Release date: 2022/10/?? Deadline: 2022/10/?? 23:00

Motivation

Consider a simple abstraction of a computer network consisting of n routers communicating information to one another. The communication is established by connecting some pairs of routers using links in such a way that for any pair of routers x and y, it is possible to send data from x to y through a sequence of links. To send a large file over a network, it is customary to divide it into smaller *packets*, and send those packets one-by-one. The choice of the packet size is determined the underlying network: if a packet is larger than the *capacity* of a link, the link is unable to transfer the packet and simply drops it. On the other hand, having too small packets is undesirable due to a per-packet overhead involved in the transfer. Therefore, given a source and a destination in the network, it makes sense to determine the size of the largest packet that can be transferred through the network from the source to the destination.

Problem Definition

Let n denote the number of routers in our network, and let's say that the routers are labeled $0, \ldots, n-1$. Each link is specified by a tuple (u, v, c), where u and v are the endpoints of the link and c is its capacity. Such a link is able to transfer a packet of size at most c from u to v and from v to u (every link is *bidirectional* and has the same capacity in either direction). Given the identities s and t of the *source* and the *target* router respectively, we wish to determine the largest C such that a packet of size C can be transferred over the network from s to t. Your job is to write a program to find that C and a sequence of routers $v_0, v_1, \ldots, v_{r-1}$, where $s = v_0$ and $v_{r-1} = t$, such that for each i, there exists a link of capacity at least C between v_{i-1} and v_i.

Let m denote the number of links in the network. You may assume that the network is connected, that is, it is possible to transfer a small enough packet from any node to any other node. Note that a pair of routers can have zero, one, or more than one links of different capacities between them. For full credit, your program must run in time $O(m \log m)$. (In fact, it is possible to do this in time $O(m)$. Figure out how in the vacation following the major exam.)

Submission Specifications

Submit a single file named `a5.py`. Your file must contain the implementation of a function `findMaxCapacity`. The function must take the following as parameters in the given order.

1. `n`: the number of routers in the network,
2. `links`: a list of 3-tuples of integers. A tuple (u, v, c) in this list represents a bidirectional link of capacity c between u and v, where $u, v \in \{0, \ldots, \texttt{n}-1\}$,
3. `s`: the source router ($\texttt{s} \in \{0, \ldots, \texttt{n}-1\}$),
4. `t`: the target router ($\texttt{t} \in \{0, \ldots, \texttt{n}-1\}$).

The function must return a pair `(C,route)`, where,

1. `C` is the largest number such that a packet of size `C` can be transferred over the network from `s` to `t`.
2. `route` is a list of numbers from the set $\{0, \ldots, \texttt{n}-1\}$ such that `route[0]` is `s`, `route[len(route)-1]` is `t`, and for each `i`, there exists a link of capacity at least `C` between `route[i-1]` and `route[i]`.

Example Test Cases

```
>>> findMaxCapacity(3,[(0,1,1),(1,2,1)],0,1)
(1,[0,1])
```

```
>>> findMaxCapacity(4,[(0,1,30),(0,3,10),(1,2,40),(2,3,50),(0,1,60),(1,3,50)],0,3)
(50,[0,1,3])

>>> findMaxCapacity(4,[(0,1,30),(1,2,40),(2,3,50),(0,3,10)],0,3)
(30,[0,1,2,3])

>>> findMaxCapacity(5,[(0,1,3),(1,2,5),(2,3,2),(3,4,3),(4,0,8),(0,3,7),(1,3,4)],0,2)
(4,[0,3,1,2])

>>> findMaxCapacity(7,[(0,1,2),(0,2,5),(1,3,4), (2,3,4),(3,4,6),(3,5,4),(2,6,1),(6,5,2)],0,5)
(4,[0, 2, 3, 5])
```