Assumptions

Some assumptions due to incomplete requirements in the task:

* City cannot be used in one path twice
* We don’t need end-point for multiple create/update
* We don’t need to keep created cities/ways on application restart/shut down

Decisions

To maintain data consistency I used ReadWriteLock. It’s better than simple synchronize because it provide more flexible approach to solving multithread issues and performance (multiple reading threads, modes etc.). Also I left default “unfair” mode because it normally has higher throughput than fair lock, but this mode can indefinitely postpone reader or writer threads.

In comparison with “no blocks” approach we have delays with write actions. Any write thread will postpone other concurrent threads, and it will be postponed if there is at least 1 read or write thread with lock. The same for read threads unless read threads do not block other read threads. In our service write action is fast, so it’s ok to use unfair read-write lock for low load and fair read-write for big load.

Of course this is not the best solution, because we can block only part of graph like for example ConcurrentHashMap, but I haven’t found ready-made solution. Or another way to keep data in DB and just rebuild cached graph every time we put changes.

In accordance with the assumptions above I decided to keep only vertexes instead of edges in algorithm to path availability check. Also I decided to use separate component with prototype scope for graph actions because we don’t need to keep changes on restart but this component is still “single” for main service.

Thanks for the jgrapht library I don’t have to create my own graph structure. Also I’ve tried to find search algorithm for all paths but I haven’t found any suitable and working well.

I haven’t divided service to sdk and core because we have no other micro-services here. There is no documentation (comments, swagger etc.), just a small readme, because this is a test project without any chance to be supported later, and it’s not a common library to document all methods with javadoc.