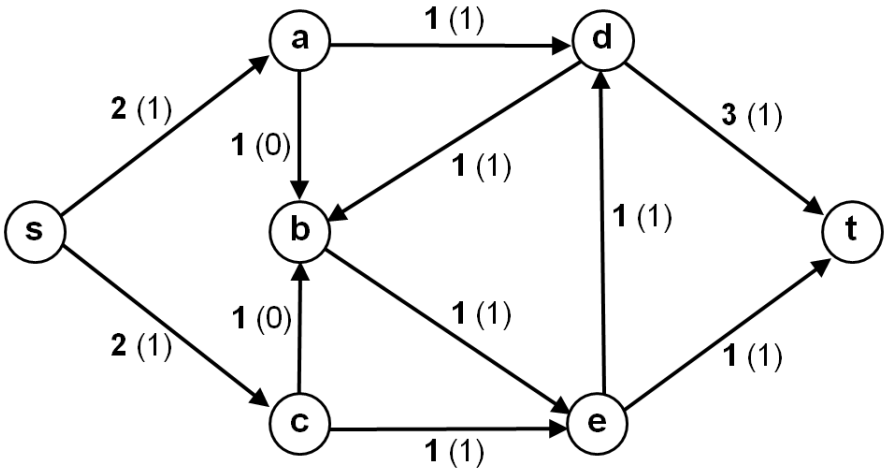
(c) State (but do not prove) the Max-Flow Min-Cut Theorem. Explain briefly why the theorem guarantees that the flow which you found in part (b) must be maximum. [5]

SOLUTION:

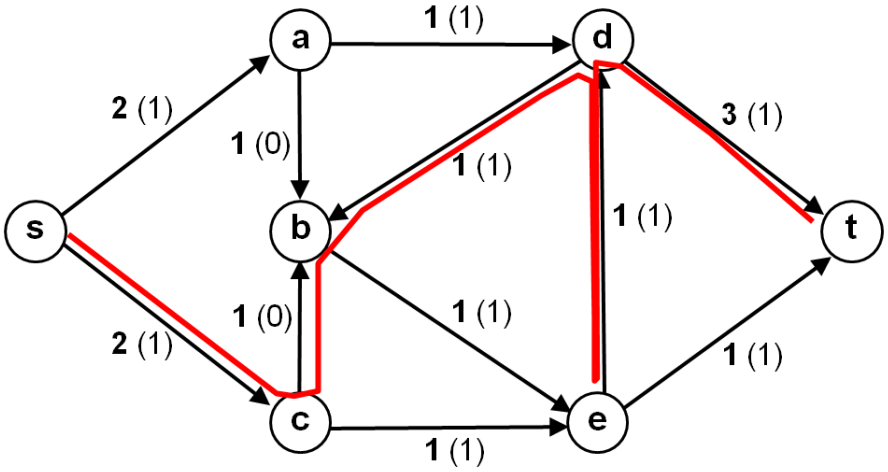
Question is unseen but similar to coursework. Part (c) is book-work.

(a) sa, ad, db, be, et and sc, ce, ed, dt, are f-unsaturated because each edge directed forward has flow value less than capacity (there are no backward edges). [2]

The updated flow is shown below with an f-unsaturated tree for the new flow values: [4]



(b) First iteration constructs the following maximal unsaturated tree:



[1]

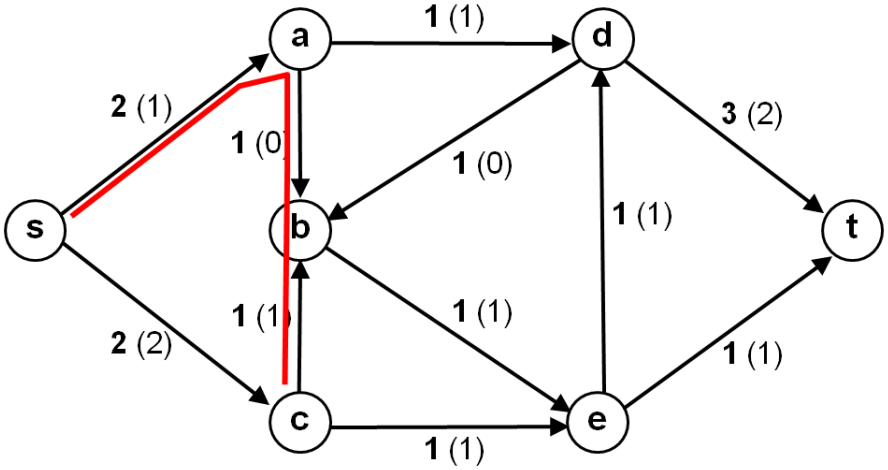
An f-unsaturated st-path is identified:

edges:	sc	cb	bd	dt
slack:	1	1	1	2

, with slack value $\min(1, 1, 1, 2) = 1$. [6]

Updating the network gives the following, with new f-unsaturated tree shown:

© Queen Mary, University of London (2014)



The Ford-Fulkerson algorithm terminates with this iteration since the tree fails to reach vertex t. [3] [2]

(c) The Max-Flow Min-Cut Theorem says that the maximum value of a flow in a network is equal to the minimum capacity of an st-cut in that network. [2]

In part (b) we achieved a flow value $|f| = 4$. Since edges dt and et form a cut of capacity 4, the Max-Flow Min-Cut Theorem says this flow must be maximum. [3]

© Queen Mary, University of London (2014)

Question 4 (a) Define what is meant by a *matching* in an undirected graph. [2]

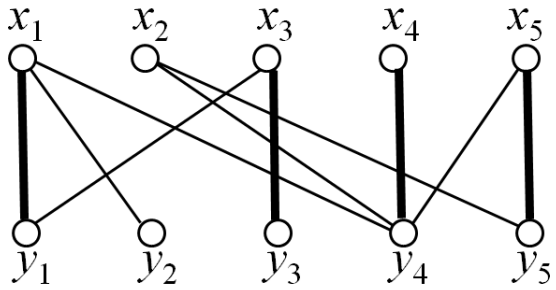
(b) A bipartite graph G is specified as consisting of a partitioned vertex set $V = X \cup Y$, $X = \{a, b, c, d\}$ $Y = \{p, q, r, s\}$ together with edges

$$ap,\ aq,\ bq,\ br,\ cp,\ cr,\ cs,\ dr,\ ds.$$

A matching M in G is specified to consist of the edge subset $\{aq,\ br,\ cs\}$.

Specify an M-alternating forest in G and use it to produce a matching M' of greater cardinality than M. [6]

(c) In the bipartite graph shown below, the bold edges specify a matching.



Explain in detail:

(i) how Kőnig's bipartite matching algorithm will determine that this matching is maximum; [4]

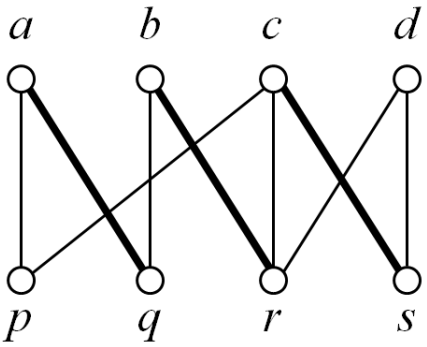
(ii) how the algorithm may also be used to identify a minimum cover of the graph. Specify the cover which will be identified. [3]

(d) Prove that, in any graph G, the size of a minimum cover is not less than the size of a maximum matching. [5]

SOLUTION:
Question is unseen but similar to coursework. Part (d) was given in lecture notes.

(a) A matching in a graph G is a subset of the edges of G no two of which share a common vertex **OR** it is a subgraph of G in which every vertex has degree at most 1. [2]

(b) A drawing like the following is expected:



© Queen Mary, University of London (2014)

Start with $F = \{d\}$ since d is the single M-unsaturated vertex in X.

Now repeatedly increment F by

(1) Adding non-M-edges leaving F from X;

(2) Adding M-edges leaving F from Y.

Thus: ds, dr, sc, rb, cp, bq, qa. [3]

[Not unique - similar sequences gain equal marks. A single drawing showing the final forest is also sufficient for full marks as shown below:]

Now chose an unsaturated Y vertex in F: p.

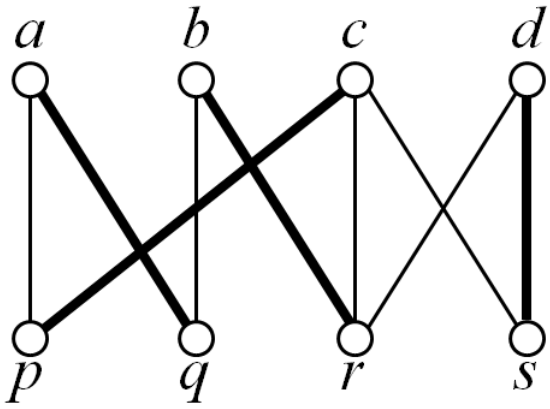
Find the unique path p to d in F: $pcsd$.

Reverse M and non-M edges along this path to get:

$$M' = \{aq, br, cp, ds\}.$$

[3]

[A drawing showing the final matching is also sufficient for full marks as shown below:]



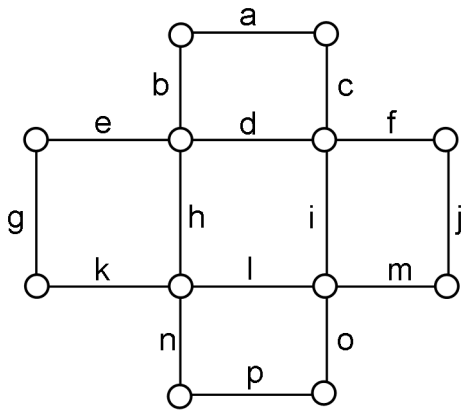
(c) (i) Kőnig's bipartite matching algorithm will construct a maximal M-unsaturated forest consisting of the four edges x_2y_5, x_2y_4, x_4y_4, x_5y_5. Since this forest does not contain an unsaturated vertex in Y the algorithm will terminate with the current matching declared maximum. [4]

(ii) In the maximal forest in part (a) is F then a minimum cover is $X \backslash F \cup (Y \cap F) = \{x_2, x_4, x_5\} \cup \{y_4, y_5\}$. [3]

(d) Let M be a matching and let C be a cover. Every edge of M must be incident with a vertex of C. But no vertex of C may be incident with more than one matching edge by definition of matching. So $|C| \geq |M|$. [5]

© Queen Mary, University of London (2014)

Question 5 A graph G is specified by the drawing shown below:



(a) Explain what it means to say that the graph G is *Eulerian*. [2]

(b) The Euler Tour Algorithm works by successively finding maximal closed trails. Explain what it means to say that $T = a,\ c,\ d,\ e,\ g,\ k,\ h, b$ is a maximal closed trail in G. [3]

(c) Suppose, continuing from the closed trail specified in part (b), the Euler Tour Algorithm identifies two further maximal closed trails: $T' = f,\ i,\ m,\ j$ and $T'' = l,\ o,\ p,\ n$. Explain how these trails are combined to form an Euler tour of G and specify an Euler tour which may result from this combination. [9]

(d) Suppose that edges h and m are deleted from G. Specify a minimum-cardinality subset of the remaining edges which could be doubled in order to again obtain an Eulerian graph. Hence give a minimum-length closed walk in G which does not use edges h and m but which passes every other edge. [6]

SOLUTION:
Question is unseen but similar to coursework. Parts (a) and (b) are book-work.

(a) The graph is Eulerian because all degrees are even; therefore, by Euler-Hierholzer, there exists an Euler tour: a closed trail passing every edge. [2]

[Any subset/permutation of the above which demonstrates a clear understanding of 'Eulerian' is OK.]

(b) The trail is *maximal* because it cannot be extended: no edge remains unused which can be followed after edge b. It is *closed* because it has returned to its initial vertex. [3]

(c) Closed trails are 'glued' together by identifying a common vertex and then interpolating one trail into the other at this point. Common to T and T' is the vertex incident with edges c, d, f and i. Insert T', suitably rotated, to come between c and d in T: [5]

$$a\ c\ (i\ m\ j\ f)\ d\ e\ g\ k\ h\ b.$$

Now insert T'', suitably rotated, between edges i and m of T' **or** between edges k and h of T. This gives

$$a\ c\ (i\ (o\ p\ n\ l)\ m\ j\ f)\ d\ e\ g\ k\ h\ b$$

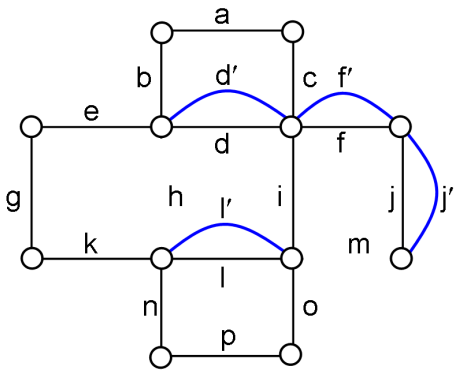
or

$$a\ c\ (i\ m\ j\ f)\ d\ e\ g\ k\ (l\ o\ p\ n)\ h\ b$$

respectively. [4]

© Queen Mary, University of London (2014)

(d) The edges to double are d, f, j and l, as shown below (the drawing is not necessary but may help mak [2]



An Euler tour of the resulting graph is:

$a\,c\,f\,j\,j'\,f'\,d'\,d\,i\,l\,l'\,o\,p\,n\,k\,g\,e\,b$ giving closed walk $a\,c\,f\,j\,j\,f\,d\,d\,i\,l\,l\,o\,p\,n\,k\,g\,e\,b$

[4]

[There are many variations: give the benefit of the doubt unless obviously wrong!]

End of Paper

© Queen Mary, University of London (2014)