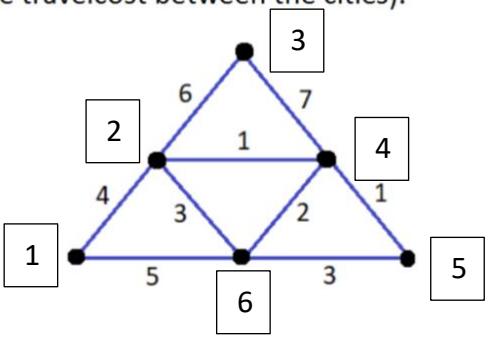
Traveling Salesman (***)

Make use of Z3 to find the cheapest path to visit all six 'cities' in this picture. (The numbers along the edges indicate the travelcost between the cities).



Problem Definition:

In this instance, we have a graph consisting of six cities. We aim to find the most efficient route that visits each city once without exceeding a maximum total distance. By decreasing maximum total distance until the model is unsat, we define the cheapest route.

City Variables:

Each city is represented by an integer variable:

$$\forall i \in Z^+, i \in \{1,..,6\} \Rightarrow city_i \in Z^+$$

Total Distance:
The total distance of the route is represented by:

$$TotalDistance \in Z^+$$

Distance Function:

- The function distance $city_i \in Z^+ for\ i \in \{1, \dots, 6\}$ maps a pair of cities (x, y) to a non-negative integer representing the distance between them.
- The domain of this function is restricted to the set $\{(x, y) \in Z^+ \times Z^+ : 1 \leq x \leq 6 \wedge 1 \leq y \leq 6\}$, representing all possible pairs of the six cities.
- The function is defined piecewise for specific pairs of cities, with a default value of 0 for pairs not explicitly mentioned.

The assertion can be represented as:

$$\forall (x,y) \in Z^+ \times Z^+ : \begin{cases} distance(x,y) = 4, & if\ (x,y) \in \{(1,2),(2,1)\} \\ distance(x,y) = 5, & if\ (x,y) \in \{(1,6),(6,1)\} \\ distance(x,y) = 3, & if\ (x,y) \in \{(2,6),(6,2)\} \\ distance(x,y) = 6, & if\ (x,y) \in \{(2,3),(3,2)\} \\ distance(x,y) = 1, & if\ (x,y) \in \{(2,4),(4,2)\} \\ distance(x,y) = 7, & if\ (x,y) \in \{(3,4),(4,3)\} \\ distance(x,y) = 2, & if\ (x,y) \in \{(6,4),(4,6)\} \\ distance(x,y) = 1, & if\ (x,y) \in \{(5,4),(4,5)\} \\ distance(x,y) = 3, & if\ (x,y) \in \{(5,6),(6,5)\} \\ 0, & otherwise \end{cases}$$

Constraints:

- Each city must be assigned a distinct value from 1 to 6:

$$\forall i,j \in Z^+, i \neq j \Rightarrow city_i \neq city_j$$

- The total distance must not exceed the maximum allowed distance:

$$totalDistance \leq max$$

- The route must form a valid path with non-zero distances between consecutive cities:

$$\forall i \in Z^+, i \in \{1,..,5\} \Rightarrow distance(city_i, city_{i+1}) > 0$$

Z3 SMT Solution

Code:

```
(declare-const city1 Int)
(declare-const city2 Int)
(declare-const city3 Int)
(declare-const city4 Int)
(declare-const city5 Int)
(declare-const city6 Int)

(define-const max Int 16)

(declare-fun distance (Int Int) Int)

(declare-const totalDistance Int)

(assert (forall ((x Int) (y Int))
   (=> (and (>= x 1) (<= x 6) (>= y 1) (<= y 6))
      (= (distance x y)
         (ite (and (= x 1) (= y 2)) 4
         (ite (and (= x 2) (= y 1)) 4
         (ite (and (= x 1) (= y 6)) 5
         (ite (and (= x 6) (= y 1)) 5
         (ite (and (= x 2) (= y 6)) 3
         (ite (and (= x 6) (= y 2)) 3
         (ite (and (= x 2) (= y 3)) 6
         (ite (and (= x 3) (= y 2)) 6
         (ite (and (= x 2) (= y 4)) 1
         (ite (and (= x 4) (= y 2)) 1
         (ite (and (= x 3) (= y 4)) 7
         (ite (and (= x 4) (= y 3)) 7
```

```
            (ite (and (= x 6) (= y 4)) 2
            (ite (and (= x 4) (= y 6)) 2
            (ite (and (= x 4) (= y 5)) 1
            (ite (and (= x 5) (= y 4)) 1
            (ite (and (= x 6) (= y 5)) 3
            (ite (and (= x 5) (= y 6)) 3
            0))))))))))))))))))
        )
    )
    )
)
(assert (<= 1 city1 6))
(assert (<= 1 city2 6))
(assert (<= 1 city3 6))
(assert (<= 1 city4 6))
(assert (<= 1 city5 6))
(assert (<= 1 city6 6))

(assert (distinct city1 city2 city3 city4 city5 city6))

(assert
    (=
        totalDistance
        (+
            (distance city1 city2)
            (distance city2 city3)
            (distance city3 city4)
            (distance city4 city5)
            (distance city5 city6)
        )
    )
)
```

```
(assert
   (not
      (or
         (= (distance city1 city2) 0)
         (= (distance city2 city3) 0)
         (= (distance city3 city4) 0)
         (= (distance city4 city5) 0)
         (= (distance city5 city6) 0)
      )
   )
)
(assert
   (<=
      totalDistance
      max
   )
)
(check-sat)
(get-value(
   totalDistance
   city1
   city2
   city3
   city4
   city5
   city6
))
```

Output:

```
sat
((totalDistance 16)
 (city1 3)
 (city2 2)
 (city3 4)
```

```
(city4 5)
(city5 6)
(city6 1))
```