Australian National University
Department of Computer Science

Computer Science COMP4600 2014 – Assignment Two

Part I: Network Flow

Due: Sunday 21 September
Late Penalty: 10% per day

Part II: Complexity Theory of this assignment should be ready around the third week in August.
It will also be due Sunday 21 September.

No programming is needed for this assignment. *Be precise. Avoid long-winded answers. Answer clearly and legibly. Indicate clearly the question and part to which you are providing an answer for.*

Submit a typed or scanned copy of your assignment by email to `paulette.lieby@anu.edu.au`, or a hard copy to Paulette Lieby, room N222 (under the door is ok), CSIT Building 108 ANU. You may also drop your hard copy into the COMP4600 assignment box on level 1 in the CSIT Building.

The total mark of this assignment is 100 points.
Don't forget to add your name/student ID.

Neat handwriting is acceptable. You may use the accompanying worksheets for help in performing the network flow algorithms.

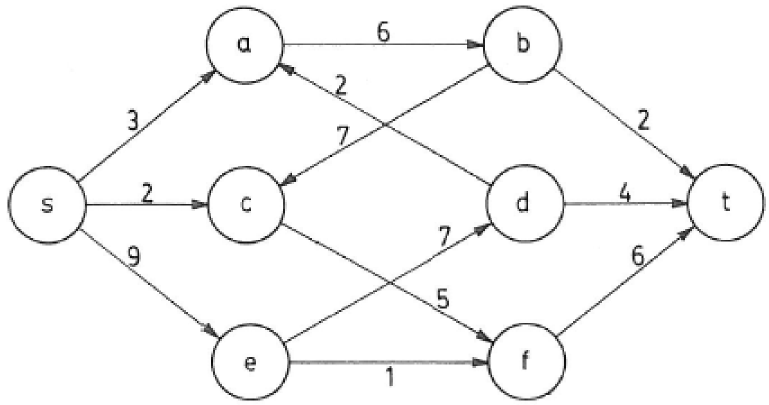


Figure 1: Flow network G for Question 1.

Question 1 (30/100).

1. Perform Edmonds-Karp's algorithm on graph G in Figure 1. For each step, show
   - the residual network G_f and the capacities of its edges
   - the chosen augmenting path and its residual capacity
   - the augmented flow

   Give the value of the maximum flow and show a minimum cut.

2. Perform the generic push-relabel algorithm on graph G in Figure 1. For each step, show
   - the residual network G_f and the capacities of its edges
   - the height and excess for each vertex
   - qualify the step (push or relabel)

   Give the value of the maximum flow and show a minimum cut.

Question 2 (20/100).

Let $G = (V, E)$ be a network flow with capacity function c and flow f. Assume there exists an augmenting path P in G_f resulting in the new flow f' after augmentation.

1. Show that f' is indeed a flow.
2. Explain how an edge (u, v) might be in $G_{f'}$ even though it is not in G_f.

Question 3 (20/100).

Let $G = (V, E)$ be a network flow with preflow f and height function h. Assume that in `Initialise Preflow`, the vertex height is initialised as

for `each` $u \in V$
    $h(u) = 0$
$h(s) = |V| - 2$

Show that Lemma 8 in the slides is still valid, thus ensuring the correctness of the push-relabel algorithm under this new definition of the height function.

Question 4 (20/100).

1. Show that at each point of the execution of a push-relabel algorithm, there exists at least one integer $0 < k \leq |V| + 1$ for which no vertex u has height $h(u) = k$.
2. Suppose that we have found a maximum flow in a flow network $G = (V, E)$ using a push relabel algorithm. Use the fact in 1. to give an simple algorithm to find a minimum cut.

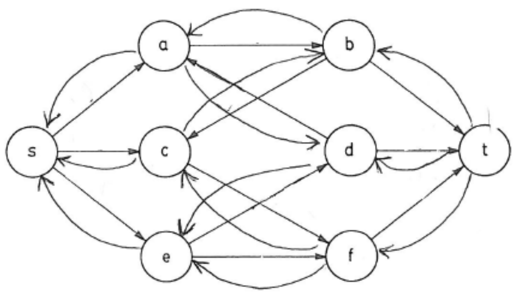
Question 5 (10/100).

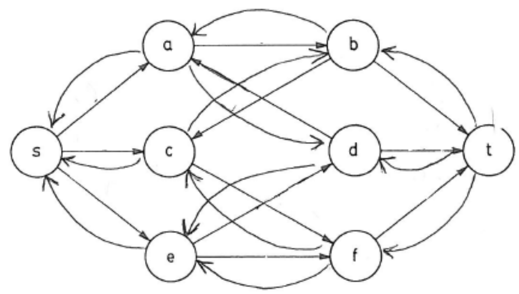
Suppose that a flow network $G = (V, E)$ violates the assumption that the network contains a path $s \rightsquigarrow v \rightsquigarrow t$ for all vertices $v \in V$. Let u be a vertex for which there is no path $s \rightsquigarrow u \rightsquigarrow t$. Show that there must exist a maximum flow f in G such that $f(u, v) = f(v, u) = 0$ for all vertices $v \in V$.
Hint: think of augmenting paths; your answer should be one sentence long...

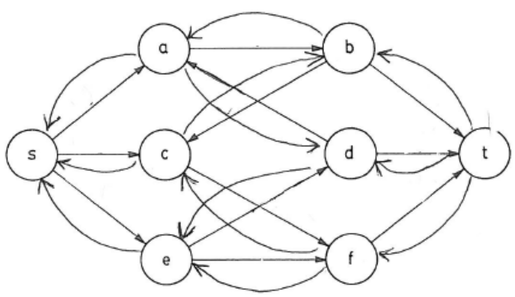
COMP4600, 2014 Assignment Two

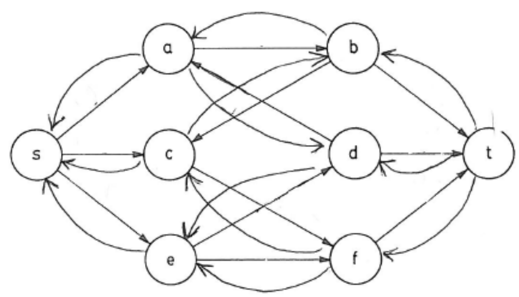
Worksheets for Question 1

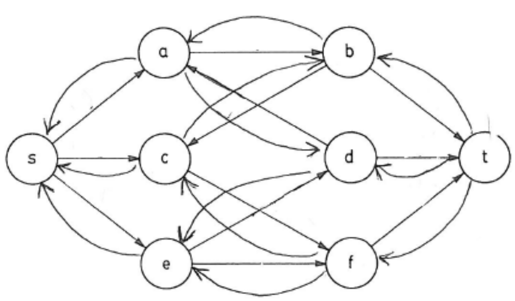
Use as many sheets as necessary. Number them and indicate which question is answered.

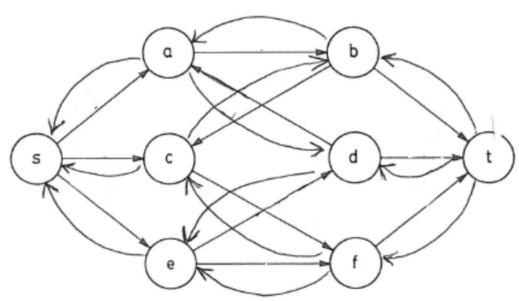


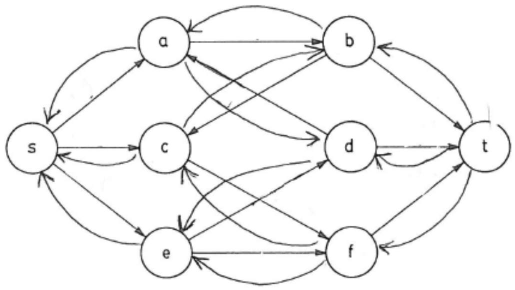
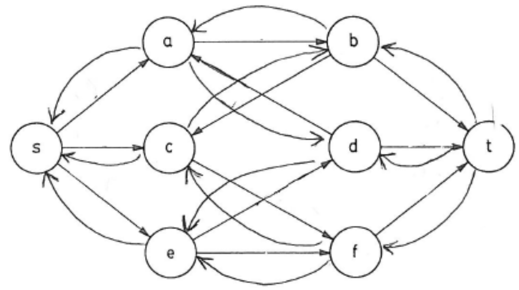
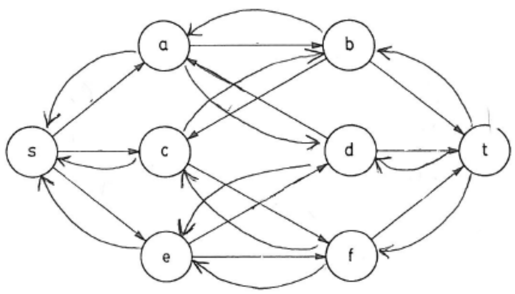
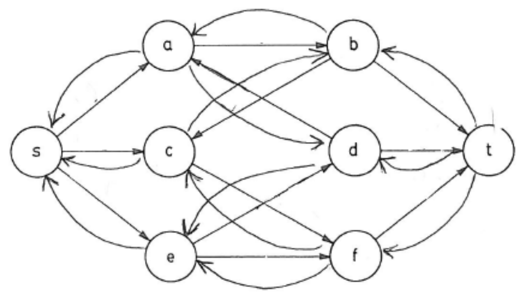
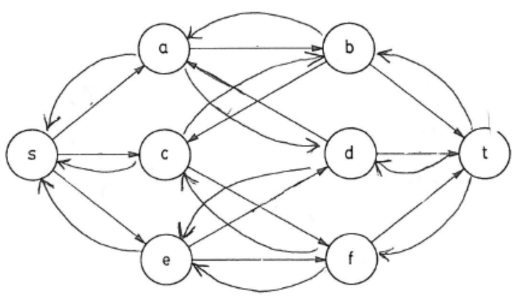
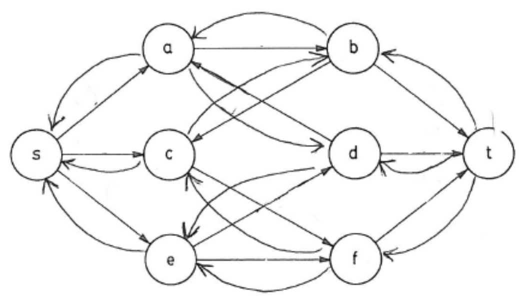
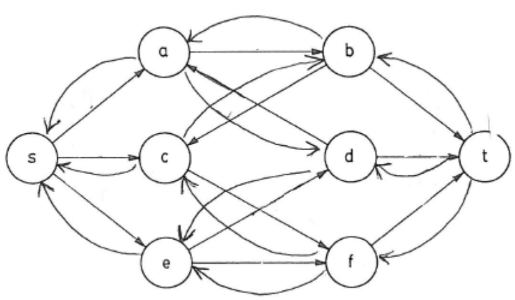
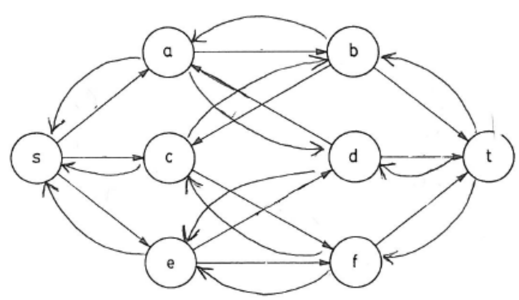


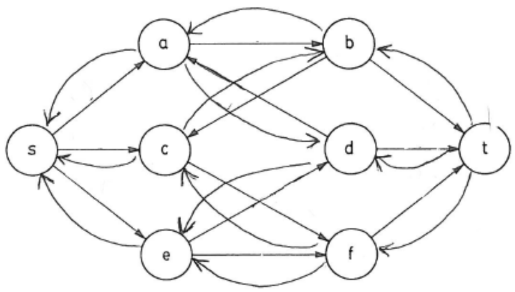


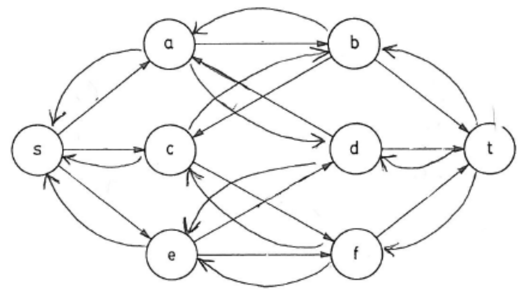


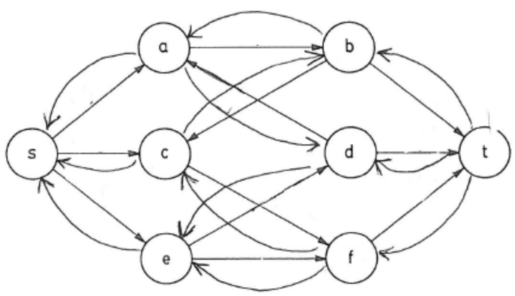


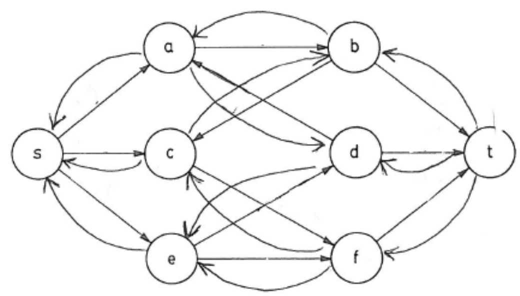


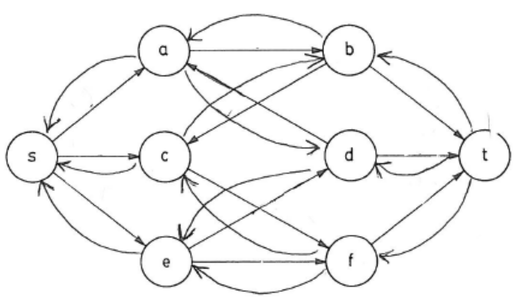

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

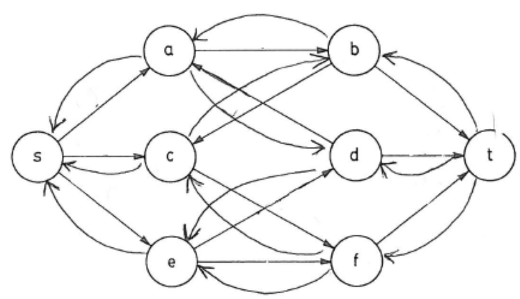

a
b
s
c
d
t
e
f

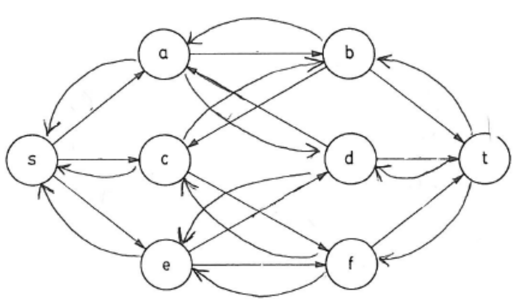

a
b
s
c
d
t
e
f

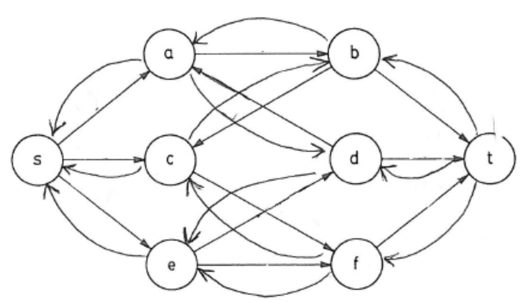

a
b
s
c
d
t
e
f

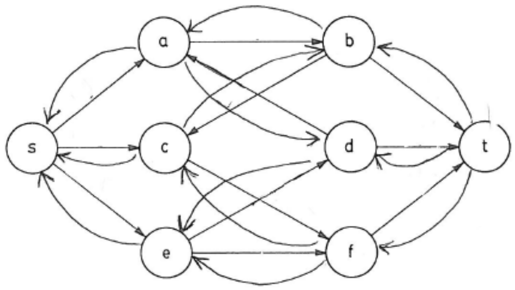

a
b
s
c
d
t
e
f

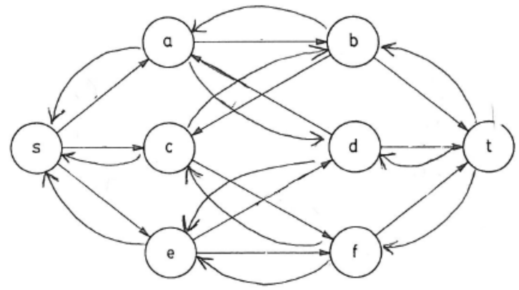

a
b
s
c
d
t
e
f

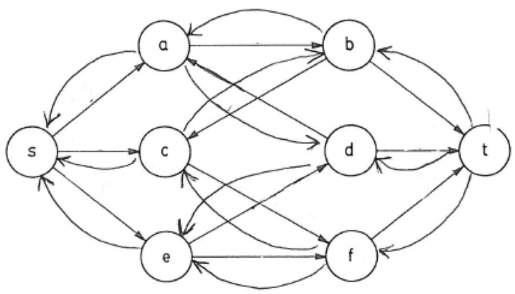

a
b
s
c
d
t
e
f

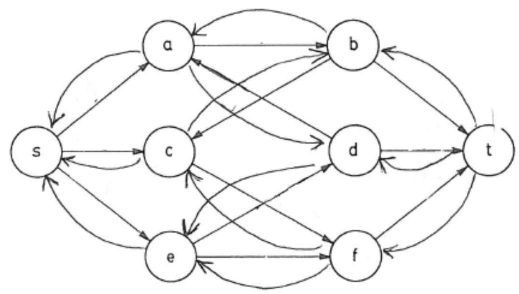

a
b
s
c
d
t
e
f

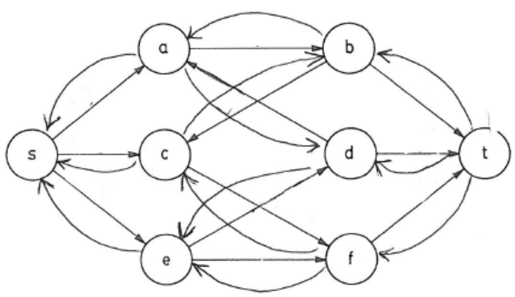

a
b
s
c
d
t
e
f

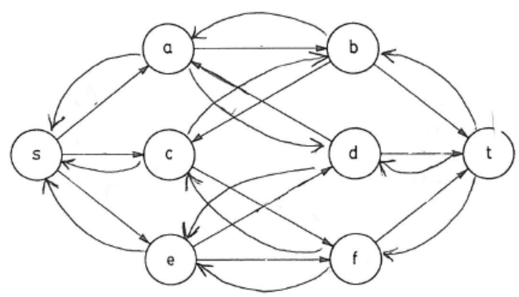

a
b
s
c
d
t
e
f

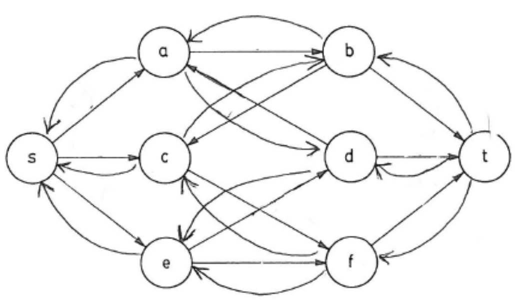

a
b
s
c
d
t
e
f

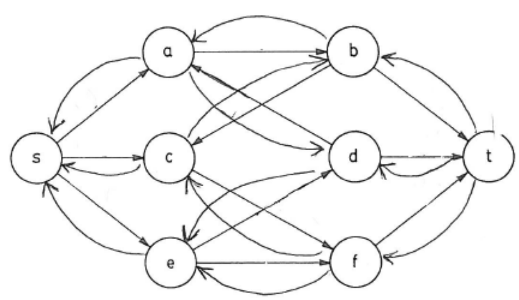

a
b
s
c
d
t
e
f

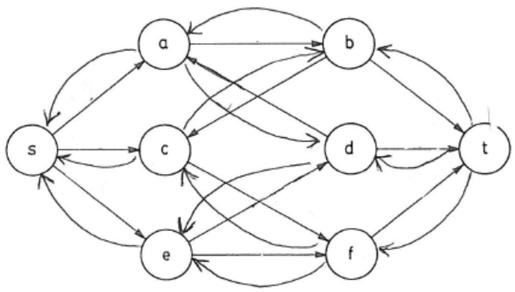
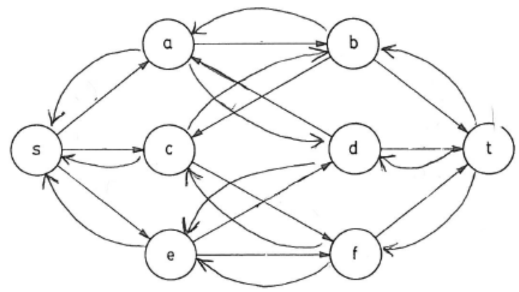
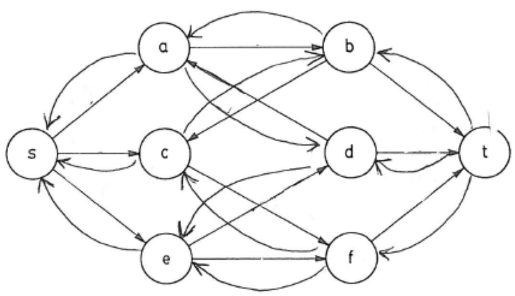
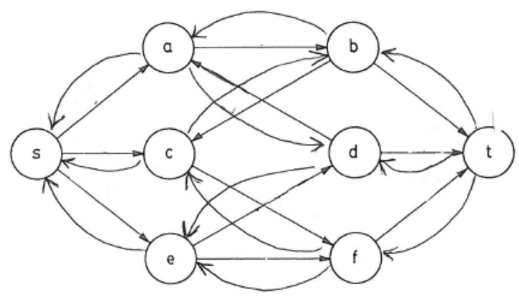
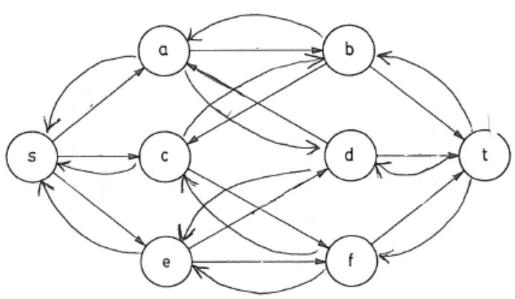
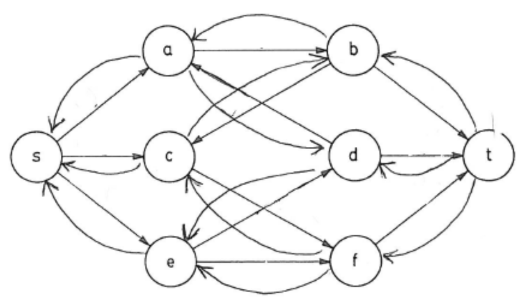
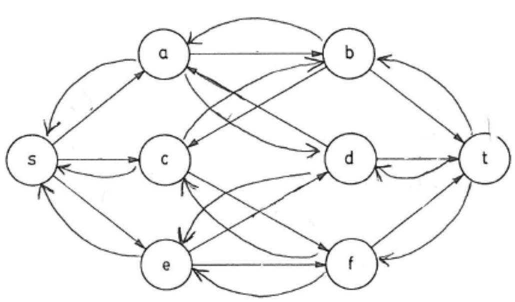
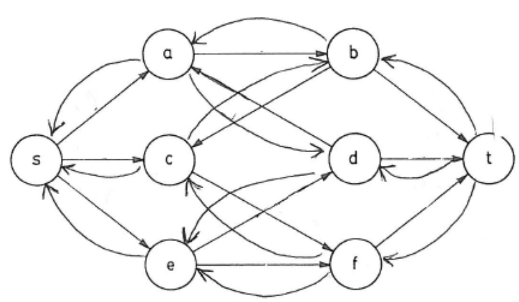

a
b
s
c
d
t
e
f

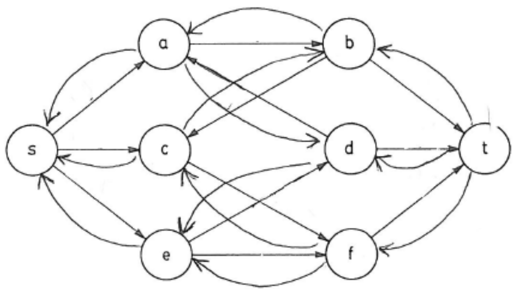
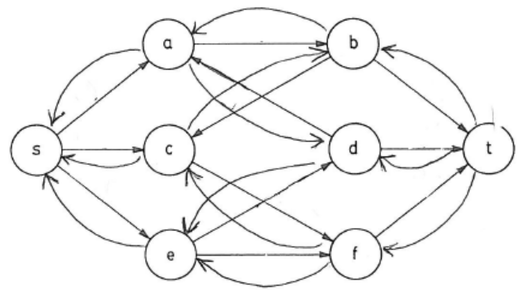
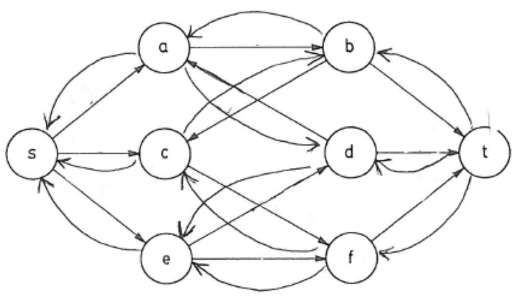
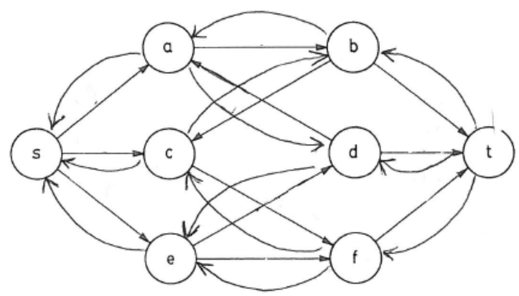
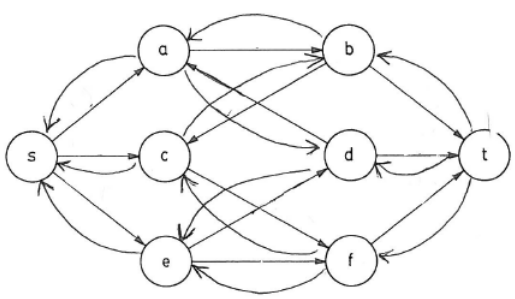
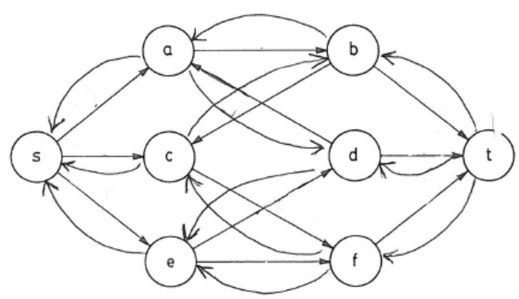
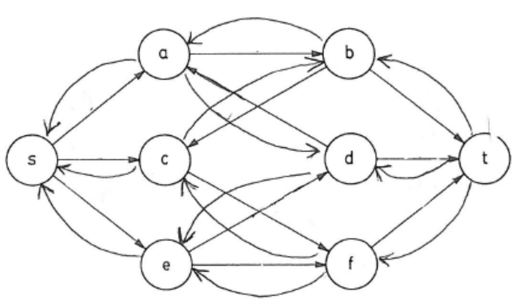
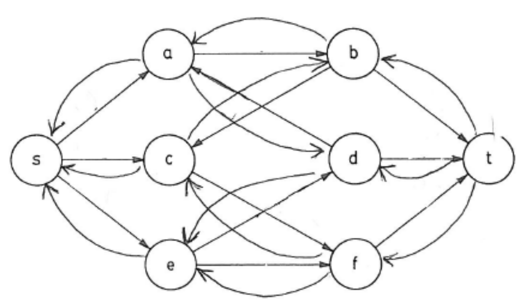

a
b
s
c
d
t
e
f

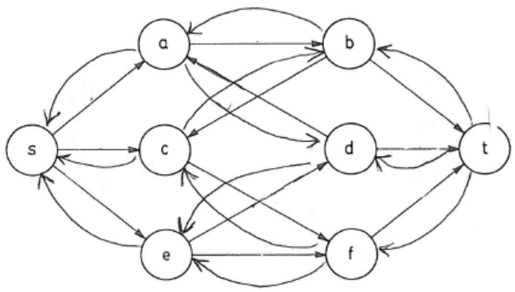

a
b
s
c
d
t
e
f

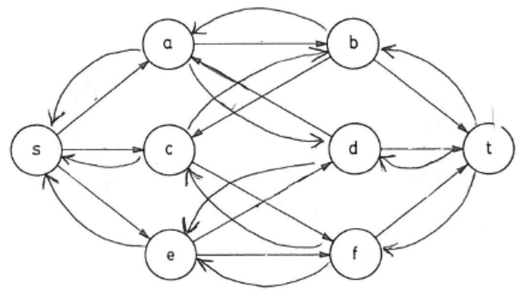

a
b
s
c
d
t
e
f

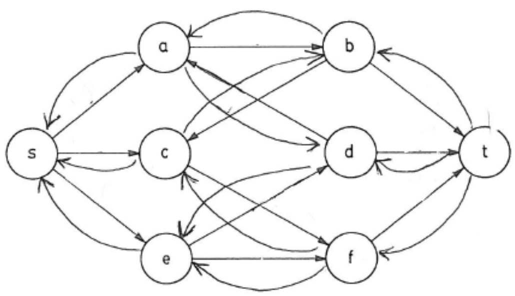

a
b
s
c
d
t
e
f

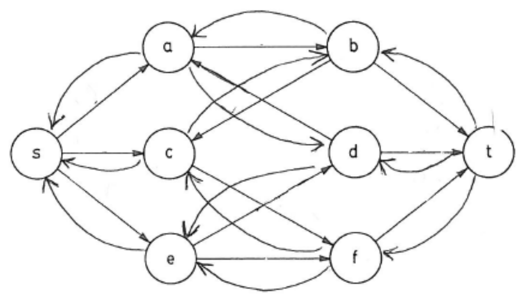

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

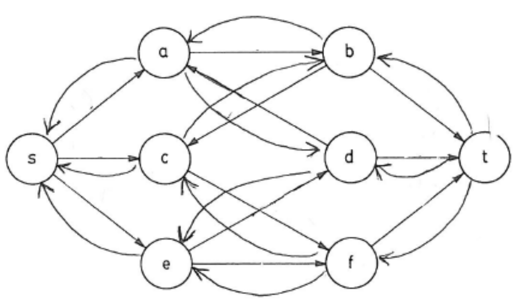

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

a
b
s
c
d
t
e
f

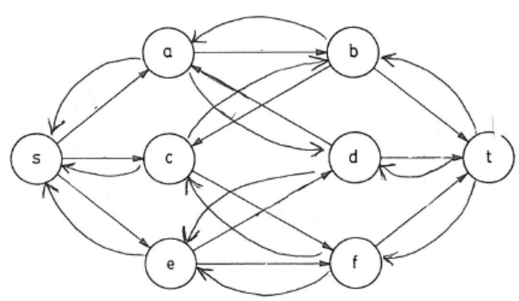

a
b
s
c
d
t
e
f

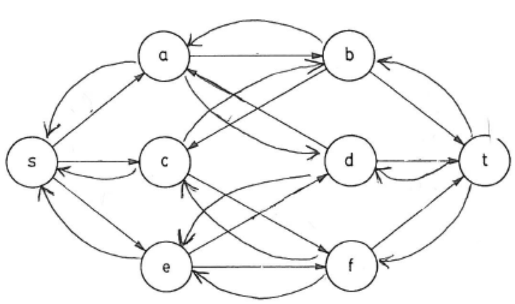

a
b
s
c
d
t
e
f

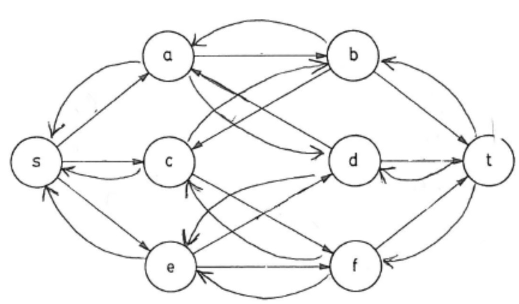

a
b
s
c
d
t
e
f


a
b
s
c
d
t
e
f


a
b
s
c
d
t
e
f


a
b
s
c
d
t
e
f


a
b
s
c
d
t
e
f


a
b
s
c
d
t
e
f