```
class AbstractLock:
  def acquire(record, op, txn_id):
    ''' return True if `op` can execute on `record`
        concurrently with other lock holders;
        adds txn_id to set of lock holders
    '''
  def release(txn_id):
    ''' called when a transaction commits or aborts,
        releasing its locks, remove `txn_id` from
        lock holders
    '''


class ZSet:
  class AbstractLock:
    def acquire(record, op, txn_id):
      if (op.is_add() and self.mode == ADD) or
         (op.is_read() and self.mode == READ):
        self.holders.add(txn_id)
        return True
      # ...

    def release(txn_id):
      self.holders.remove(txn_id)
      if self.holders.empty():
        self.mode = None
```