D.4 Ergänzen Sie den Algorithmus `dijkstra` zu `dijkstra_paths`, der zusätzlich eine Liste `p` mit den Vorgängerknoten zurückliefert (vgl. Skript). Implementieren Sie außerdem eine rekursive Funktion `shortest_path`, die einen kürzesten s-v-Pfad P_v als Knotenliste liefert.

```
5   def dijkstra_paths(G,s):
6       m = len(G)
7       d = [None]*m                # (geschaetzte) Pfadkosten
8       d[s] = 0                    # sind 0 fuer den Startknoten
9       Q = {u for u in range(m)}   # alle Knoten
10      p = [None]*m                # Liste der Vorgängerknoten
11      while Q:
12          # Auswahl von v mit kleinstem d-Wert
13          (_,v) = min({(d[u],u) for u in Q if d[u]!= None})
14          # v aus Q entfernen, d[v] ist endgueltig
15          Q.remove(v)
16          # Schaetzungen fuer Nachfolger von v aktualisieren
17          for u in G[v]:
18              alt = d[v] + G[v][u]            # alternative Kosten
19              if d[u]==None or alt < d[u]:
20                  d[u] = alt
21                  p[u] = v                    # setze v als Vorgänger von u
22
23      return d,p
24
25
26  # s-v-Pfad bestimmen
27  def shortest_path(s,v,p):
28      if v == s:
29          return [s]
30
31      return shortest_path(s,p[v],p) + [v]
32
```