1 Fast Backups (10 marks)

You are the system administrator responsible for processing the backups of your company and the data needs to be backed up in different data centres. During certain times of the day you have total control over your company's data centres and their network connections, but you need to process the backup requests as fast as possible.

Your company has D data centres represented by $0, 1, \ldots, |D| - 1$. And you have a list `connections` of the direct communication channels between the data centres. `connections` is a list of tuples (a, b, t) where:

- $a \in \{0, 1, \ldots, |D|-1\}$ is the ID of the data centre from which the communication channel departs.
- $b \in \{0, 1, \ldots, |D| - 1\}$ is the ID of the data centre to which the communication channel arrives.
- t is a positive integer representing the maximum throughput of that channel.

Regarding `connections`:

- You cannot assume that the communication channels are bidirectional.
- You can assume that for each pair of data centers there will be at most one direct communication channel in each direction between them.
- You can assume that for every data centre $\{0, 1, \ldots, |D| - 1\}$ there is at least one communication channel departing or arriving at that data centre.
- You cannot assume that the list of tuples `connections` is given to you in any specific order.
- The number of communication channels $|C|$ might be significantly less than $|D|^2$, therefore you should not assume that $|C| = \Theta(|D|^2)$.

Moreover, each data centre has overall limits on the amount of incoming data it can receive per second, and also on the amount of outgoing data it can send per second. `maxIn` is a list of integers in which `maxIn[i]` specifies the maximum amount of incoming data that data centre i can process per second. The sum of the throughputs across all incoming communication channels to data centre i should not exceed `maxIn[i]`. Similarly, `maxOut` is a list of integers in which `maxOut[i]` specifies the maximum amount of outgoing data that data centre i can process per second. The sum of the throughputs across all outgoing communication channels from data centre i should not exceed `maxOut[i]`.

The backup request that you receive has the following format: it specifies the integer ID `origin` $\in \{0, 1, \ldots, |D| - 1\}$ of the data centre where the data to be backed up is located and a list `targets` of data centres that are deemed appropriate locations for the backup data to be stored. `targets` is a list of integers such that each integer i in it is such that $i \in \{0, 1, \ldots, |D| - 1\}$ and indicates that backing up data to server i is fine. Regarding those inputs:

- You can assume that `origin` is not contained in the list `targets`.
- You cannot assume that the list of integers `targets` is given to you in any specific order, but you can assume that it contains no duplicated integers.
- The data to be backed up can be arbitrarily split among the data centres specified in `targets` and each part of the data only needs to be stored in one of those data centres.

Your task is to determine the maximum possible data throughput from the data centre `origin` to the data centres specified in `targets`.

You should implement a function `maxThroughput(connections, maxIn, maxOut, origin, targets)` that returns the maximum possible data throughput from the data centre `origin` to the data centres specified in `targets`.

1.1 Complexity

Given an input with $|D|$ data centres and $|C|$ communication channels, your solution should have time complexity $O(|D| \cdot |C|^2)$

1.2 Example

Consider the example below:

```
# Example
connections = [(0, 1, 3000), (1, 2, 2000), (1, 3, 1000),
(0, 3, 2000), (3, 4, 2000), (3, 2, 1000)]
maxIn = [5000, 3000, 3000, 3000, 2000]
maxOut = [5000, 3000, 3000, 2500, 1500]
origin = 0
targets = [4, 2]

# Your function should return the maximum possible data throughput from the
# data centre origin to the data centres specified in targets.
>>> maxThroughput(connections, maxIn, maxOut, origin, targets)
4500
```

Warning

For all assignments in this unit, you may **not** use python **dictionaries** or **sets**. This is because the complexity requirements for the assignment are all deterministic worst-case requirements, and dictionaries/sets are based on hash tables, for which it is difficult to determine the deterministic worst-case behaviour.

Please ensure that you carefully check the complexity of each in-built python function and data structure that you use, as many of them make the complexities of your algorithms worse. Common examples which cause students to lose marks are **list slicing**, inserting or deleting elements **in the middle or front of a list** (linear time), using the `in` keyword to **check for membership** of an iterable (linear time), or building a string using **repeated concatenation** of characters. Note that use of these functions/techniques is **not forbidden**, however you should exercise care when using them.

Please be reasonable with your submissions and follow the coding practices you've been taught in prior units (for example, modularising functions, type hinting, appropriate spacing). While not an otherwise stated requirement, extremely inefficient or convoluted code will result in mark deductions.

These are just a few examples, so be careful. **Remember that you are responsible for the complexity of every line of code you write!**