D.8 Implementieren Sie einen Algorithmus `cluster`, der zu vorgegebener Distanzfunktion d über der Menge $U = \{0, \ldots, m-1\}$ eine optimale Lösung von MAXIMUM l-CLUSTERING zurückliefert. Dabei sollen Sie `kruskal` aus der vorhergehenden Aufgabe unverändert verwenden und so das Problem MAXIMUM l-CLUSTERING auf MST reduzieren. Ihr Algorithmus besteht aus den folgenden Schritten:

1. Konstruktion eines vollständigen Graphen aus der Distanzfunktion
2. Aufruf von `kruskal`
3. Entfernen der $l-1$ teuersten Kanten aus dem Spannbaum
4. Ausgabe einer Liste der Zusammenhangskomponenten des Graphen mit den verbliebenen Kanten

Führen Sie eine Laufzeitanalyse durch.

```python
# l-CLUSTERING
from d07_kruskal import kruskal
from b12_comp import comp

def cluster(m,d,l):
    # 1. vollstaendigen Graph mit m Knoten, m^2 Kanten und Kosten d[u,v] konstruieren
    G = [{j:d[i,j] for j in range(m) if i!=j} for i in range(m)]    # O(m^2)

    # 2. Alg. kruskal unveraendert aufrufen
    (mst,_) = kruskal(G)    # O(k*log(m))

    # 3. die l-1 teuersten Kanten löschen
    # Liste mit Kanten und dazugehörigen Kosten erstellen
    li = []
    for e in mst:   # O(k)
        (u,v) = e
        dist = d[u,v]
        li.append((dist,(u,v)))

    # Liste reversed sortieren und ersten l-1 Elemente "entfernen"
    edges = sorted(li, reverse=True)[l-1:]      # O(k*log(k))


    # 4. Zush.komponenten bestimmen,  via B.12 (comp)
    # dazu Graph mit allen Knoten konstuieren und Kanten einf.
    Graph = [set() for _ in range(m)]
    for e in edges:     # O(k)
        (_,(u,v)) = e
        Graph[u].add(v)
        Graph[v].add(u)

    # Zusammenhangskomponenten bestimmen
    comps = comp(Graph)     # O(m+k)

    return comps
```

Laufzeitanalyse

Es gilt $k \leq m^2$:

$$\rightarrow O(m^2) + O(m^2 \cdot \log(m)) + O(m^2) + O(m^2 \cdot \log(m^2)) + O(m^2) + O(m^2 + m)$$
$$\in O(m^2 \cdot \log(m))$$

Comp(Graph) in $O(m)$, weil der Graph genausoviele Kanten hat, wie der Spannbaum und der Spannbaum ist in $O(m)$, weil Spannbaum ist ein Baum und im Baum gilt $k = m - 1$. Also gilt für Comp:

$$O(m + k) = O(m + m) = O(m)$$

Gleiches würde für „edges“ gelten. Da müsste dann wahrscheinlich auch $O(m \cdot \log m)$ hin