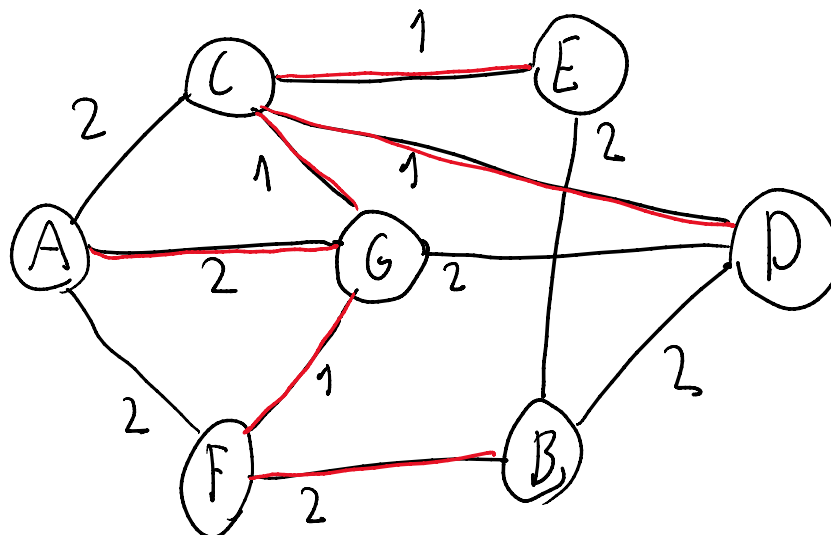


Oppgave 1:



Fringe: G2a F2a C2a

A	B	C	D	E	F	G
0	-∞	-2	-∞	-∞	-2	2
0	0	A	0	0	A	A

Fringe: F1q C1q D2q

A	B	C	D	E	F	G
0	-∞	-1	-2	-∞	1	2
0	0	G	G	0	G	A

Fringe: C1q B2+ D2q

A	B	C	D	E	F	G
0	-2	1	-2	-999	1	2
0	f	G	G	0	G	A

Fringe: E1c D1c B2f

A	B	C	D	E	F	G
0	-2	1	-1	1	1	2
0	f	G	C	C	G	A

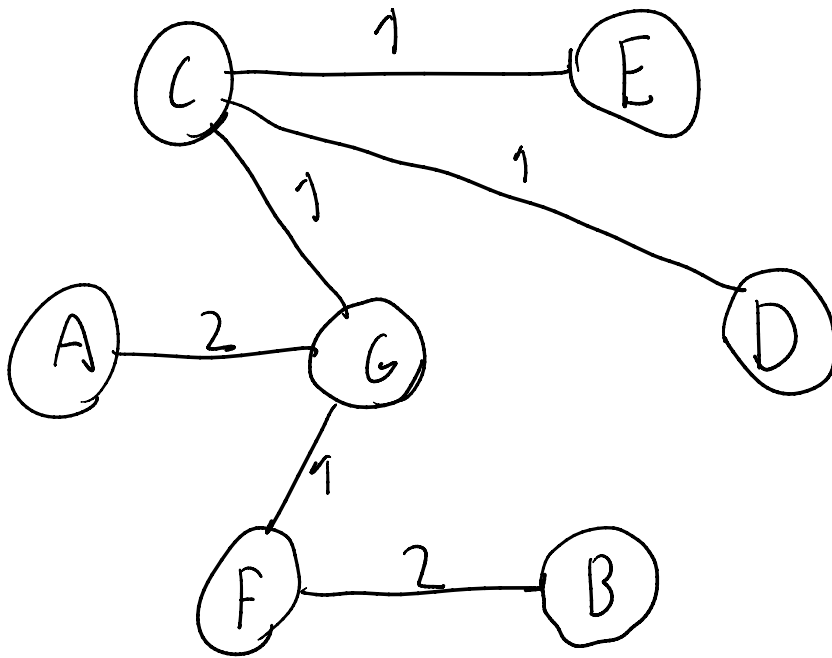
Fringe: D1c B2f

A	B	C	D	E	F	G
0	-2	1	1	1	1	2
0	f	G	C	C	G	A

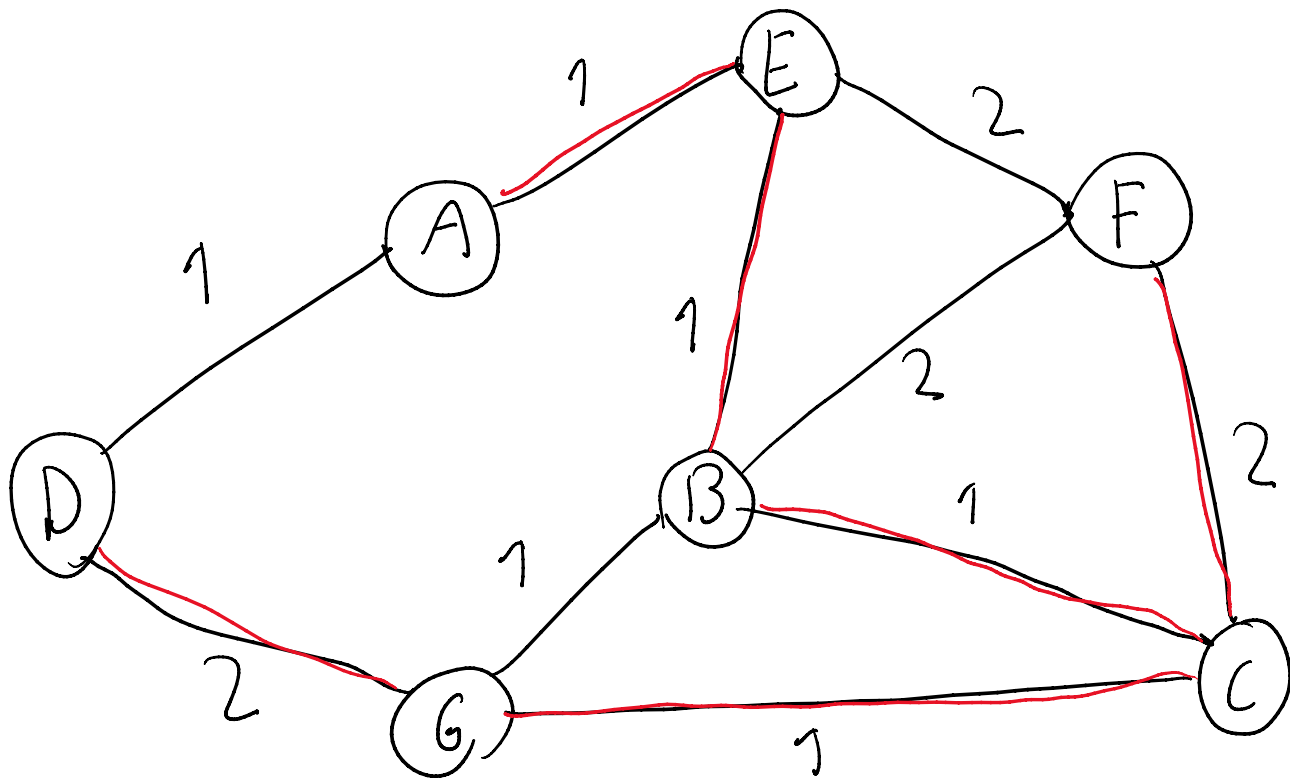
Fringe: B2f

A	B	C	D	E	F	G
0	2	1	1	1	1	2
0	f	G	C	C	G	A

MST:



oppgave 2:



Fringe: G1c B1c F2c

A	B	C	D	E	F	G
-qqq	-1	O	-qqq	-qqq	-2	1
O	C	O	O	O	C	C

Fringe: B1c F2c D3q

A	B	C	D	E	F	G
-qqq	1	O	-3	-qqq	-2	1
O	C	O	G	O	C	C

Fringe: E2b F2c D3q

A	B	C	D	E	F	G
-qqq	1	O	-3	2	-2	1
O	C	O	G	B	C	C

Fringe: F2c A3e D3d

A	B	C	D	E	F	G
-3	1	0	-3	2	2	1
E	C	O	G	B	C	C

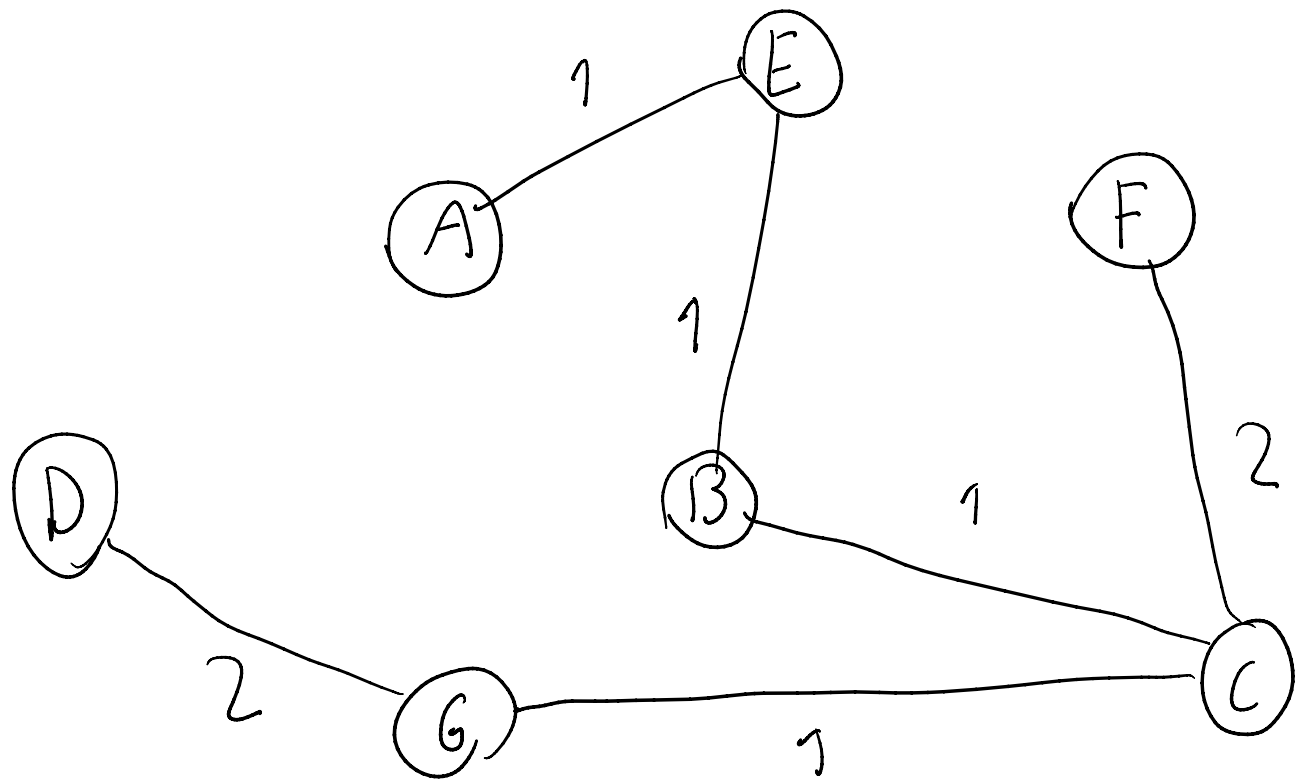
Fringe: A3e D3d

A	B	C	D	E	F	G
3	1	0	-3	2	2	1
E	C	O	G	B	C	C

Fringe: D3d

A	B	C	D	E	F	G
3	1	0	3	2	2	1
E	C	O	G	B	C	C

SP:



Oppgave 3

```
1 void brettUt(Node* node) {
2     /* For loop som går igjennom hver nabo til noden.
3      * Jeg kunne lagt til en if setning som sjekket om noden finnes, men siden for loopen bare
4      * kjøres når det finnes naboer trengs ikke det så lenge antNaboer er riktig.
5      */
6     for(int i = 0; i < node->antNaboer; i++) {
7         //Sjekker om naboer er besøkt, Hvis naboer er besøkt vil den bli erstattet med en kopi.
8         if(node->naboer[i]->besøkt) {
9             node->naboer[i] = kopier(node->naboer[i]);
10        }
11        //Deretter sier vi at naboer er besøkt.
12        node->naboer[i]->besøkt = true;
13        //Går gjennom hele treet rekursivt.
14        brettUt(node->naboer[i]);
15    }
16 }
```