Queen Mary
University of London

B. Sc. Examination by course unit 2014

MTH6105 Algorithmic Graph Theory

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.

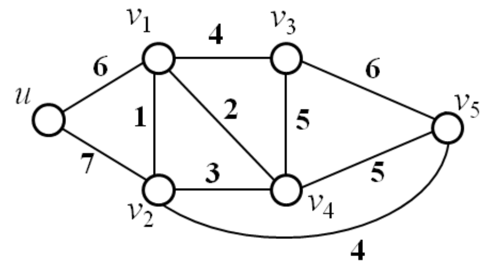
Important note: the Academic Regulations state that possession of unauthorized material at any time by a student who is under examination conditions is an assessment offence and can lead to expulsion from QMUL.

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

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]

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

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

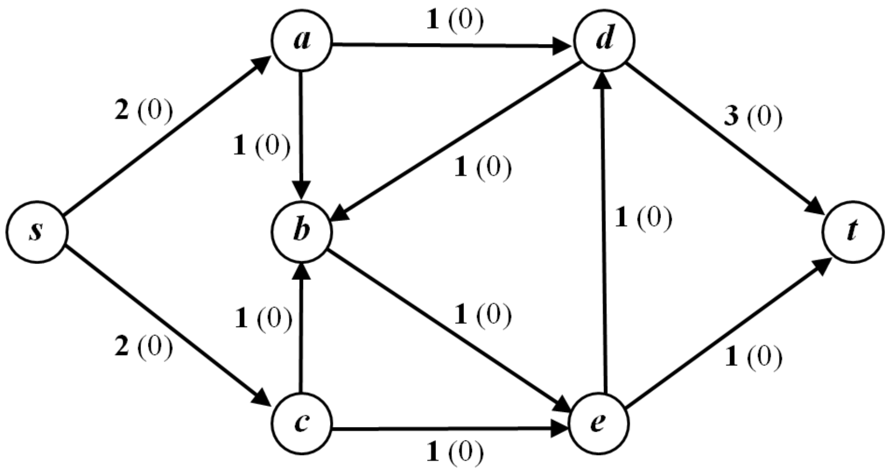
together with a list of weighted edges (the first entry in each pair is the edge, the second entry is its weight):

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

(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]

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).



(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]

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

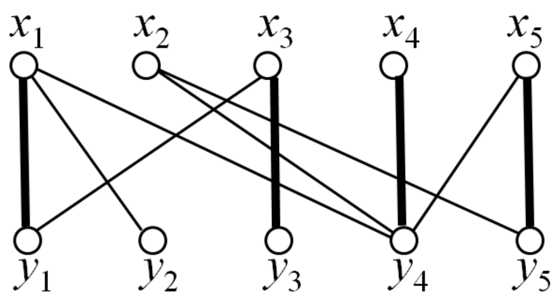
(b) A bipartite graph G is specified as consisting of a partitioned vertex set $V = X \cup Y,\ X = \{a, b, c, d\},\ Y = \{p, q, r, s\}$ together with edges

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

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.



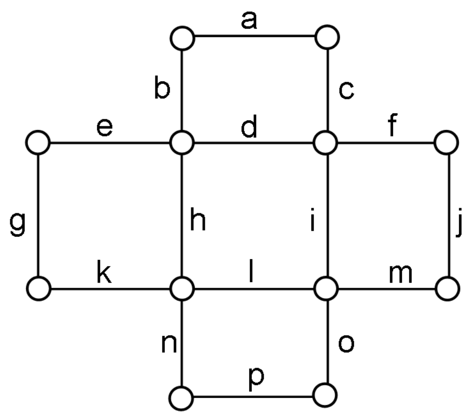
Explain in detail:

(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]

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]

End of Paper