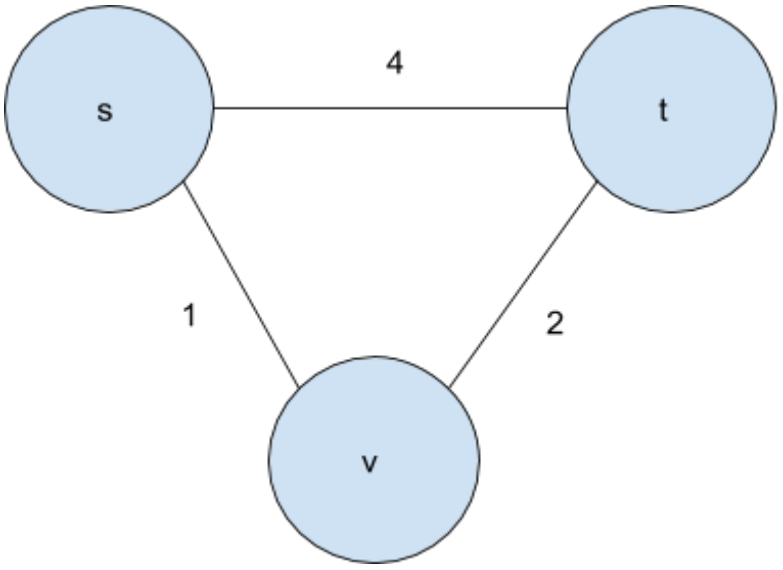
Aufgabe 64:

a) Ja, wir widerlegen diese Aussage durch ein Gegenbeispiel:



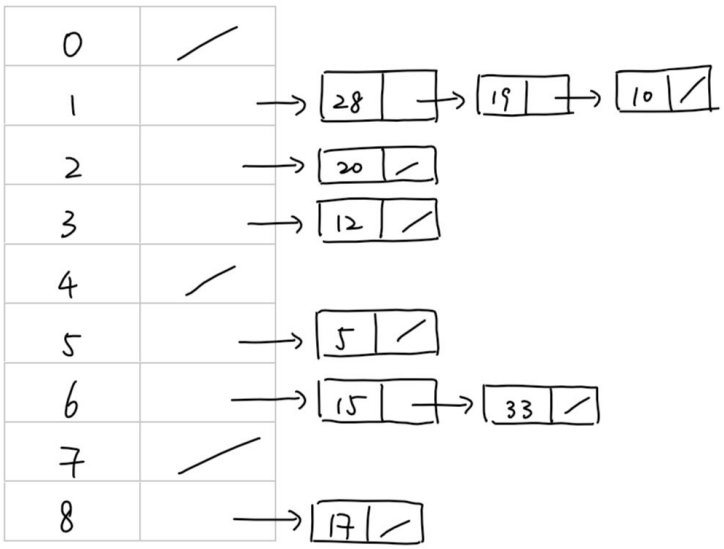
Wie das Bild oben, ist der kürzeste Weg von s nach t: s → v→ t mit der Länge 3. Aber wenn wir beispielsweise 2 zu allen Kantengewichten des Graphen addieren, ist die Länge von s → v→ t gleich 7. Trotzdem ist die Länge von s → t nur 6. D.h. in diesem Fall ist der kürzeste Weg s → t statt s → v→ t, was den kürzesten Weg geändert hat.

b) Nein. Begründung: Im Algorithmus für einen Spannbaum betrachtet man jedes mal nur die günstigste Kante in dem Graphen. Und die Reihenfolge nach den Kantengewichten von allen Kanten bleibt unverändert, obwohl man darauf noch C addiert. Also, was geändert wird, ist nur das Gesamtkantengewicht vom Spannbaum, was sich um (n-1)*C erhöht. Aber der kürzeste Spannbaum bleibt gleich und ist immer optimal.

c) Der kürzeste zusammenhängende Teilgraph soll alle möglichen negativen Kanten enthalten, ansonsten gibt es dann mindestens einen Spannbaum, der mehr negative Kanten enthält, also das Gesamtkantengewicht ist im Vergleich dazu weniger. Wir können den Algorithmus von Kruskal wieder verwenden, dadurch dass man eine Kante hinzu fügt, genau dann wenn diese Kante entweder wie vorher im Algorithmus mit zwei nicht zusammenhängenden Komponenten verbindet oder deren Kantengewicht negativ ist. Und wenn man nach den Kantengewichten sortiert, kommen alle Kanten mit negativen Kantengewichten zuerst. Machen wir dann folgendes:

1) Ergänzen wir erstmal alle Kanten mit negativen Kantengewichten in den Lösungsbaum.
2) Danach führen wir den Algorithmus von Kruskal für alle verbleibende Kanten (mit positiven Kantengewichten) aus.

Aufgabe 67:



0	22
1	88
2	
3	
4	4
5	15
6	28
7	17
8	59
9	31
10	10

a) 5 mod 9 = 5 28 mod 9 = 1 19 mod 9 = 1 15 mod 9 = 6

20 mod 9 = 2 33 mod 9 = 6 12 mod 9 = 3 17 mod 9 = 8

10 mod 9 = 1

b) 10 mod 11 = 10 22 mod 11 = 0 31 mod 11 = 9 4 mod 11 = 4

15 mod 11 = 4 → (4 + 1) mod 11 = 5 ; 28 mod 11 = 6

17 mod 11 = 6 → (6+1) mod 11 = 7

88 mod 11 = 0 → (0 + 1) mod 11 = 1

59 mod 11 = 4 → 5 → 6 → 7 → 8

Aufgabe 69:

a)

Hash-Funktion := mod 3; Offene Adressierung:
9 mod 3 = 0
10 mod 3 = 1
13 mod 3 = 1, 1 schon besetzt von 10 → (1 + 1) mod 3 = 2 - Position für 13

0	9
1	10
2	13

Entfernen 10 mit naiver Löschen-Methode

0	9
1	NIL
2	13

Suchen 13:
h(13) = 1
T[1] != 13 (weil T[1] = NIL)
T[1] == NIL → return “13 nicht vorhanden”

Aber 13 ist in der Tabelle vorhanden, Position 2. **Widerspruch↯**

b)

INITIALIZE(T):

```
global deleted = [True, True, …, True] with length = size of hash table
```

HASH-INSERT(T, k):

```
i = 0
repeat
    j = h(k ,i)
    if deleted[j] == True:                  //free or deleted
        T[j] = k
        deleted[j] = False                  //not free anymore
        return j
    else
        i = i + 1
until i == m
error “Hash table overflow”
```

HASH-RESEARCH(T, k):

```
i = 0
repeat
        j = h(k, i)
        if T[j] == k and deleted[j] == False:        //value is in j-slot and it is valid
                return j
        i = i + 1
until i == m or T[j] = NIL
return NIL
```

HASH-DELETE(T, k):

```
deleted[k] = True                                    //value stays, but not valid
```

Wir fügen übrigens noch ein Flag ”deleted” hinzu. True bedeutet, dass dieses Fach zurzeit von keinem Element besetzt ist. Es kann zwei unterschiedlichen Möglichkeiten geben:

1) Das vorher in diesem Fach besetzte Element wurde schon gelöscht
2) Dieses Fach wird noch nie besucht und dieses Wert in diesem Fach ist noch NIL

Zuerst haben wir eine globale Liste deleted mit True initializiert.
Beim Löschen stellen wir Flag ”deleted” auf True, aber wir löschen nicht einfach den Schlüssel.
Beim Einfügen suchen wir eine passende Position mit Flag ”deleted” True, schreiben wir den Schlüssel einfach auf oder überschreiben wir den alten Schlüssel. Es kommt darauf an, ob dieses Fach vorher schon einmal besucht wurde.
Beim Suchen haben wir gar nichts geändert. Da wir beim Löschen haben wir nur Flag ”deleted” auf True gesetzt, stattdessen einfach das Fach wieder auf NIL setzt. Dadurch treffen wir dann kein Problem bei if-Bedingung: T[j] == NIL