Übungen zur Vorlesung Einführung in die Informatik II

Ausgabe: 14. Mai 2002 Abgabe: 21. Mai 2002, 12.00 Uhr

Die Abgabe erfolgt ausschließlich über das System eClaus! Siehe
http://www.informatik.uni-stuttgart.de/ifi/fk/lehre/ss02/info_II_02.html

Die Bearbeitungszeitraum ist wegen des Feiertags an Pfingsten um einen Tag verlängert! Die Übungsgruppen 1, 2, 3, 4 und 5 (Montag 13h15) nehmen bitte an der Übung am Dienstag um 11h50-13h20 im V20.04 teil. Die Übungsgruppen 6, 7, 8, 9 und 11 (Montag 15h00 und 16h45) nehmen bitte an der Übung am Dienstag um 11h50-13h20 im 1.034 teil. Ist Ihnen die Teilnahme an diesen Terminen nicht möglich, so nehmen Sie bitte an den Übungen am Donnerstag um 13h15 im 1.034 oder 16h45 im 1.034 teil oder geben Sie die Votieraufgaben ebenfalls schriftlich im eClaus System ab.

Aufgabe 1: (Packungsalgorithmen) **schriftlich**

Sie haben die Aufgabe, eine gegebene Menge von n Objekten, deren Größe jeweils zwischen 0 und 100 liegt, in Behälter der Kapazität 100 zu füllen. Schreiben Sie ein Ada 95 Programm, das mit folgenden drei Methoden versucht, dabei mit möglichst wenig Behältern auszukommen:

a. (2.5 Punkte) Berechnen Sie mit *Backtracking* die minimal benötigte Behälteranzahl. Achten Sie darauf, dass Sie Berechnungen, die zu keiner Verbesserung mehr führen können, frühzeitig abbrechen.

b. (2.5 Punkte) Die *"First-Fit"-Heuristik* legt ein Objekt in den erstmöglichen Behälter (d.h., sei $B(j)$ der Füllstand des Behälters j, so wähle zu einem Objekt der Größe i den Behälter k mit $k := \min\{j | B(j) + i \leq 100\}$). Ist z.B. der erste Behälter zu 50% und der zweite zu 60% belegt, so würde ein Objekt der Grösse 35 in den ersten Behälter gelegt. Bei einem Objekt der Größe 55 würde ein neuer Behälter begonnen. Schreiben Sie einen Algorithmus, der die Anzahl der benötigten Behälter berechnet, wenn die "First-Fit"-Heuristik verwendet wird.

c. (2.5 Punkte) Bei der *"Best-Fit"-Heuristik* wählt man den Behälter so, dass der verbleibende Platz im gewählten Behälter möglichst gering wird (wähle zu einem Objekt der Größe i den Behälter j mit $B(j) + i$ maximal unter der Bedingung $B(j) + i \leq 100$). Im obigen Beispiel würde das Objekt der Grösse 35 in den zweiten Behälter gelegt. Schreiben Sie einen Algorithmus, der die Anzahl der benötigten Behälter berechnet, wenn die "Best-Fit"-Heuristik verwendet wird.

d. (1 Punkt) Führen Sie ihre Algorithmen für $n = 12$ und für 1000 zufällig erzeugte Beispiele aus und geben Sie aus, wieviel Behälter jedes Verfahren im Mittel benötigt hat.

e. (2.5 Punkte) Untersuchen Sie, welchen Einfluss eine aufsteigende bzw. absteigende Sortierung der Objekte nach ihrer Größe auf das Ergebnis der benötigten Behälteranzahl hat (zur Sortierung können Sie den Algorithmus von Aufgabenblatt 2 verwenden). Hat es einen Einfluss auf die Laufzeit beim Backtracking?

f. (2 Punkte) Konstruieren Sie ein Beispiel, bei dem die "First-Fit"-Heuristik im Vergleich zum Optimum sehr viele Behälter benötigt. Wie viele Behälter benötigt die "Best-Fit"-Heuristik für dieses Beispiel?

g. (1 Punkt) Finden Sie ein Beispiel, bei dem die "Best-Fit"-Heuristik mehr Behälter benötigt als die "First-Fit"-Heuristik.

Aufgabe 2: (Graphalgorithmen) **zum Votieren**

Wiederholen Sie die Grundlagen zu (ungerichteten) Graphen aus der Vorlesung Einführung in die Informatik I - vgl. Skript Lagally, Februar 2002:
http://pcbs13.informatik.uni-stuttgart.de/ifi/bs/lehre/ei1/2001/htm/ei1ws01.htm
Ein Graph $G = (V, E)$ besteht aus einer Menge V von Knoten (vertices) und einer Menge E von Kanten (edges). Zwei Knoten heißen *adjazent*, wenn sie durch eine Kante verbunden sind. Eine Kante e heißt *inzident* zu einem Knoten v, wenn einer der Endpunkte von e der Knoten v ist. Für die *Adjazenzmatrix* $A = (a_{ij})$, $1 \leq i, j \leq n$ gilt $a_{ij} = 1$ gdw. Knoten i und j sind durch eine Kante verbunden. Ein *Weg* ist eine Knotenfolge $w_0 w_1 ... w_k$, wenn w_i und w_{i+1} durch eine Kante verbunden sind. Ein Graph G heißt *zusammenhängend* gdw. für alle Knoten u und v ein Weg von u nach v existiert.

a. (1 Punkt) Entwerfen Sie eine Ada 95 Funktion, die zu einem ungerichteten Graphen $G = (V, E)$, der als Adjazenzmatrix gegeben ist, entscheidet, ob G zusammenhängend ist oder nicht.

b. (1 Punkt) Entwerfen Sie eine Ada 95 Prozedur, die zu einem Knoten u alle von u erreichbaren Knoten ausgibt (also alle Knoten v, für die ein Weg von u nach v existiert) - dies ist die Zusammenhangskomponente von u.

c. (1 Punkt) Sei $G = (V, E)$ ein ungerichteter zusammenhängender Graph. Ein Knoten $v \in V$ heißt *Artikulationspunkt*, wenn durch Entfernen des Knotens v aus G, der Graph G in mehrere Zusammenhangskomponenten zerfällt. Entwerfen Sie einen naheliegenden Algorithmus, der die Artikulationspunkte in G findet. Schätzen Sie den Aufwand des Algorithmus ab.

d. (4 Punkte) Vermutlich hat ihr Algorithmus aus Teil c den Aufwand $|V| * T_a(G)$, wenn T_a der Aufwand des Algorithmus aus Teil a ist.

   Finden Sie eine effizientere Methode mit linearem Aufwand. (**Hinweis:** Wandeln Sie den Algorithmus zur Tiefensuche ab (siehe Skript zur Einführung Informatik I))

Es werden nur 20 Punkte des Aufgabenblattes gewertet. In eClaus gibt es eine Multiple Choice Aufgabe, in der Sie angeben können, ob Sie wenigstens eine Aufgabe votieren wollen. Diese Aufgabe wird dann mit der in der Übung erreichten votierten Punkteanzahl bewertet.

Alle weiteren Informationen zur Vorlesung und den Übungen finden Sie unter

http://www.informatik.uni-stuttgart.de/ifi/fk/lehre/ss02/info_II_02.html