Adrian Willi

Exercise 11

Task 1

a) Given the adjacency list:

a	b	c	d
b	a	f	
c	d		
d	e	f	
e	a		
f	e		

Draw the directed graph for the above adjacency list.

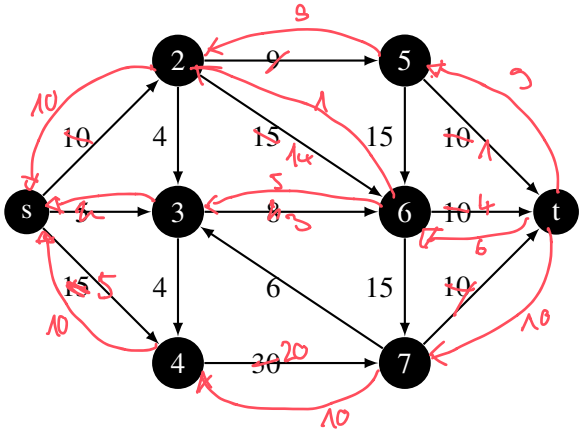
b) Given the adjacency matrix:

	a	b	c	d	e	f	g
a	0	1	0	1	0	0	0
b	0	0	1	0	0	1	0
c	0	0	0	0	1	0	0
d	0	1	0	0	0	0	0
e	0	0	0	0	0	0	1
f	1	0	0	1	0	0	0
g	0	0	1	0	0	0	0

Draw the directed graph for the above adjacency matrix and perform a depth-first search and a breadth-first search starting from a. (Give the vertices in the sequence they are encountered if you process the neighbors of each vertex in alphabetical order.)

Task 2

Calculate the maximum flow in the following s-t graph with the capacities shown.

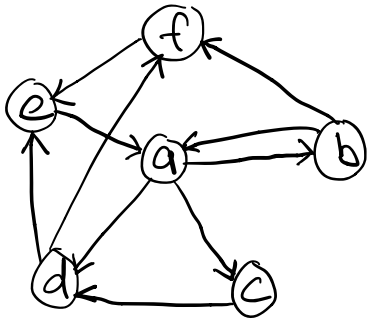


10 s → 4 → 7 → t
5 s → 3 → 6 → t
5 s → 2 → 5 → t
1 s → 2 → 6 → t
3 s → 4 → 7 → 3 → 6 → t
28

The maximum flow is 28.

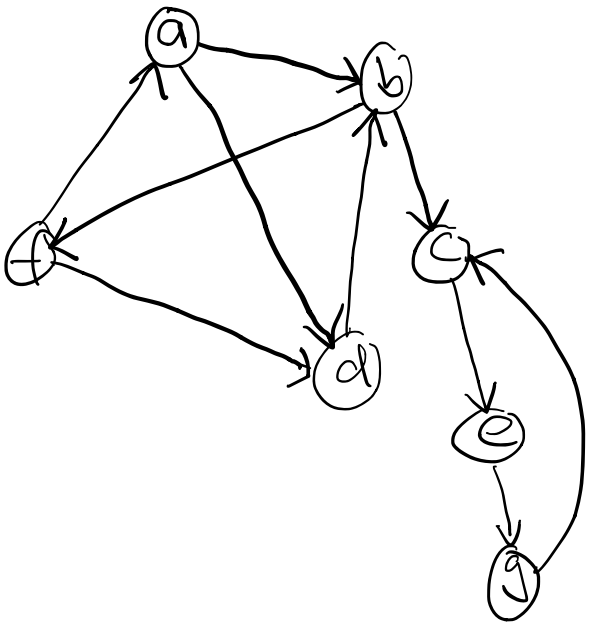
Task 1

a)



b)

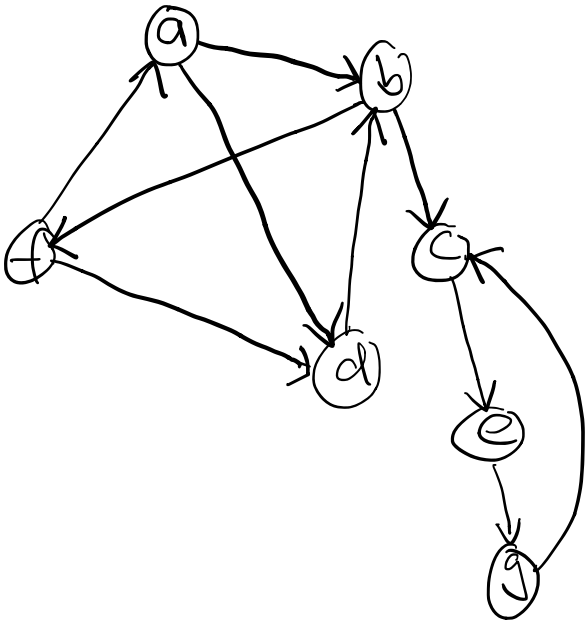
DFS



~~d~~ 4
~~f~~ 5
~~g~~ 1
~~e~~ 2
~~c~~ 3
~~b~~ 6
~~a~~ 7

g, e, c, d, f, b, a

BFS



~~a~~ ~~b~~ ~~d~~ ~~c~~ ~~f~~ ~~e~~ ~~g~~

a, b, d, c, f, e, g

Task 3

a) Calculate a maximum matching in the bipartite graph with the following adjacency matrix.

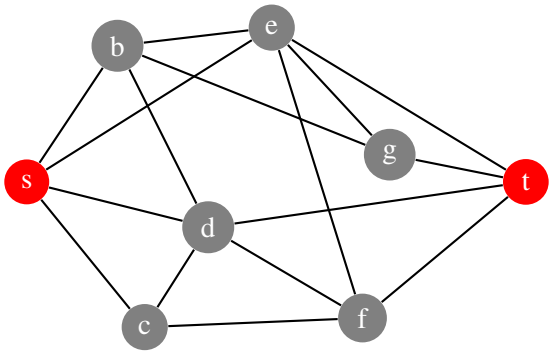
	a1	a2	a3	a4	a5	b1	b2	b3	b4	b5
a1	0	0	0	0	0	1	0	1	0	0
a2	0	0	0	0	0	1	0	0	0	0
a3	0	0	0	0	0	0	1	0	0	1
a4	0	0	0	0	0	0	0	1	0	0
a5	0	0	0	0	0	0	0	0	1	0
b1	1	1	0	0	0	0	0	0	0	0
b2	0	0	1	0	0	0	0	0	0	0
b3	1	0	0	1	0	0	0	0	0	0
b4	0	0	0	0	1	0	0	0	0	0
b5	0	0	1	0	0	0	0	0	0	0

b) Find a minimum s-t cut in the network you used to solve a).

Hint: This is surprisingly hard to see „by eye“, even in this fairly small example. Fortunately, we can use following general criterion: A minimum cut is obtained for S as *the set of all vertices reachable from s in the residual graph* (including backwards edges), and $T = V \setminus S$.

Task 4

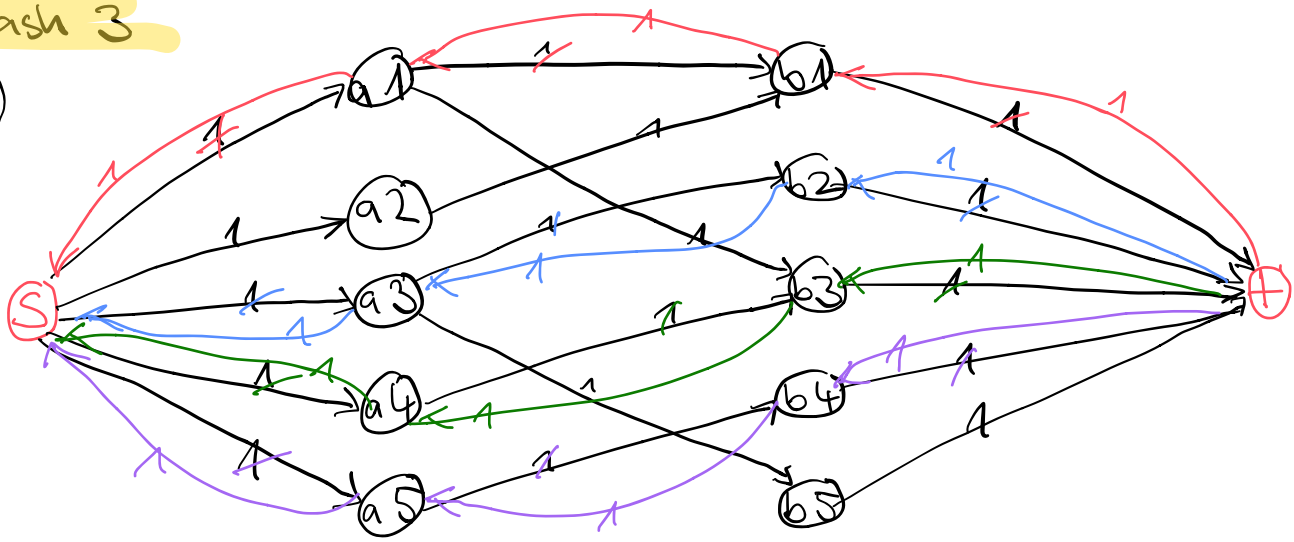
How many connections may fail without interrupting the communication between s and t?



Hint: Replace every undirected edge (connection) by two directed edges in opposite directions, each with capacity 1.

Task 3

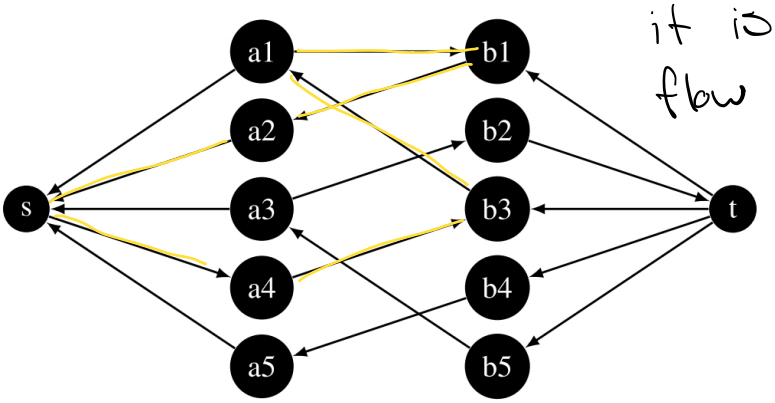
a)



1 s → a1 → b1 → t
1 s → a3 → b2 → t
1 s → a4 → b3 → t
1 s → a5 → b4 → t
4

We find a maximum of 4 pairs:
(a1, b1), (a3, b2), (a4, b3), (a5, b4)

b)



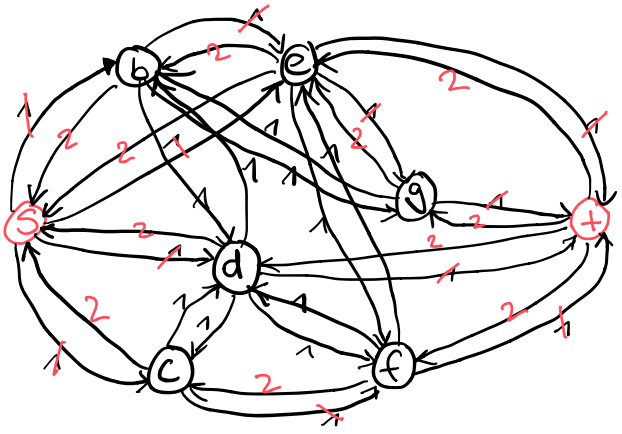
For my solution above it is the same but the flow through the path is just the other way around.

$S = \{s, a1, a2, a4, b1, b3\}$

$T = V \setminus S$

The cut edges are (s, a3), (s, a5), (t, b2), (t, b4), (t, b5)

Task 4

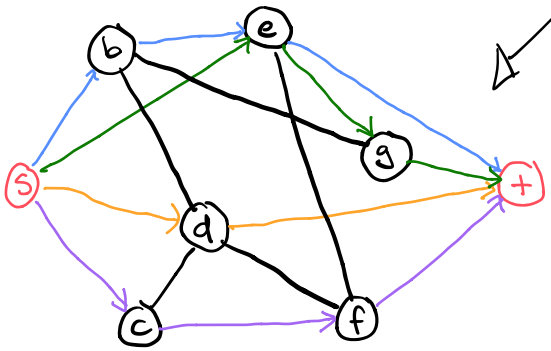


1 S → b → e → t

1 s → c → f → t

1 s → d → t

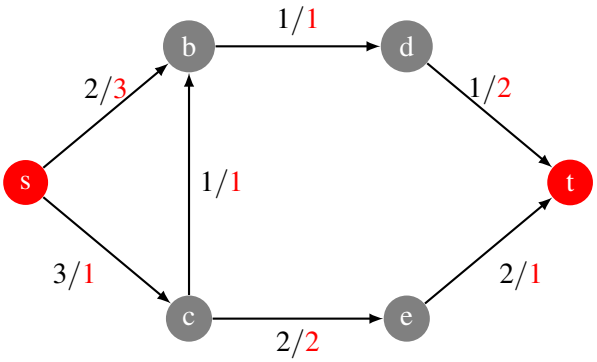
1 s → e → g → t



As all edge capacities are 1, this flow must be realized by 4-edge-disjoint paths from s to t (see below). According to the min-cut max-flow theorem the minimum cut needed to seperate s from t is also 4. Thus any 3 connections can be omitted without interrupting the connection from s to t.

Task 5

Calculate the min-cost max-flow in the following network. Capacities are given in black, costs in red.



Begin with the path $s \to b \to d \to t$.

Task 6

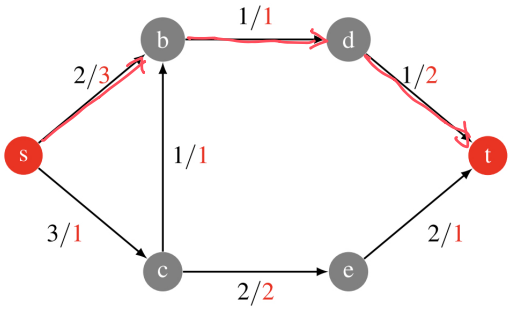
Dijkstra is slower because it searches the best connections to all locations. A* is an optimization to find the path to a single destination. It combines Greedy Best-First search and Dijkstra!

Visit the following website and read about the disadvantages of the Dijkstra algorithm and how it can be improved with the A* algorithm:
`http://theory.stanford.edu/~amitp/GameProgramming/AStarComparison.html`

Task 7 (*optional)

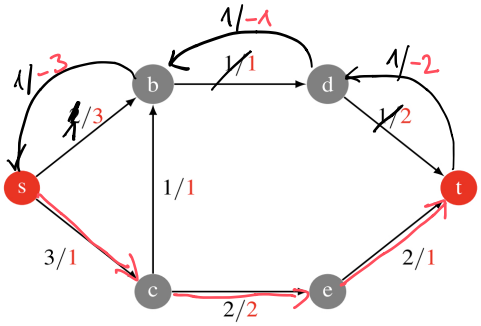
Implement a program that reads in a weighted adjacency matrix and calculates the maximum flow in the network using the Ford-Fulkerson algorithm.

Task 5



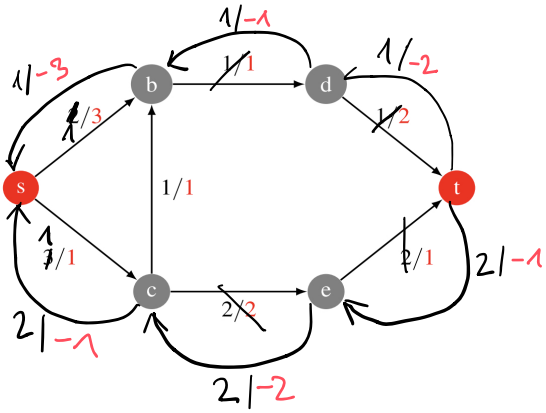
Begin with path $s \to b \to d \to t$

are black / capacity	path	are red / cost
1	$s \to b \to d \to t$	$1 \cdot 6 = 6$
2	$s \to c \to e \to t$	$2 \cdot 4 = 8$
3		14



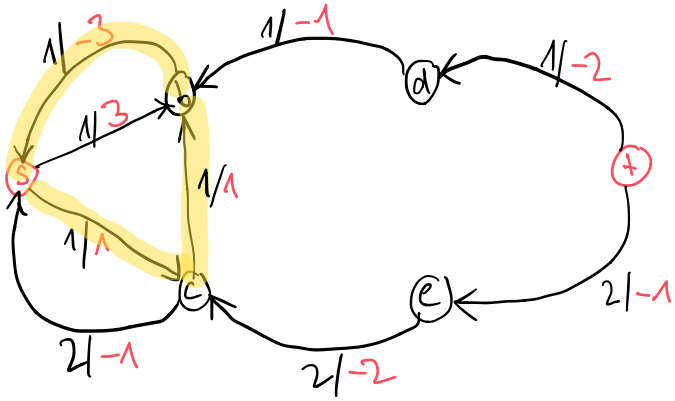
1 — $s \to c \to b \to s$ — $1 \cdot (-1) = -1$

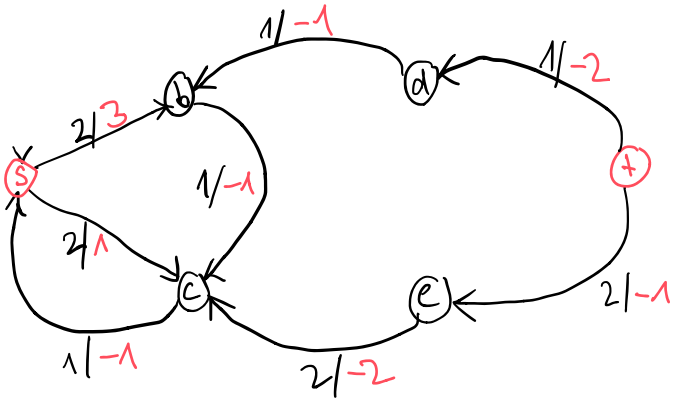
4 — 13



Till here it is just Ford-Fulkerson

Now we look for cycles with negative cost. We find $s \to c \to b \to s$ with capacity 1 and negative cost of -1.





Final graph. Remaining cycles are have not negative costs!

Solutions to Exercise 11

Solution to Task 1

a) —

b) Output order for DFS: g e c d f b a
(Stack: $a_\downarrow\ b_\downarrow\ c_\downarrow\ e_\downarrow\ g_\downarrow\ g_\uparrow\ e_\uparrow\ c_\uparrow\ f_\downarrow\ d_\downarrow\ d_\uparrow\ f_\uparrow\ b_\uparrow\ a_\uparrow$),
Output order for BFS: a b d c f e g (this is also how the queue looks like).

To display the graphs and the search results you can also use the corresponding R script (for checking).

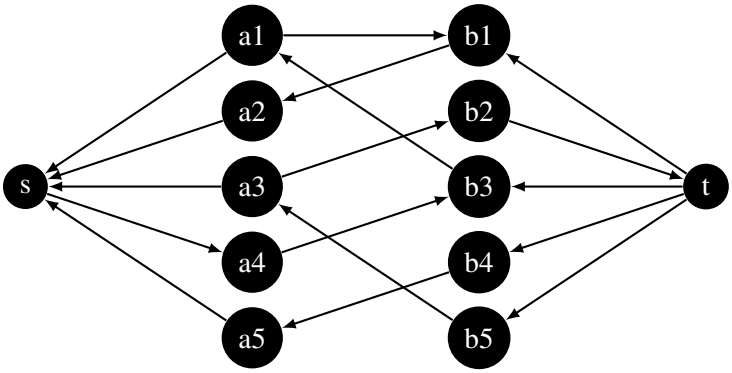
Solution to Task 2

The maximum flow is 28. This is obtained e.g. with:
$s \to 4 \to 7 \to t$ with 10.
$s \to 2 \to 5 \to t$ with 9.
$s \to 3 \to 6 \to t$ with 5.
$s \to 4 \to 7 \to 3 \to 6 \to t$ with 3.
$s \to 2 \to 6 \to t$ with 1.
If we start e.g. with $s \to 2 \to 6 \to t$ with capacity 10, backwards edges need to be used.

Solution to Task 3

a) Draw the bipartite graph with edges as specified, and add a start node s with edges to the nodes $a1, a2, \ldots, a5$ and an end node t with edges from the nodes $b1, b2, \ldots, b5$. Each edge has capacity 1. We can now use the Ford-Fulkerson algorithm to calculate an (integer) maximum flow in this network, which corresponds to a maximum matching in the bipartite graph. We find a maximum of 4 pairs, e.g. $(a3, b5), (a5, b4), (a1, b3), (a2, b1)$.

b) Considering the residual graph corresponding to the matching given above (see picture), we obtain the cut $S = \{s, a1, a2, a4, b1, b3\}$, $T = V \setminus S$. The cut edges are $(s, a3)$, $(s, a5)$, $(b1, t)$, $(b3, t)$.



Solution to Task 4

The maximum flow in the graph is equal to 4 (Ford-Fulkerson) when all edge capacities are 1. (Replace the undirected edges by directed edges in both directions.) As all edge capacities are 1, this flow must be realized by 4 edge-disjoint paths from s to t. (Draw them!) According to the min-cut max-flow theorem the minimum cut needed to separate s from t is also 4. Thus any 3 connections can be omitted without interrupting the connection from s to t.

Solution to Task 5

The maximum flow with minimum cost has capacity 3 and flows as follows:
$s \to c \to e \to t$ with capacity 2.
$s \to c \to b \to d \to t$ with capacity 1.
Its cost is $2 \cdot 4 + 5 = 13$.

If we start with the path $s \to b \to d \to t$ (capacity is 1, cost $1 \cdot 6 = 6$),
the next path is $s \to c \to e \to t$ (capacity is 2, cost $2 \cdot 4 = 8$).
In the residual network there are no further paths from s to t, so we have found the maximum possible flow.

There is still the cycle $s \to c \to b \to s$ with negative cost -1 and capacity 1. Resolving this cycle reduces the cost from $1 \cdot 6 + 2 \cdot 4 = 14$ to 13. (This corresponds to rerouting one unit of flow from $s \to b \to c$ to the direct edge $s \to c$.)

As now there are no further negative cost-cycles in the residual graph, we have found the desired cost-optimal maximum flow.

Solution to Task 7

See R solutions.