RWTH Aachen **SoSe 2007**
Lehrgebiet Theoretische Informatik Blatt 12
Kneis—Mölle—Richter—Rossmanith—Thiemann 2.7.2007

Übungen zur Vorlesung Datenstrukturen und Algorithmen

T32

Beweisen Sie die Korrektheit des Algorithmus von Prim. Es soll angenommen werden, daß der Eingabegraph zusammenhängend ist und seine Kantengewichte durch die Funktion *weight* kodiert sind.

```
procedure Prim(s):
Q ← new Priority−Queue();
for v in V do Q.insert(v) with key ∞; parent[v] ← v od;
Q.decrease_key(s) to 0;
while Q ≠ ∅ do
  v ← Q.extract_min;
  forall u ∈ Q adjacent tov do
    if weight(v,u)<Q.key(u) then Q.decrease_key(u) toweight(v,u); parent[u] ← v fi
  od
od
```

T33

Geben Sie eine Folge von Union- und Find-Operationen an, die zu einem Baum der Höhe vier führt. Verwenden Sie dabei sowohl die Rangheuristik als auch Pfadkompression.

H27 (10 Punkte)

Zeigen oder widerlegen Sie die Korrektheit des folgenden Greedy-Algorithmus zum Finden von minimalen Spannbäumen auf Graphen:

Beginne mit n isolierten Zusammenhangskomponenten, die jeweils einen der Knoten des Eingabegraphen enthalten. Solange es noch zwei Komponenten A, B gibt, verbinde diese mit einer Kante minimalen Gewichts unter allen Kanten zwischen A und B.

H28 (10 Punkte)

Modifizieren Sie die Klasse *Partition* aus der Vorlesung so, daß zusätzlich die Technik *union-by-rank* (also die Rangheuristik) verwendet wird.

```
public class Partition {
  int[] s;
  public Partition(int n) {
    s = new int[n];
    for(int i=0; i<n; i++) s[i]=i;
  }
  public int find(int i) {
    int p=i, t;
    while(s[p] ≠ p) p=s[p];
    while(i ≠ p) { t=s[i]; s[i]=p; i=t; }
    return p;
  }
  public void union(int i, int j) { s[find(i)] = find(j); }
}
```