DHBW Lörrach
Duale Hochschule
Baden-Württemberg

Algorithmen und Komplexität
TIF 21A/B
Dr. Bruno Becker

Übungsblatt 5: Graphen

Übungsblatt 5 –Aufgabe 1

Gegeben sei eine Adjazenz-Liste *A*, die Graphen mit maximal *n* Knoten abspeichern kann.
Die verkettete Liste mit den Nachbarn eines Knoten sei einfach verkettet und unsortiert, d.h A[i] zeigt auf den ersten Nachbarn von *i* (oder ist *null*).

a) Erstellen Sie eine Methode *InsertNode(int i)*, die einen Knoten mit dem Schlüssel *i* (*i*<*n*) in die Adjazenzliste *A* einfügt.

b) Erstellen Sie eine Methode *InsertEdge(int i, j),* die eine Kante <*i,j*> in den Graphen einfügt. Wie oft kommt die Kante der Adjazenzliste vor?

a)

```
public void insert node (int i);
{  a[i] = null; // Nichts zu tun
}
```

b)

```
public void insert edge (int I,j);
{ // Füge j in Adjazenzliste von i als 1. Element ein
 Private help = new node();
 help.key = j;
 help.next = a[i];  // zeigt auf 1. Nachbarn von i oder ist NULL
 a[i] = help; // Einfügen von j am Anfang

// Füge i in Adjazenzliste von j als 1. Element ein
 help = new node();
 help.key = i;
 help.next = a[j];  // zeigt auf 1. Nachbarn von j oder ist NULL
 a[j] = help; // Einfügen von i am Anfang
}
```

Übungsblatt 5 –Aufgabe 1

Gegeben sei eine Adjazenz-Liste *A*, die Graphen mit maximal *n* Knoten abspeichern kann.
Die verkettete Liste mit den Nachbarn eines Knoten sei einfach verkettet und unsortiert, d.h A[i] zeigt auf den ersten Nachbarn von *i* (oder ist *null*).

c) Erstellen Sie eine Methode *DeleteNode(int i),* die einen Knoten und alle angrenzenden Kanten aus dem Graphen löscht.
d) Welchen Aufwand haben die Operationen im mittleren und schlechtesten Fall?
e) Was ändert sich am Aufwand, wenn die Nachbarlisten der Knoten sortiert gehalten werden?

c) Hilfsmethode:

```
public void deletefromadjlist (int i,j);
{ // Löscht i aus  Adjazenzliste von j – Annahme i ist in Liste enthalten
 private node h1 = a[j];
 private node h2 = a[j].next;
 while (h2.key ≠ i)
    {  h1 = h1.next;
               h2 = h2.next;
    } // h2 zeigt auf das zu löschende Element, h1 auf das Element davor
 h1.next = h2.next;
 h2.next = Null;
}
```

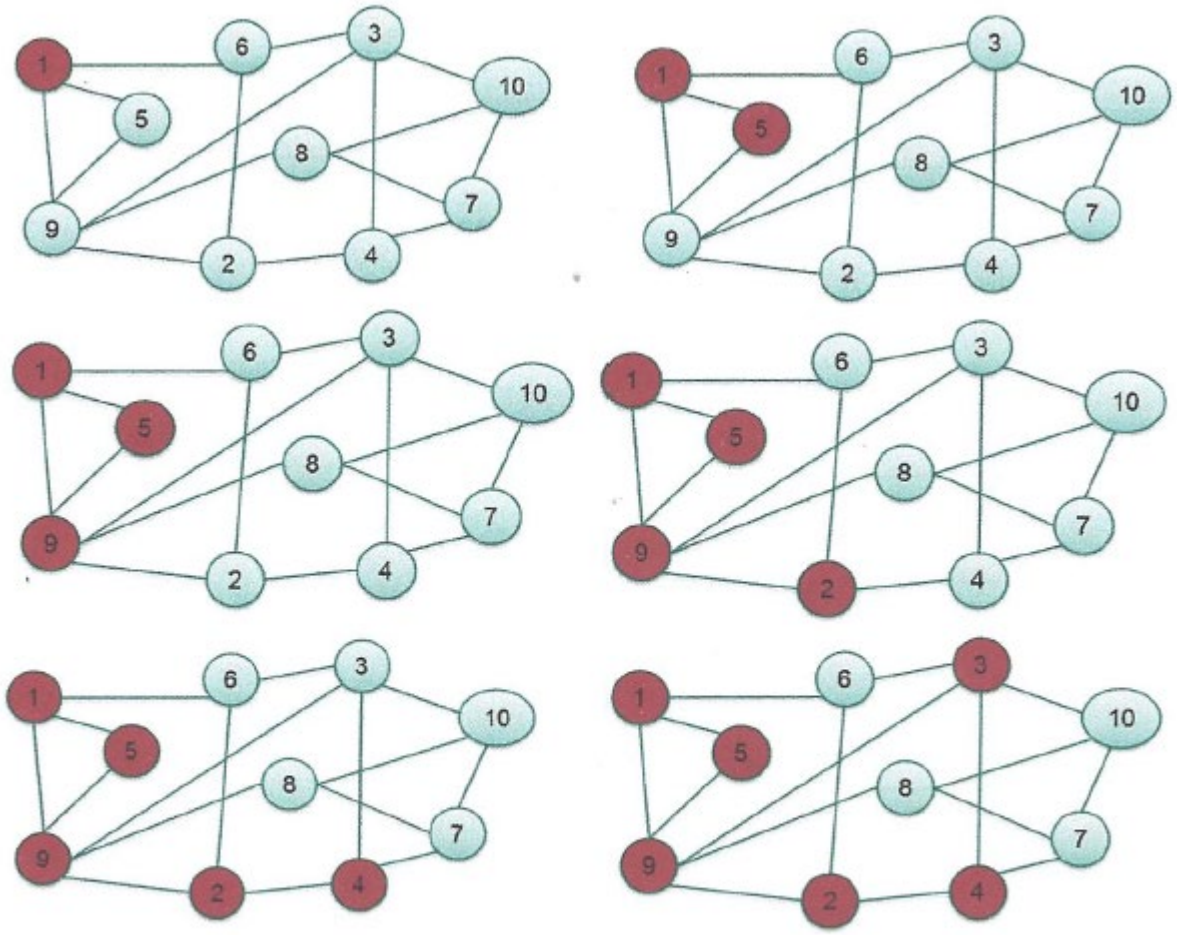
d) $O(1), O(1), O(|V|^2)$
e) $O(1), O(|V|), O(|V|^2)$

```
public void deletenode (int i);
{ // Für jeden Nachbarn j von i wird i mit der
Hilfsmethode aus der Adjazenzliste von j gelöscht
   private node help = a[i];
   while (help ≠ Null)
     {  deletefromadjlist (I, help.key);
        help = help.next;
     }
   a[i] = Null;
}
```

Übungsblatt 5 –Aufgabe 2

Tiefensuche

Schritt	Knoten	Nachbarn	markiert
1	1	5 6 9	1
2	5	1 9	1 5
3	9	1 2 3 5 8	1 5 9
4	2	4 6 9	1 5 9 2
5	4	2 3 7	1 5 9 2 4
6	3	4 6 9 10	1 5 9 2 4 3

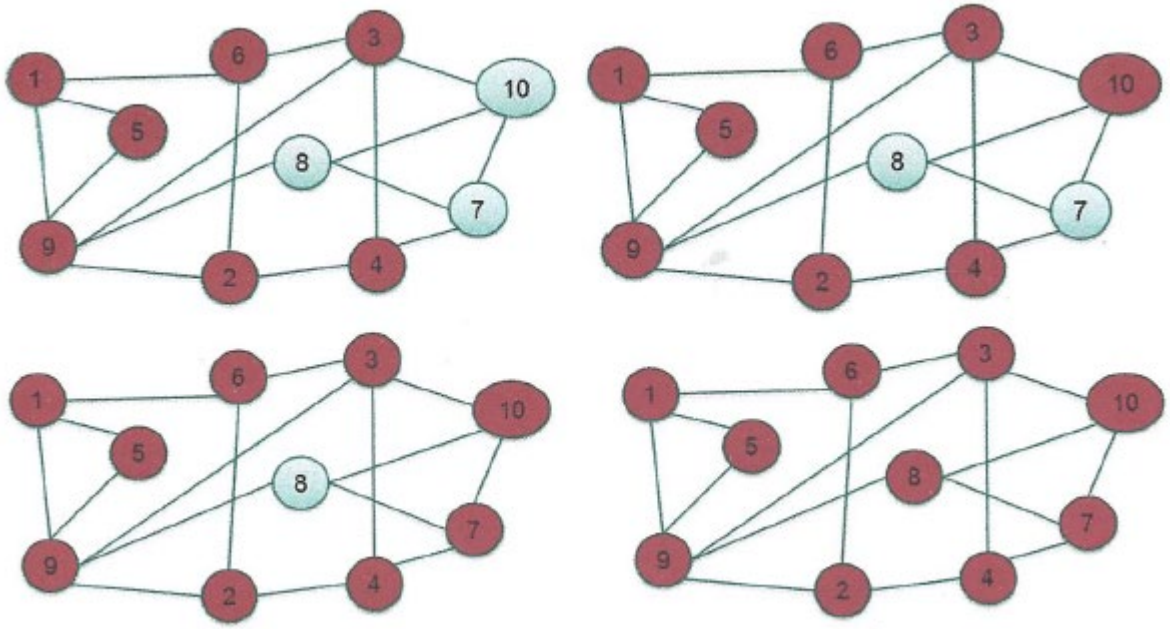


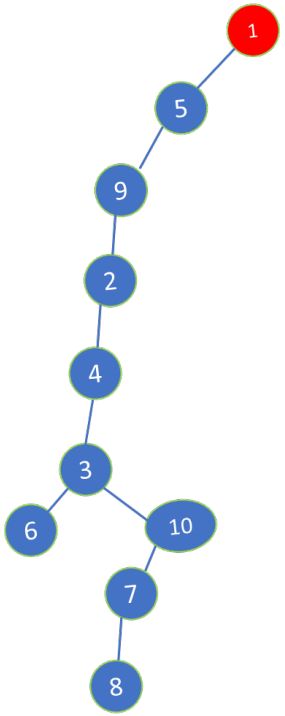
4

Übungsblatt 5 –Aufgabe 2

Tiefensuche

Schritt	Knoten	Nachbarn	markiert
1	1	5 6 9	1
2	5	1 9	1 5
3	9	1 2 3 5 8	1 5 9
4	2	4 6 9	1 5 9 2
5	4	2 3 7	1 5 9 2 4
6	3	4 6 9 10	1 5 9 2 4 3
7	6	1 2 3	1 5 9 2 4 3 6
8	3	4 6 9 10	1 5 9 2 4 3 6
9	10	3 7 8	1 5 9 2 4 3 6 10
10	7	4 8 10	1 5 9 2 4 3 6 10 7
11	8	7 9 10	alle

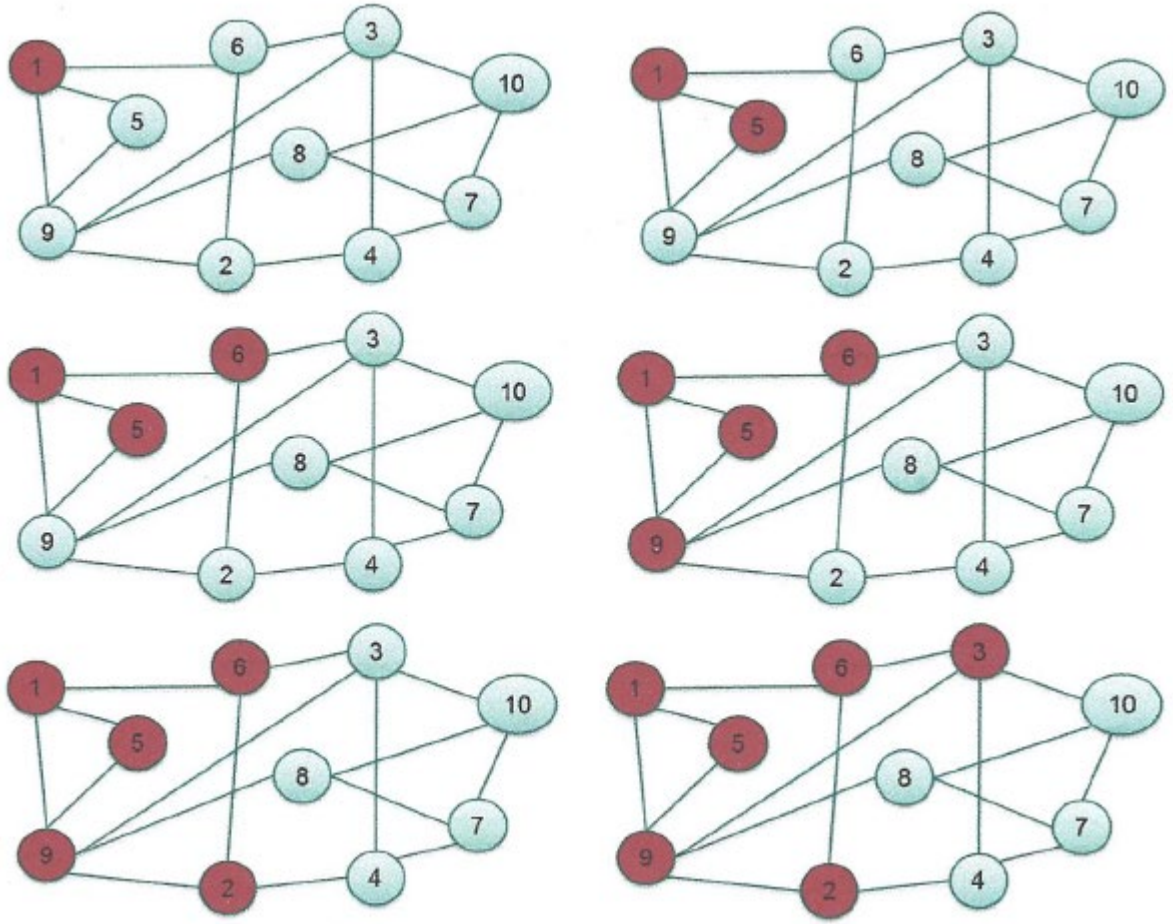




Übungsblatt 5 –Aufgabe 3

Breitensuche

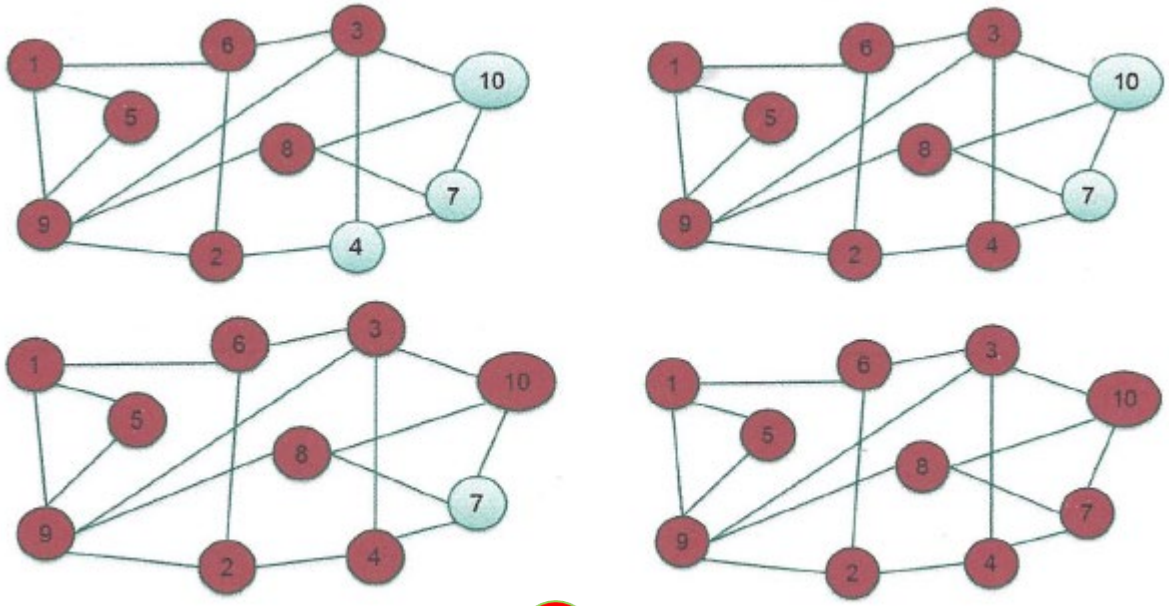
Schritt	Knoten	Nachbarn	markiert
1	1	5 6 9	1
2	5	1 9	1 5
3	6	1 2 3	1 5 6
4	9	1 2 3 5 8	1 5 6 9
5	2	4 6 9	1 5 6 9 2
6	3	4 6 9 10	1 5 6 9 2 3

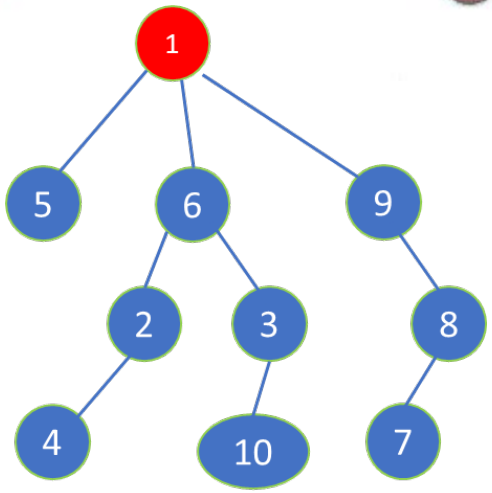


Übungsblatt 5 –Aufgabe 3

Breitensuche

Schritt	Knoten	Nachbarn	markiert
1	1	5 6 9	1
2	5	1 9	1 5
3	6	1 2 3	1 5 6
4	9	1 2 3 5 8	1 5 6 9
5	2	4 6 9	1 5 6 9 2
6	3	4 6 9 10	1 5 6 9 2 3
7	8	7 9 10	1 5 6 9 2 3 8
8	4	2 3 7	1 5 6 9 2 3 8 4
9	10	3 7 8	1 5 6 9 2 3 8 4 10
10	7	4 8 10	alle

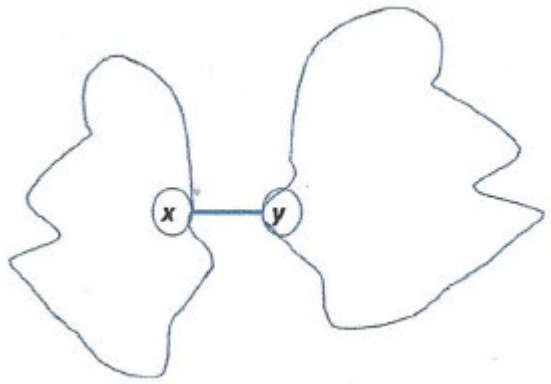




Übungsblatt 5 –Aufgabe 4

Eine Kante in einem zusammenhängenden, ungewichteten Graphen heißt ***Brücke***, wenn das Entfernen dieser Kante den Graphen in zwei Teile zerfallen lässt.

Entwerfen Sie einen möglichst effizienten Algorithmus, der zu einem gegebenen Graphen G und einer Kante <i,j> ermittelt, ob die Kante <i,j> eine Brücke ist.



a= <x,y> ist Brücke von G ⇔ G´= (V, E \ {a}) besteht aus 2 Zusammenhangskomponenten

Idee: DFS mit Start x auf G´:

If (y ε DFS(G´,x)
  Ausgabe („ a ist keine Brücke“)
else
  Ausgabe („ a ist Brücke“)