#uni/semester3/alda/vorlesung8

Algorithmus von Prim

```
void minimumSpanningTree(Graph G, Vertex[ ] p ) {
    kl = ∅;   // leere Kandidatenliste
    Set<Vertex> baumKnoten;   // Knoten im aufspannenden Baum
    for (jeden Knoten v) {
        c[v] = ∞; p[v] = undef;
    }
    s = 1; c[s] = 0;   // irgendeinen Startknoten wählen
    kl.insert(s);
    while (! kl.empty() ) {
        lösche Knoten v aus kl mit c[v] minimal;
        baumKnoten.add(v);
        for (jeden Nachbarknoten w von v)
            if ( ! baumKnoten.contains(w)) {
                if (c[w] == ∞)   // w noch nicht in Kandidatenliste
                    kl.insert(w);
                if ( c(v,w) < c[w] ) { // c[w] verbessert sich
                    p[w] = v;
                    c[w] = c(v,w);
                }
            }
        }
    }
```

Eingabe: Gewichteter Graph G.

Ausgabe: Vorgängerfeld p, das den minimal aufspannenden Baum darstellt.

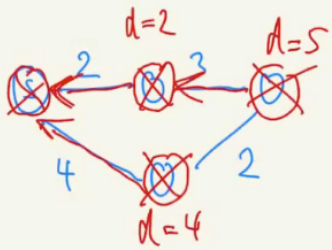
In der Kandidatenliste kl sind alle Knoten abgespeichert, die als nächstes besucht werden können.

c[v] gibt für jeden Kandidaten das Gewicht der billigsten Kante an, mit der er von den bereits besuchten Knoten erreicht werden kann. Für die noch nicht besuchten Knoten, die noch keine Kandidaten sind, ist c[w] = ∞.

c(v,w) = Gewicht der Kante (v,w).

➜ Set<Vertex>.. ist wichtig, weil man sonst in einer endlosen Schleife kommt!

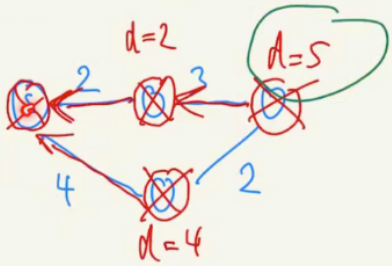
Unterschied Dijkstra – Prim:



Dijkstra

○ Kand.

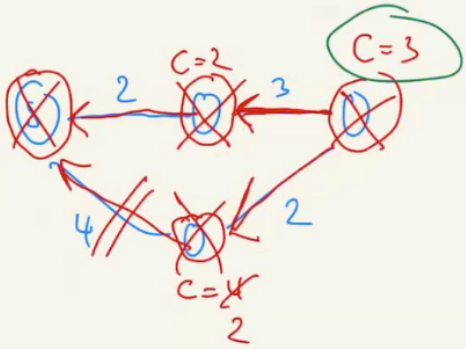
× besucht



Dijkstra

○ Kand.

× besucht



Prim