

Norman May

B6, 29, Raum C0.05
68131 Mannheim
Telefon: (0621) 181-2517
Email: norman@pi3.informatik.uni-mannheim.de

Matthias Brantner

B6, 29, Raum C0.05
68131 Mannheim
Telefon: (0621) 181-2517
Email: msb@pi3.informatik.uni-mannheim.de

Algorithmen und Datenstrukturen
Wintersemester 2004/05

3. Lösungsvorschlag
19. November 2004

Aufgabe 1

6 Punkte

Zeigen Sie, daß die Erreichbarkeitsrelation eine Äquivalenzrelation auf der Knotenmenge eines ungerichteten Graphen $G = (V, E)$ definiert.

Lösung

Eine Relation $R \subseteq V \times V$ heißt Äquivalenzrelation wenn Sie reflexiv, symmetrisch und transitiv ist.

Zu zeigen: Die Erreichbarkeitsrelation $R \subseteq V \times V$ definiert eine Äquivalenzrelation auf V .

R ist reflexiv:

Per Definition eines ungerichteten Graphen bildet jeder Knoten einen Pfad der Länge 0. Also gilt $\forall v \in V : (v, v) \in R$.

R ist symmetrisch:

Angenommen $(u, v) \in R$, dann existiert ein Pfad $\langle v_0, v_1, \dots, v_k \rangle$ mit $v_0 = u$ und $v_k = v$. Konstruiere dazu einen Pfad $\langle v_k, v_{k-1}, \dots, v_0 \rangle$ in G . Zu zeigen, $\langle v_k, v_{k-1}, \dots, v_0 \rangle$ ist ein Pfad in G .

Widerspruchsbeweis: Sei $\langle v_k, v_{k-1}, \dots, v_0 \rangle$ kein Pfad in G . Also existiert mind. ein $(v_i, v_{i-1}) \notin E, 1 \leq i \leq k$. Da aber E eine Menge von ungeordneten Tupeln ist müßte somit auch $(v_{i-1}, v_i) \notin E$ gelten. Dies ist aber ein Widerspruch zur Annahme, daß $\langle v_0, v_1, \dots, v_k \rangle$ ein Pfad in G ist.

R ist transitiv:

Aus $(u, v) \in R$ folgt, es existiert ein Pfad $\langle v_0, v_1, \dots, v_k \rangle$ mit $v_0 = u$ und $v_k = v$. Aus $(v, w) \in R$ folgt, es existiert ein Pfad $\langle v_k, v_{k+1}, \dots, v_{k+l} \rangle$ mit $v_k = v$ und $v_{k+l} = w$. Somit existiert ein Pfad $\langle v_0, v_1, \dots, v_k, v_{k+1}, \dots, v_{k+l} \rangle$ in G und es gilt $(u, w) \in R$.

Aufgabe 2**4 Punkte**

Es seien die Graphen $G = (V, E)$ und $G' = (V', E')$ gegeben mit $|V| = |V'|$

Aufgabe 2 a)**2 Punkte**

Wie viele verschiedene Bijektionen gibt es zwischen den Mengen V und V' ?

Lösung $|V|!$ **Aufgabe 2 b)****2 Punkte**

Wie viele induzierte Teilgraphen existieren für den Graphen G ?

Lösung $2^{|V|}$

Aufgabe 3**10 Punkte**

Das Problem k -Clique berechnet, ob ein ungerichteter Graph $G = (V, E)$ einen *vollständigen* Teilgraphen $G' = (V', E')$ enthält mit $|V'| = k$ für eine beliebige natürliche Zahl $k \leq |V|$.

Schreiben Sie ein Programm in einer von Ihnen gewählten Programmiersprache (C, C++, Java oder Scheme), welches das Problem k -Clique berechnet. Benutzen Sie zur Berechnung folgende zwei Funktionen, deren Signatur exemplarisch in Java gegeben ist.

Zur Unterstützung finden Sie auf der Homepage eine Beispielklasse in Java (Datei: Clique.java), welche ein Grundgerüst vorgibt. Auf die weiteren Programmiersprachen sollte diese Struktur leicht übertragbar sein.

Geben Sie die Implementierung zusammen mit Testfällen bei Ihrem Tutor per E-mail und ausgedruckt ab. Kommentieren Sie Ihre Implementierung ausreichend.

```
/**
 * select an initial set of vertices from S
 * @param k – the number of vertices to choose
 * @param S – the current set of vertices
 * @return the initial set of vertices
 */
private int[] firstKSubSet(int k, int[] S);

/**
 * move to the next subset of vertices of the set of vertices in S
```

```

    * @param s – the old set of vertices
    * @param k – the size of the clique
    * @param S – the set of available vertices
    * @return the next subset of vertices
    */
private int[] nextKSubSet(int[] s, int k, int[] S);

/**
 * returns for a given set of nodes V and an
 * adjacency matrix E if the graph contains
 * a k-clique?
 */
public boolean kClique(int k);

```

Lösung

```

public class Clique {

    private int[] V; // set of nodes
    private boolean[][] E; // edges as adjacency matrix

    public Clique(int[] V, boolean[][] E) {
        // initialize members
        this.V = V;
        this.E = E;
    }

    /**
     * returns for a given set of nodes V and an
     * adjacency matrix E if the graph contains
     * a k-clique?
     */
    public boolean kClique(int k) {
        int[] s = firstKSubSet(k, V);
        while (s != null) {
            int c = 0;
            for (int i = 0; i < k - 1; ++i) {
                for (int j = i; j <= k - 1; ++j) {
                    if (E[V[s[i]]-1][V[s[j]]-1]) {
                        ++c;
                    }
                }
            }
            if (c == (((k*(k-1))/2)))
                return true;
        }
        s = nextKSubSet(s, k, V);
    }
}

```

```

    return false;
}

/**
 * move to the next subset of vertices of the set of vertices in S
 * @param s – the old set of vertices
 * @param k – the size of the clique
 * @param S – the set of available vertices
 * @return the next subset of vertices
 */
private int[] nextKSubSet(int[] s, int k, int[] S) {
    int n = S.length - 1;
    int i = 0;
    while (i < k && (s[k-i-1] == (n-i))) {
        if (i == k-1)
            return null;
        ++i;
    }
    s[k-i-1] = s[k-i-1]+1;
    for (int j = k - i + 1; j <= k-1; ++j) {
        s[j-1] = s[j-2] + 1;
    }
    return s;
}

/**
 * select an initial set of vertices from S
 * @param k – the number of vertices to choose
 * @param S – the current set of vertices
 * @return the initial set of vertices
 */
private int[] firstKSubSet(int k, int[] S) {
    if (k > S.length)
        return null;
    int[] s = new int[k];
    for (int i=0; i <= k-1; ++i) {
        s[i] = i;
    }
    return s;
}

private static void usage() {
    System.err.println("usage: java Clique <int>");
}

public static void main(String[] args) {

```

```

if (args.length != 1) {
    usage();
    System.exit(-1);
}

int [] V1 = { 1, 2, 3, 4, };
int [] V2 = { 1, 2, 3, 4, 5, 6};

boolean[][] E1 = {{ false, true, true, true },
                  { true, false, true, true },
                  { true, true, false, true },
                  { true, true, true, false } };

boolean[][] E2 = {{ false, true, false, false, true, true },
                  { true, false, false, false, true, true },
                  { false, false, false, false, false, false },
                  { false, false, false, false, false, false },
                  { true, true, false, false, false, true },
                  { true, true, false, false, true, false } };

Clique q1 = new Clique(V1, E1);
Clique q2 = new Clique(V2, E2);

try {
    int k = Integer.parseInt(args [0]);
    System.out.println("is " +k +"-Clique: " +q1.kClique(k));
    System.out.println("is " +k +"-Clique: " +q2.kClique(k));
} catch (NumberFormatException e) {
    usage();
    System.exit(-2);
}

}

}

```