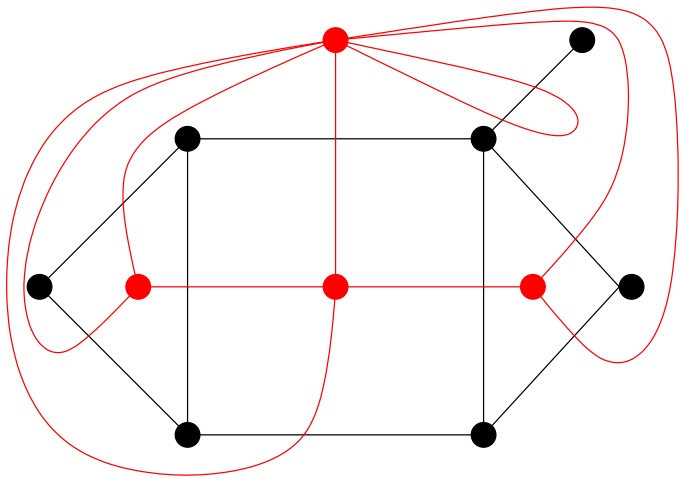
Praktikum *Planare Graphen*

Steffen Mecke

17. November 2006

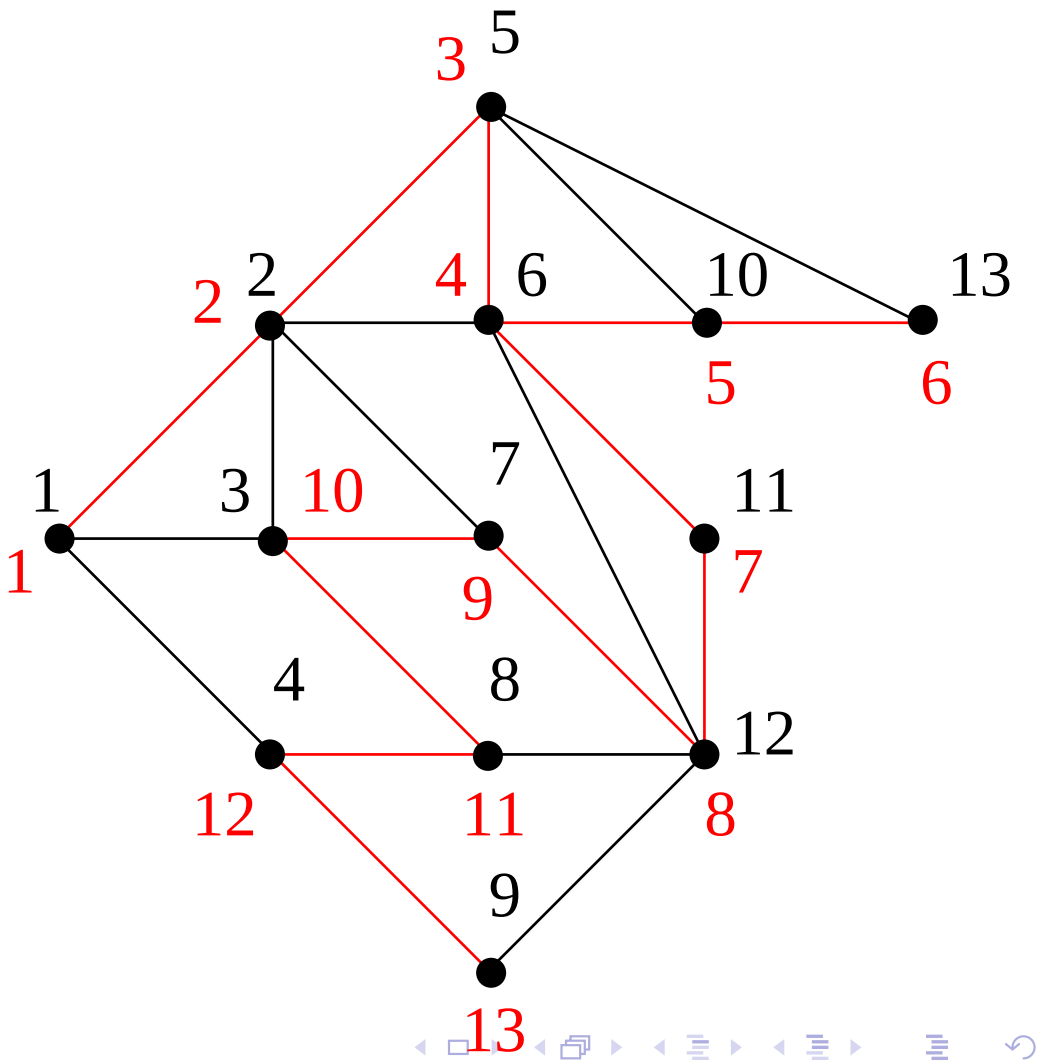
Dualgraph

- Facette $\rightsquigarrow$ Knoten
- adjazente Facetten $\rightsquigarrow$ Kante
- ‚Antenne' $\rightsquigarrow$ Schleife
- Schnitt $\leftrightsquigarrow$ Kreis



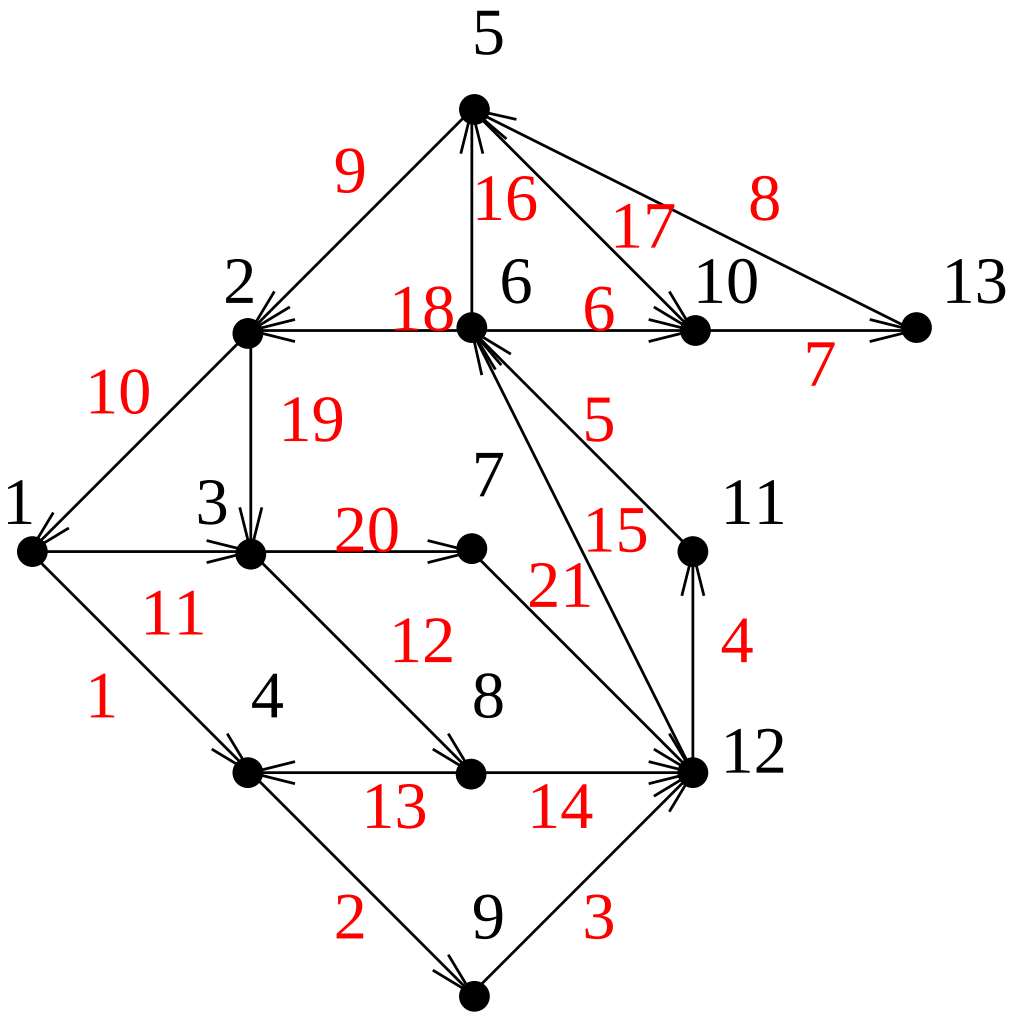
Graphsuche

- Breitensuche (BFS)
- Tiefensuche (DFS)



Graphsuche

- Breitensuche (BFS)
- Tiefensuche (DFS)
- Auswahlstrategie:
  z. B. Right-First-DFS



de.uka.algo.planalgo.praktikum.graph.PlanarGraph

Ein PlanarGraph ist ein

- JUNG-Graph
- mit zusätzlicher Kantenordnung an den Knoten:

```
List<PlanarEdgeTip> getEdgeTipList();
PlanarEdgeTip   getNextEdge(PlanarEdgeTip t)
PlanarEdgeTip   getPreviousEdge(...)
```

- PlanarEdgeTip ist eines der Enden einer Kante:

```
PlanarEdge   getEdge()
PlanarEdgeTip   getOpposite()
PlanarVertex   getVertex()
boolean   isIncoming()
```

de.uka.algo.planalgo.praktikum.graph.PlanarGraph II

- PlanarFace: Repräsentiert Facetten. Interface ähnlich wie PlanarVertex
- Set getIncident...-Methoden des Interface Graph sind für Algorithmen auf planaren Graphen meist weniger geeignet.
- API unter `http://i11www.iti.uni-karlsruhe.de/teaching/WS_0607/planalgo/jungX/doc/index.html`

```
public static void RF_DFS(PlanarGraph g){
  Queue<PlanarVertex> candidates = new ...;
  PlanarVertex start = ...
  candidates.add(start);
  while (!candidates.isEmpty()) {
    PlanarVertex currentVertex = candidates.poll();
    // TODO: start with right edge relative to incoming edge
    for (PlanarEdgeTip t :  currentVertex.
        getEdgeTipList()) {
      PlanarEdge e = t.getEdge();
      PlanarVertex o = (PlanarVertex) e.getOpposite
          (currentVertex);
      if (!o.containsUserDatum("visited")) {
        candidates.add(o);
        o.addUserDatum("visited", e, UserData.CLONE
            );
}}}   }
```