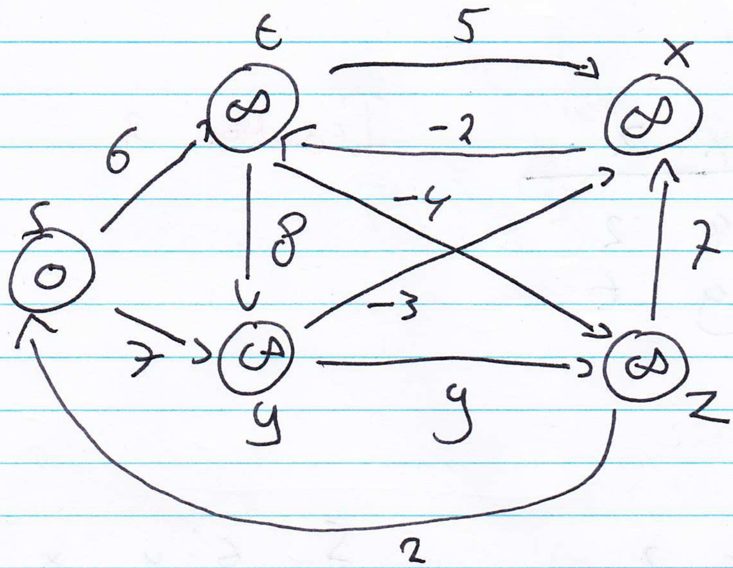
V Algoritmen en Datastructuren

1.



2. We have five vertices so 4 iterations will take place.

1.

	s	t	y	x	z
d	0	∞	∞	∞	∞
π	/	/	/	/	/

Initialisation

2.

	s	t	y	x	z
d	0	6	7	∞	∞
π	/	s	s	/	/

Iteration 1

3.

	s	t	y	x	z
d				~~11~~ 4	2
π				~~t~~	t

3.

	s	t	y	x	z
d:	0	6	7	∞	∞
				11	2
				4	

	s	t	y	x	z
π:	/	s	s	/	/
				t	t
				y	

	s	t	y	x	z
d:	0	6	7	4	2
π:	/	s	s	y	t

Iteration 2

4.

	s	t	y	x	z
d:	0	6	7	4	2
		2			

	s	t	y	x	z
π:	/	s	s	y	t
		x			

	s	t	y	x	z
d:	0	2	7	4	2
π:	/	x	s	y	t

Iteration 3

5.

	s	t	y	x	z
d:	0	2	7	4	2
					-2

	s	t	y	x	z
π:	/	x	s	y	t
					t

	s	t	y	x	z
d:	0	2	7	4	-2
π:	/	x	s	y	t

Iteration 4.

Check:

Nothing changes after all iterations so there is no negative weight cycle.

2.

a. The exchange rate $a_1 \rightsquigarrow a_k$ is equal to:

$$c(a_1, a_k) = c(a_1, a_2) \cdot c(a_2, a_3) \cdot \ldots \cdot c(a_{k-1}, a_k)$$

b. This is the case ~~where~~ if the graph contains a cycle ~~with a positive~~ elem. $a_1 \rightsquigarrow a_1$ with $c(a_1, a_1) > 1$

c. Relaxation condition

~~if(d(A,B) < d(A,C) · d(C,B)) then d(A,B)~~

$$\text{if } (\, d[B] < d[C] \cdot c(C,B)\,) \text{ then } d[B] \leftarrow d[C] \cdot c(C,B);$$

d.

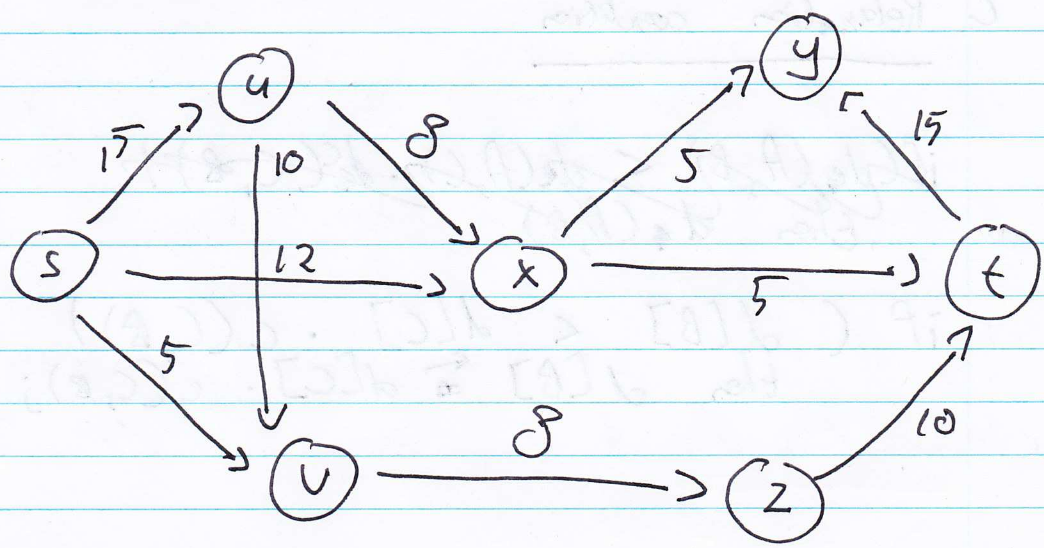
$d[s] \leftarrow 1$

for each $v \in V - \{s\}$
  do $d[v] \leftarrow 0$

for $i \leftarrow 1$ to $|V| - 1$
  do for each edge $(u, v) \in E$
    do if $d[v] < d[u] \cdot c(u, v)$
      $d[v] \leftarrow d[u] \cdot c(u, v)$

for each edge $(u, v) \in E$
  do if $d[v] < d[u] \cdot c(u, v)$
    then report a positive-weight cycle exists

3.



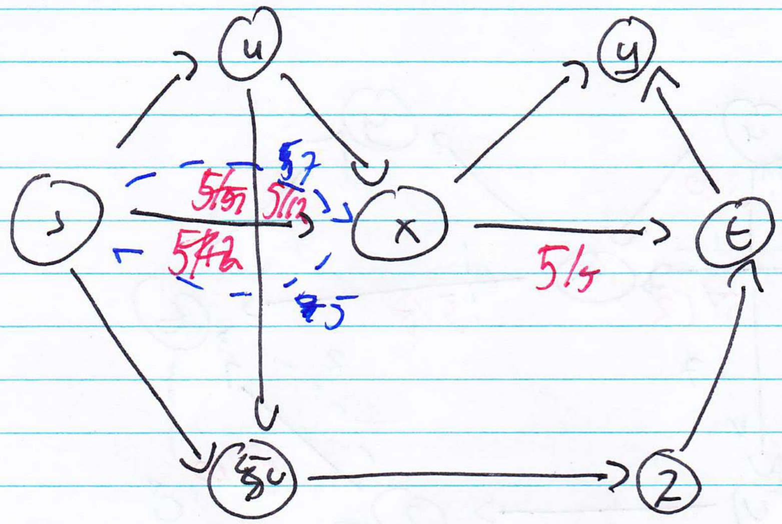
1.

	s	u	x	y	t	v	z
flow:	0	0	0	0	0	0	0

Initialisation

Iteration 1

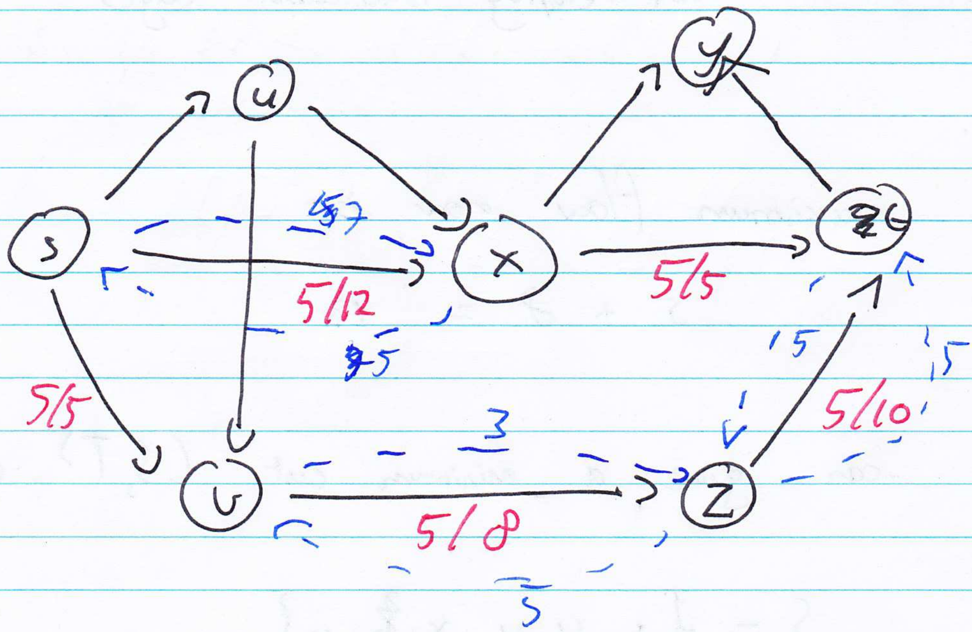
We first take path: s ⟶ x ⟶ t



	s	u	x	y	t	v	z
Flow	~~5~~0	0	5	0	5	0	0

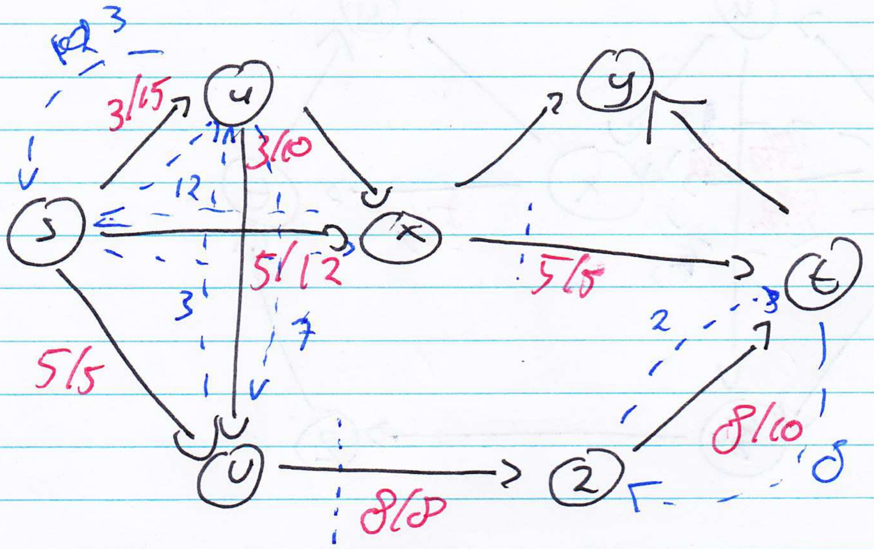
Iteration 2

We take shortest path: s ⟶ v ⟶ Z ⟶ t



Iteration 3

We take shortest path : $s \to u \to v \to z \to t$



Now we are done because we can't find a path with only using:

- ~~neo~~ non-full forward edges
- non-empty backward edges

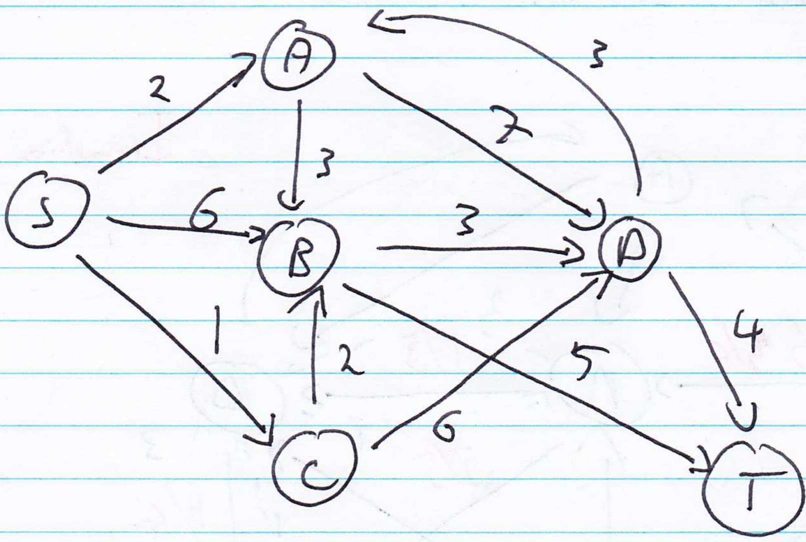
The maximum flow now is:

$$5 + 8 = 13$$

We can give a minimum cut (S, T) with:

$$S = \{s, u, v, x, \cancel{z}, y\}$$
$$T = \{\cancel{y}, t, z\}$$

4.



a. This problem corresponds to the max flow problem, we can use the Ford-Fulkerson algorithm

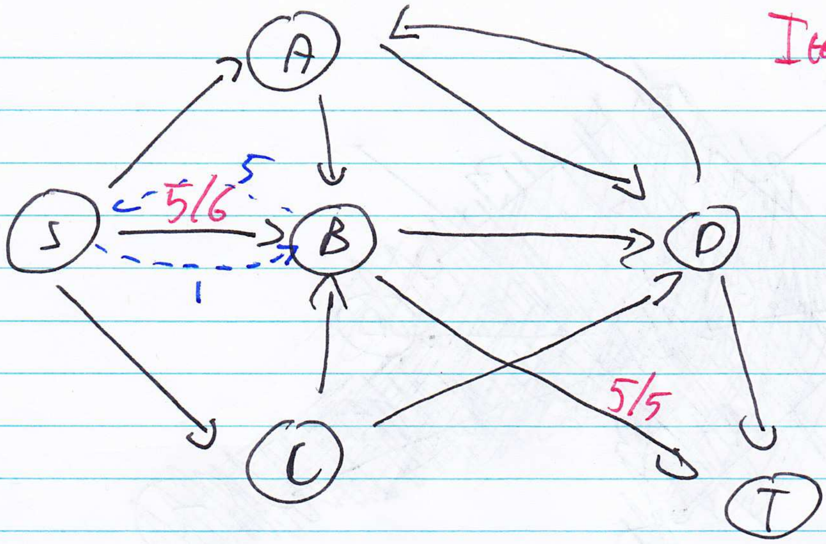
b.

S	A	B	C	D	T
0	0	0	0	0	0

Initialisation

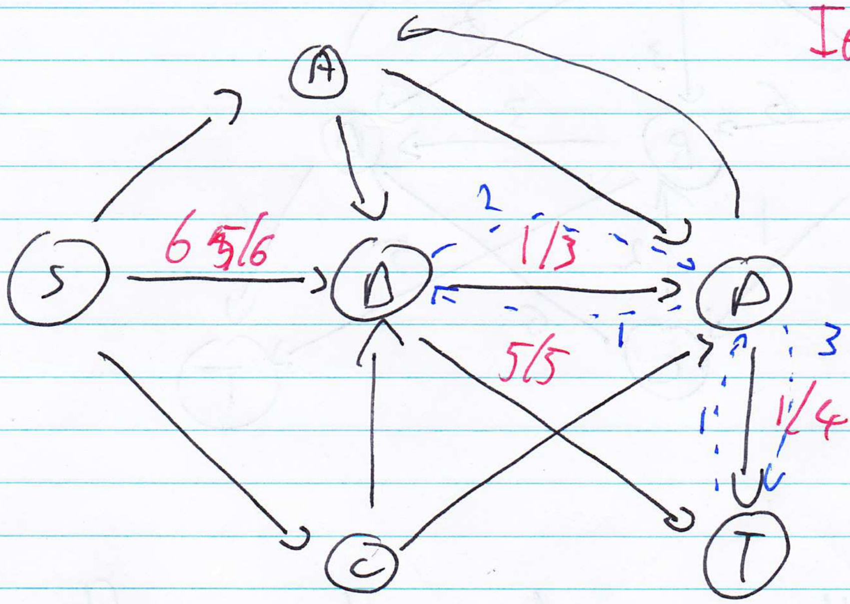
PB: ~~[illegible]~~

Iteration 1



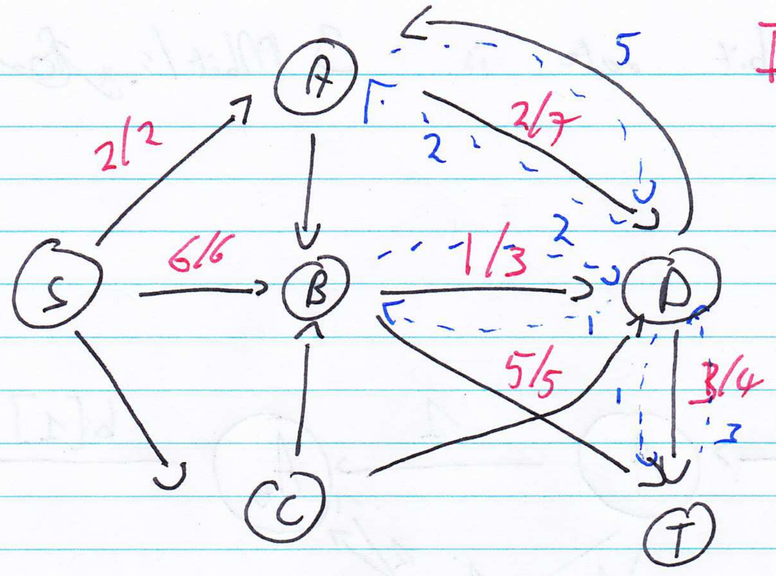
Shortest path: S ⟶ B ~~→ D →~~ → T

Iteration 2



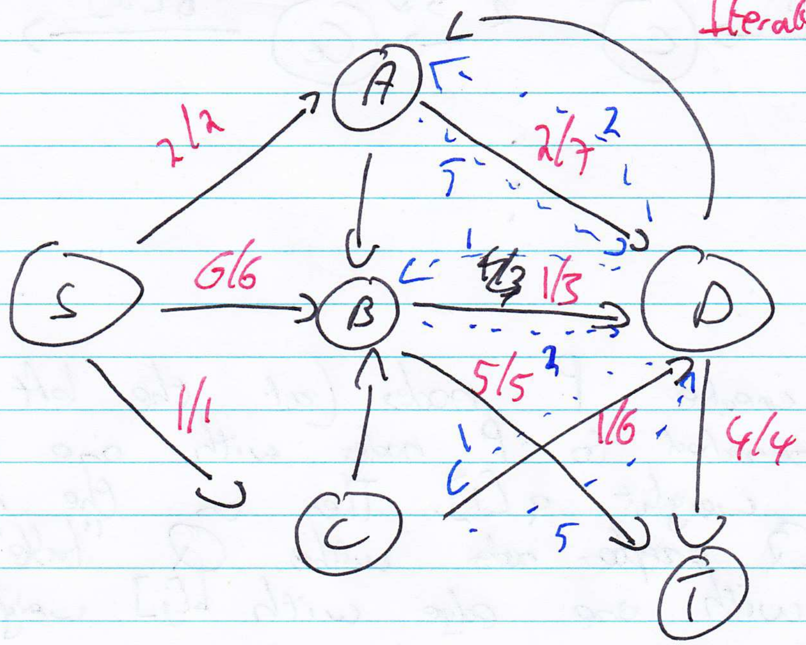
Shortest path: S ⟶ B ⟶ D ⟶ T

Iteration 3



Shortest path: S ⟶ A ⟶ D ⟶ T

Iteration 4



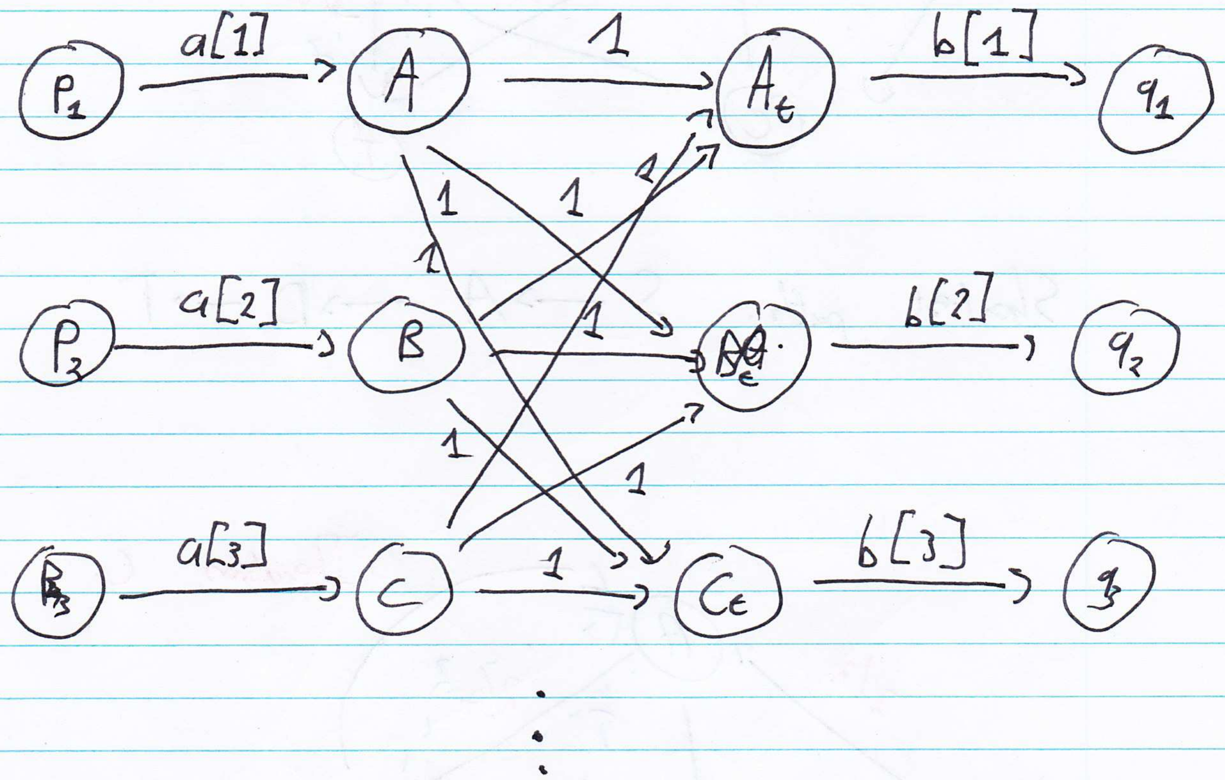
Shortest path: S ⟶ C ⟶ D ⟶ T

Total flow:

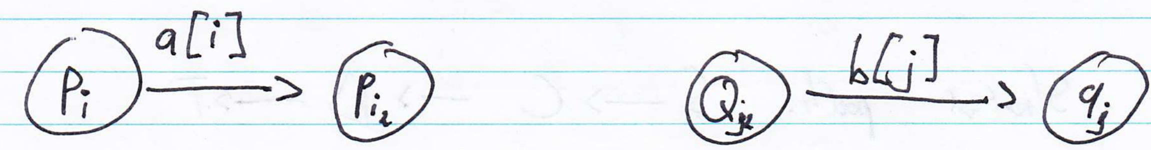
S	A	B	C	D	T
0	2	6	1	4	9

c. The max bit rate is 9 Mbit/s. ~~for the routing ch~~

5.



We can create P nodes (at the left of the diagram) connected to P nodes with one weighted edge with weight a[i]. Then on the right we create Q ~~accepto~~ ~~nodes~~ with Q "table" nodes connected with one edge with b[j] weight:



Now we can connect each node P_{i_2} to all nodes Q_{j_2} with an edge with weight 1. This ensures that no two family members sit at the same table