

Prof. Petra Mutzel
Gunnar Klau
Ivana Ljubic
René Weiskircher

Wintersemester 2000/2001

**Klausur zur Vorlesung
Algorithmen und Datenstrukturen 2
18. Mai 2001**

a.) Machen Sie bitte die folgenden Angaben in deutlicher Blockschrift:

Name: _____ Vorname: _____

Matrikelnummer: _____ Studienkennzahl: _____

- b.) Legen Sie während der Klausur Ihren Studentenausweis vor sich auf das Pult.
- c.) Diese Klausur enthält Multiple-Choice-Fragen. Für jede dieser Aufgaben kann es eine oder mehrere richtige Lösungen geben. Markieren Sie diese **deutlich und unzweideutig** durch Ankreuzen. Alle Multiple-Choice-Aufgaben sind gleich viel wert; für jede richtige Antwort bekommen Sie Pluspunkte, für jedes falsche Kreuz gibt es Minuspunkte.
- d.) Denken Sie daran, dass keinerlei Hilfsmittel erlaubt sind – weder Taschenrechner, irgendwelche Unterlagen, Handys,...

VOR DER ABGABE AUSZUFÜLLEN:

e.) Geben Sie bitte die Anzahl der zusätzlich abgegebenen Blätter an: _____

f.) Kreuzen Sie bitte die von Ihnen bearbeiteten Aufgaben in der ersten Zeile der Tabelle an:

Aufgabe	A 1	A 2	A 3	A 4	A 5	A 6		Note
bearbeitet							—	_____
maximale Punktzahl	6	10	12	6	10	16	60	_____
erreichte Punktzahl								

Viel Erfolg!

Aufgabe 1: Tiefensuche

(6 Punkte)

Betrachten Sie den Graphen in der Abbildung. Auf diesen wird der Algorithmus **Tiefensuche** angewendet, der wie folgt definiert ist:

```

Algorithmus Tiefensuche(Graph G) {
  Liste L = leere List;
  Markiere alle Knoten von G als unbesucht;
  Fuer alle Knoten v von G:
    Falls v unbesucht:
      Besuche(v)
}

```

```

Algorithmus Besuche(Knoten v) {
  Markiere v als besucht;
  Haenge v an das Ende der Liste L;
  Fuer alle Kanten e, die von v ausgehen:
    Sei w Zielknoten von e;
    Falls w nocht nicht besucht:
      Besuche(w);
}

```

Gehen Sie davon aus, dass die Schleife im Algorithmus **Tiefensuche** die Knoten des Graphen in zufälliger Reihenfolge durchläuft. Betrachten Sie nun den unten abgebildeten Graphen. Jeder Aufruf des Algorithmus **Tiefensuche** produziert eine Liste L. Kreuzen Sie unten die Listen an, die möglicherweise als Liste L von dem Algorithmus **Tiefensuche** für den abgebildeten Graphen berechnet werden.

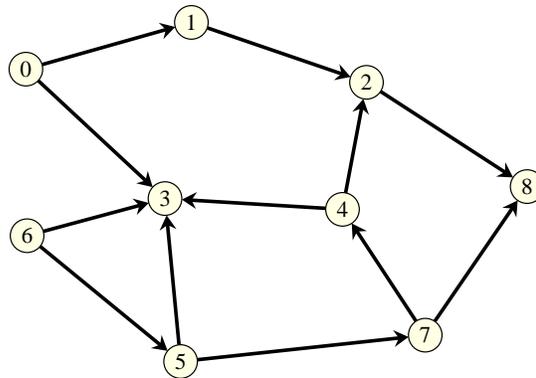


Abbildung 1: Der Graph, auf dem Algorithmus **Tiefensuche** läuft

- a) 0,1,2,8,4,7,3,5,6
 b) 6,5,7,8,3,4,2,0,1
 c) 4,3,2,8,1,0,6,5,7
 d) 0,1,2,8,4,3,6,5,7

Aufgabe 2: Minimum Spanning Tree

(10 Punkte)

Gegeben ist der unten abgebildete Graph mit Kantengewichten. Führen Sie anhand dieses Graphen den Kruskal Algorithmus zum Finden eines minimalen aufspannenden Baumes aus. Verwenden Sie dabei die folgenden Verbesserungen für den Algorithmus:

- a.) Vereinigung nach Höhe des Bäume
- b.) Pfadverkürzung (Pfadkompression)

Zeichnen Sie jeweils die verwendete Datenstruktur nach der Behandlung jeder Kante. Welche Kanten enthält der berechnete Spanning Tree?

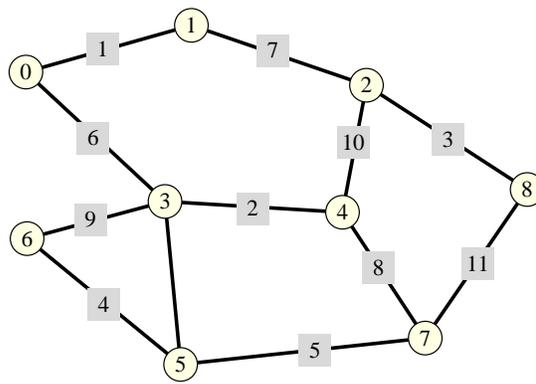


Abbildung 2: Der ungerichtete Graph mit Kantengewichten für das Minimum Spanning Tree Problem

Aufgabe 3: Rucksackproblem

(12 Punkte)

Gegeben ist eine Menge von n Typen von Gegenständen, wobei $size[i]$ das Volumen von Typ i ist und $val[i]$ der Wert von Typ i . Von jedem Typ gibt es unendlich viele Gegenstände. Ferner ist ein Behälter mit Volumen m gegeben. Betrachten Sie den folgenden Algorithmus, der zwei Felder Sum und $Best$ mit Indizes von 1 bis m berechnet. Zu Beginn des Algorithmus enthalten beide Felder nur 0en.

```

Fuer j = 1 bis n
{
  Fuer i = 1 bis m
  {
    Falls i-size[j] >= 0
    {
      Falls Sum[i] < (Sum[i-size[j]]+val[j])
      {
        Sum[i] = Sum[i-size[j]]+val[j]
        Best[i] = j
      }
    }
  }
}

```

- a.) Gegeben sind die Gegenstände 1 bis 5 mit den Größen 3,4,7,8 und 9 und den Werten 4,5,10,11,13. Geben Sie die Inhalte der Felder Sum und $Best$ nach Ausführung des obigen Algorithmus an, falls der Wert m gleich 17 ist.
- b.) Wie sieht die optimal Lösung für das Rucksackproblem in diesem Fall aus, d.h. wieviele Gegenstände von jedem der 5 Typen werden in der optimalen Lösung eingepackt?
- c.) Wie findet man Allgemein nach Ausführung des obigen Algorithmus heraus, wie viele Gegenstände man von welchem Typ einpacken soll?

Aufgabe 4: Simulated Annealing

(6 Punkte)

Im Simulated Annealing Ansatz zur Lösung von schweren Problemen wird jeweils eine neue Lösung zufällig aus der aktuellen Lösung generiert. Ist diese Lösung besser als die aktuelle Lösung, wird sie auf jeden Fall genommen. Abhängig von der aktuellen Temperatur kann auch eine schlechtere Lösung die aktuelle Lösung ersetzen. Die Wahrscheinlichkeit, dass dies geschieht sinkt aber mit der Temperatur.

Nehmen Sie an, der Wert der Lösung soll minimiert werden. Gesucht ist eine Funktion, die die Wahrscheinlichkeit angibt, dass die neue schlechtere Lösung akzeptiert wird. Wenn nun T die aktuelle Temperatur ist,

E der Wert der bisherigen Lösung und E' der Wert der neuen Lösung, welche der folgenden Funktionen ergibt dann eine sinnvolle Wahrscheinlichkeit?



a) $e^{-(E'-E)/T}$



b) $e^{(E'-E)/T}$



c) $e^{-(E-E')T}$



d) $e^{(E-E')/T}$

Aufgabe 5: Asymmetrisches TSP

(10 Punkte)

Gegeben ist ein asymmetrisches Traveling Salesman Problem mit sechs Städten durch die folgende Distanzmatrix:

$$\begin{pmatrix} \infty & 7 & 21 & 9 & 25 & 7 \\ 22 & \infty & 28 & 3 & 29 & 9 \\ 23 & 9 & \infty & 5 & 21 & 15 \\ 32 & 12 & 26 & \infty & 30 & 11 \\ 39 & 10 & 24 & 5 & \infty & 18 \\ 21 & 4 & 33 & 14 & 29 & \infty \end{pmatrix}$$

Finden Sie mit Hilfe des in der Vorlesung beschriebenen Verfahrens eine untere Schranke für eine Tour in dem Graphen. Geben Sie die Schritte des Verfahrens an.

Aufgabe 6: Suchen in Texten

(16 Punkte)

Ein Text ist als ein Feld A von t Buchstaben gespeichert. Der erste Buchstabe ist $A[0]$ und der letzte $A[t-1]$. Es soll nun das erste Vorkommen eines Teilstrings bestimmt werden, der aus n Wiederholungen des Buchstabens z besteht.

- Geben Sie Pseudocode für die Funktion `gibPos(A,z,n)` an. Diese Funktion erhält das Feld A , das Zeichen z und die Anzahl n der Wiederholungen. Ergebnis der Funktion ist der Anfangsindex des ersten Vorkommens des Teilstrings, der aus n Wiederholungen des Zeichens z besteht falls ein solches Vorkommen existiert und ansonsten der Wert t . Ihre Funktion soll für große n und t so effizient wie möglich sein. Sie können davon ausgehen, dass $n \geq 1$ gilt.
- Wie sehen die günstigsten und die ungünstigsten Fälle für Ihren Algorithmus aus, wenn der gesuchte Teilstring *nicht* in A enthalten ist? Geben Sie die Anzahl der Vergleiche Ihrer Prozedur in diesen Fällen an.