def tidlists(transactions):

tl = defaultdict(set)

for tid, t in enumerate(transactions):

for item in t:

tl[item].add(tid)

return list(tl.items())

class IntersectAll:

def \_\_and\_\_(self, other):

return other

IntersectAll = IntersectAll()

def eclat(items, minsup=0, minlen=1):

frequent\_itemsets = {(): IntersectAll}

def recurse(items, prefix):

while len(items) > 0:

item, item\_tidlist = items.pop()

l = prefix + (item,) # l is the (ordered) tuple of items we are looking for

new\_tidlist = frequent\_itemsets[prefix] & item\_tidlist

if len(new\_tidlist) >= minsup: # add frequent\_itemsets to the new frequent\_itemsets

frequent\_itemsets[l] = new\_tidlist

# define the new l-conditional database

new\_items = []

for new\_item, \_item\_tidlist in items:

new\_item\_tidlist = \_item\_tidlist & item\_tidlist

if len(new\_item\_tidlist) >= minsup:

new\_items.append((new\_item, new\_item\_tidlist))

# recurse, with l as prefix

recurse(new\_items, l)

recurse(items.copy(), ())

return {k: len(v) for k, v in frequent\_itemsets.items() if len(k) >= minlen}