Institut für Computergraphik und Algorithmen

Abteilung für Algorithmen und Datenstrukturen

TECHNISCHE UNIVERSITÄT WIEN

Prof. Petra Mutzel
Günther Raidl
Gunnar Klau
Gabriele Kodydek
René Weiskircher

Sommersemester 2002

Prüfung zur Vorlesung Algorithmen und Datenstrukturen 2 Mai 2002

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

Vorname: _____ Nachname: _____

Matrikelnummer: _____ Studienkennzahl: _____

b) Legen Sie während des Tests Ihren Studentenausweis vor sich auf das Pult.

c) Schreiben Sie die Lösungen direkt auf das jeweilige Aufgabenblatt. Wenn Ihnen das Papier ausgeht, bitten Sie die Aufsicht um Nachschub. Es ist nicht erlaubt, eigenes Papier zu verwenden!

d) Denken Sie daran, dass keinerlei Hilfsmittel erlaubt sind – weder Taschenrechner, irgendwelche Unterlagen, Mobiltelefone etc.

VOR DER ABGABE AUSZUFÜLLEN:

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

Resultat:

Aufgabe	A 1	A 2	A 3	A 4	A5	
maximale Punktzahl	10	10	10	10	10	50
erreicht						

Viel Erfolg!

Aufgabe 1.A: Dynamische Disjunkte Menge **(10 Punkte)**

Im Zusammenhang mit dem Kruskal-Algorithmus zum Finden eines minimalen aufspannenden Baums in einem Graphen wurde die Datenstruktur *Dynamische Disjunkte Menge* (DDM) in der Vorlesung behandelt. Gegeben sei $V = \{1, \ldots, n\}$. Die DDM definiert eine Funktion $f : V \to V$ und drei Operationen mit der folgenden Semantik:

- **Makeset(x):** Setzt $f(x)$ auf x.
- **Findset(x):** Liefert $f(x)$ zurück.
- **Union(x, y):** Falls $f(x) = x$ und $f(y) = y$, so gilt nach Ausführung der Operation dass für alle Elemente z aus V, für die vor der Operation $f(z) = x$ galt (das ist z.B. x), danach $f(z) = y$ gilt. Gilt $f(x) \neq x$ oder $f(y) \neq y$, ist das Ergebnis der Operation undefiniert.

a) Die DDM soll mit Hilfe eines Feldes A mit $A[x] = f(x)$ für alle $v \in V$ implementiert werden. Geben Sie für alle Operationen Pseudocode an. (4 Punkte)

b) Geben Sie für jede der Operationen jeweils die Best Case Laufzeit in Θ-Notation an. (4 Punkte)

c) Geben Sie für die Implementierung in der Vorlesung jeweils die Best Case Laufzeit für die drei Operationen in Θ-Notation an. (2 Punkte)

Aufgabe 2.A: Branch and Bound **(10 Punkte)**

Wir betrachten das Rucksack-Problem für vier verschiedenen Objekte. Die folgende Tabelle gibt jeweils für jedes der Objekte O_1, O_2, O_3 und O_4 ihr Gewicht und ihren Wert an. Der Rucksack kann ein Gewicht von 100 Einheiten transportieren.

	O_1	O_2	O_3	O_4
Gewicht g_i	50	30	25	15
Wert w_i	10	8	6	1

Sei $F = (F_1, \ldots, F_4)$ die Folge der vier Objekte absteigend sortiert nach dem Wert des Quotienten $\frac{w_i}{g_i}$. In einem Branch-and-Bound Algorithmus wird das Problem in Teilprobleme zerlegt, indem man in der durch F festgelegten Reihenfolge die Werte der Entscheidungsvariablen x_i der Objekte fixiert.
Dadurch ergibt sich ein Branch-and-Bound Baum. Für jeden Knoten v des Baums werden eine obere und eine untere Schranke für die im Teilbaum mit Wurzel v enthaltene beste Lösung bestimmt. Seien nun die Variablen x_1 bis x_k für $1 \leq k \leq 4$ schon fixiert, wobei x_i für $1 \leq i \leq k$ den Wert 1 hat, falls Objekt F_i eingepackt wird und 0 sonst.
Die untere Schranke erhält man, indem man alle Objekte, deren Variable noch nicht festgelegt ist, in der durch F gegebenen Reihenfolge durchläuft und den aktuellen Gegenstand einpackt, falls noch Platz im Rucksack ist. Dadurch erhält man eine gültige Lösung für das Problem, deren Wert die untere Schranke ist. Man erhält die obere Schranke, indem man ebenfalls alle Objekte, deren Variable noch nicht festgelegt ist, in der durch F gegebenen Reihenfolge durchläuft. Man packt aber nun alle Genstände ein, bis man zu dem ersten Gegenstand F_i kommt, der nicht mehr in den Rucksack passt. Sei r die noch freie Kapazität des Rucksacks. Dann zählt man $r\frac{g_i}{w_i}$ noch zu dem Wert der Objekte im Rucksack dazu.

a) Zeigen Sie, dass die obere Schranke korrekt ist. (5 Punkte)

b) Zeichnen Sie den Branch-and-Bound Baum für das Problem und schreiben Sie an jeden Knoten den Wert der unteren und oberen Schranke. Nehmen Sie an, dass beim Fixieren einer neuen Variable immer zuerst das Kind betrachtet wird, bei dem die Variable auf 1 gesetzt wird und dass Sie den Baum mittels Tiefensuche durchmustern. Achten Sie darauf, dass Sie nur die Teile des Baums zeichnen, die zur Berechnung der optimalen Lösung nötig sind. (5 Punkte)

Aufgabe 3.A: Simulated Annealing: Nachbarschaftsfunktion **(10 Punkte)**

Sie wollen mit Hilfe von Simulated Annealing in einem Graphen $G = (V, E)$ einen möglichst langen Pfad finden, der jeden Knoten des Graphen höchstens ein Mal besucht. Dazu brauchen Sie eine Nachbarschaftsfunktion, die aus einer gültigen Lösung eine neue gültige Lösung erzeugt. Diese Funktion soll die Bedingung *Erreichbarkeit des Optimums* erfüllen.

a) Geben Sie eine geeignete Nachbarschaftsfunktion in Pseudocode an. Beachten Sie dabei, dass der Graph G nicht unbedingt zusammenhängend ist. (5 Punkte)

b) Zeigen Sie, dass Ihre Funktion bei Eingabe eines Pfades stets wieder eine Pfad produziert, der sich vom Eingabe-Pfad unterscheidet und jeden Knoten von G maximal ein Mal besucht. (2 Punkte)

c) Sei p ein beliebiger Pfad in G. Zeigen Sie, dass man durch mehrmaliges Anwendern Ihrer Nachbarschaftsfunktion aus p einen Pfad maximaler Länge in G konstruieren kann. (3 Punkte)

Aufgabe 4.A: Radix-Trie **(10 Punkte)**

Gehen Sie davon aus, dass T ein Radix-Trie für 5-Bit-Zahlen ist. Also hat jedes in T gespeicherte Element die Form $(b_1, b_2, b_3, b_4, b_5) \in \{0,1\}^5$. In der Wurzel wird nach Bit b_1 unterschieden, in der zweiten Ebene nach Bit b_2 u.s.w.

a) Zeichnen Sie einen Radix-Trie, der die folgenden Elemente enthält:

10110
11001
10101
11110
01110

(4 Punkte)

b) Zeichnen Sie den gleichen Radix-Trie nach Einfügen von 01111. (2 Punkte)

c) Gehen Sie davon aus, dass für jeden Knoten p im Trie der Wert $p.next[0]$ der Zeiger auf den linken Sohn ist und $p.next[1]$ der Zeiger auf den rechten Sohn. Geben Sie Pseudocode für die rekursive Funktion `Einfügen(p,S,i)` an, die Schlüssel S in den Teilbaum mit Wurzel p auf Tiefe i einfügt (zum Einfügen in den gesamten Baum ruft man `Einfügen(wurzel,S,1)` auf). (4 Punkte)

Aufgabe 5.A: Parallele Präfixsummen Berechnung **(10 Punkte)**

a) Gegeben ist die Folge $A = (3, 10, 4, 5, 5, 2, 7, 7)$ von ganzen Zahlen. Geben Sie die Folge S an, die die Präfixsumme von A repräsentiert. 2 Punkte

b) Nehmen Sie an, dass $|A| = n = 2^k$ für $k \in \mathbb{N}$ ist. Geben Sie einen rekursiven parallelen Algorithmus als Pseudocode an, der n Prozessoren benutzt und die Präfixsumme in Zeit $\Theta(\log n)$ berechnet. Nehmen Sie dabei an, dass der Rechner eine CREW-PRAM Maschine ist (Concurrent Read Exclusive Write). 4 Punkte

c) Ist Ihr Algorithmus *effizient* (in dem Sinne, der in der Vorlesung für parallele Algorithmen definiert wurde)? Begründen Sie ihre Antwort. 2 Punkte