Aufgabe 1: Gleichwertige Lösungen

Für die Lösung jeder lösbaren Aufgabe gibt es eine unendliche Anzahl von (abstrakten und konkreten) Algorithmen. Das folgende Problem illustriert, dass eine Aufgabe einfacher oder komplizierter, aber auch "schlechter" oder "besser" gelöst werden kann. Diese Aufgabe ist dem Buch "*A. Solymosi & U. Grude: Grundkurs Algorithmen und Datenstrukturen in Java, Vieweg Verlag*" entnommen. Kernziel der ersten Aufgabe ist, messen und zählen zu lernen.

Aufgabenstellung

Sei `folge` eine endliche Folge von ganzen Zahlen (in Form einer Liste/Reihung) gegeben. Es ist die maximale Teilsumme von `folge` zu berechnen.

Erklärung: Sei `folge:=(5,-8,3,3,-5,7,-2,-7,3,5)` eine Folge von ganzen Zahlen. Dann ist eine **Teilfolge** dieser Folge ein zusammenhängender Teil der Folge. Jede Folge hat zwei extreme Teilfolgen: die leere Teilfolge () und die gesamte Teilfolge (auch als unechte Teilfolge bezeichnet). Teilfolgen wären z.B.: `teilfolge1 := (3,3); teilfolge2 := (-5,7,-2);` Keine Teilfolgen wären `keineteilfolge1 := (5,3,3,7,3,5); keineteilfolge2 := (-8,-5,-2);` insbesondere darf die ursprüngliche Folge nicht umsortiert werden!

Implementieren Sie nun folgende vier Algorithmen:

1. `maxteilsumme3` als **intuitive Lösung**. Erzeugen Sie der Reihe nach alle Teilfolgen der gegebenen Folge und berechnen deren jeweilige Summe. Geben Sie die größte Summe sowie die zugehörige Teilfolge aus.
   Dies läßt sich recht einfach durch entsprechend geschachtelte for-Schleifen lösen (drei geschachtelte Schleifen sollten genügen). In einer Variablen (bzw. anderer Datenstruktur) ist die jeweils aktuell grösste Summe zu speichern, so dass nicht alle Summen gespeichert werden müssen. Die zugehörigen Indizes sind ebenfalls zu speichern.

2. `maxteilsumme`2 als **erste Verbesserung**: In einer (Dreiecks-)Matrix ($n_{x}n$) werden die jeweiligen, bereits berechneten Teilsummen abgespeichert. Zu diesem Zweck wird in einer doppelt geschachtelten Schleife jede Teilsumme (vom Index `i` bis zum Index `j`) berechnet und in der Matrix an der Stelle `[i,j]` abgespeichert. In einer zweiten geschachtelten Schleife wird nun das Maximum aller Summen ermittelt. Bei der Berechnung der Summen vom Index `i` bis zum Index `j` ist, soweit vorhanden, auf die abgespeicherte Summe `[i,j-1]` zuzugreiffen, also etwa `teilsumme[von][bis] = teilsumme[von][bis -1] + folge[bis]`. Dies ist die eigentliche Verbesserung!
3. `maxteilsummeRekursiv` als **rekursive Lösung** und **zweite Verbesserung**: In diesem Algorithmus wird ein Konstruktionsprinzip angewendet, das sich bei der Konstruktion vieler Algorithmen und damit bei der Lösung vieler Probleme bewährt hat. Dieses Prinzip bzw. diese Strategie wird häufig als *Teile-und-Herrsche* bezeichnet und ist mindestens seit der Zeit von Machiavelli (1469-1527) bekannt! Folgende Vorgehensweise legt dieses Prinzip nahe: Man nimmt das gesamte Problem und teilt es in mehrere Teilprobleme ein. Man löst die Teilprobleme evtl. mit der gleichen Vorgehensweise (also rekursiv). Aus der Lösung der Teilprobleme errechnet man eine Lösung für das Gesamtproblem.

Algorithmus zur Berechnung der maximalen Teilsumme einer Folge `folge` nach der Teile-und-herrsche-Strategie:

1. Wenn `folge` nur aus einer Zahl `zahl` besteht, dann nimm das Maximum von `zahI` und `0`.
2. Wenn `folge` aus zwei oder mehr Zahlen besteht dann:
    1. Teile `folge` in zwei etwa gleich große Hälften `linkeHaelfte` und `rechteHaelfte`.
    2. Berechne die maximale Teilsumme `maxLinks` und das rechte Randmaximum `rechtesRandMax` von `linkeHaeIfte`.
    3. Berechne die maximale Teilsumme `maxRechts` und das linke Randmaximum `linkesRandMax` von `rechteHaelfte`.
    4. Das Maximum der drei Zahlen `maxLInks`, `maxRechts` und `rechtesRandMax + linkesRandMax` ist die maximale Teilsumme von `folge`.

4. `maxteilsumme`1 als **optimale Lösung**: Es wird elementeweise von links nach rechts durch die Folge `folge` durchgegangen. Bei jedem Schritt wird für die schon "untersuchte" linke Teilfolge von `folge` (sie beginnt am Anfang von Folge und endet bei dem Element, welches gerade untersucht wird) folgendes berechnet:
    1. die maximale Teilsumme dieser linken Teilfolge: Maximumbildung von bisherigem Maximum und rechtem Randmaximum; und
    2. das rechte Randmaximum dieser linken Teilfolge: Maximumbildung zwischen 0 und der Addition von aktuell untersuchtem Element und bisherigem Randmaximum.

Die Folge ist aus der Datei **`folge.dat`** zu lesen. Die Datei `folge.dat` besteht aus ganzen Zahlen, die jeweils durch ein oder mehrere Blanks getrennt sind. **Diese Datei ist durch ein kleines Programm selbst zu erstellen** (z.B. möglicher Java-Code). Dieses Programm erhält als Eingabe ein Intervall, innerhalb dessen die Zufallszahlen erstellt werden sollen, und die Anzahl der zu erstellenden Zufallszahlen, also z.B. `-100 +100 20`, soll heissen, es sind 20 Zufallszahlen im Intervall [-100;100] zu erstellen. Es kann davon ausgegangen werden, dass nicht mehr als 1000 Zahlen erstellt werden.

Die Algorithmen lesen diese Datei `folge.dat` ein und schreiben ihre Resultate in die Datei `maxteilsummeTest.csv`. Dies ermöglicht, die Daten anschliessend in MS-Excel zu visualisieren. Folgende Meßwerte sind auszugeben:

1. Anzahl eingelesener Elemente
2. Name des Algorithmus
3. Ermitteltes Ergebnis, also die maximale Teilsumme
4. Die beiden Indizes in der Datei
5. Zeitmessung der einzelnen Algorithmen: zu messen ist jeweils, wie lange der Algorithmus für die gesamte Berechnung benötigt hat. Einheit: tausendstel Sekunden (ms). Messen Sie einmal mit der Zählung der Zugriffe und einmal den reinen Algorithmus ohne "Beobachtungscode".
6. Gesamtanzahl der Zugriffe auf die Folge `folge`.
7. Zugriff auf die einzelnen Elemente der Folge. Bei dem Algorithmus `maxteilsumme`2 ist zusätzlich die Gesamtzahl der Matrixzugriffe mit auszugeben.

In der zu erzeugenden Datei `maxteilsummeTest.csv` ist eine mögliche Darstellung gewählt worden, die jedoch auch abgeändert werden kann. Dies kann nach eigenem Ermessen vorgenommen werden, jenachdem, wie in MS-Excel die Daten anschliessend visualisiert werden sollen. Insbesondere sollten für den Punkt 7. die Daten in eine eigene Zeile oder ein

eigenes Datenblatt dargestellt werden, da es sich hier um sehr viele Daten handelt.

Die Tests sind von 1000 an in Schrittweite 1000 bis zu 10000 Zahlen vorzunehmen. Da ein Vergleich es notwendig macht, zumindest in den Schleifen übereinzustimmen, ist der selbst implementierte Code nochmal mit dem in dem Buch vorgegebenem Code Teilsumme abzugleichen! Implementieren Sie den Test so, dass er automatisch ausgeführt werden kann und somit z.B. für eine Vorführung gestartet werden kann. Beachten Sie bitte, dass u.U. die ineffiziente Lösung(-en)sehr lange brauchen! Dies sollte Ihr Testprogramm beachten und sich nicht ggf. an der Stelle "aufhängen". Um die Zeitmessungen vornehmen zu können, sollten Sie einen Code mit dem "Beobachtungscode" erstellen. Sind die Tests damit durchgeführt, sollten Sie den Code kopieren und den überflüssigen "Beobachtungscode" entfernen!

Abnahme

Da die Aufgaben des ADP sehr frühzeitig im WWW zur Verfügung stehen, wird die Abnahme stark auf eine **vorbereitende Arbeit** aufgebaut.

Bis Montag Abend vor Ihrem Praktikumstermin ist eine erste Skizze der Aufgabe als *.pdf Dokument (Dokumentationskopf nicht vergessen!) mir per E-Mail zuzusenden. Ggf. können offene Fragen mit gesendet werden. Die Skizze **muss** grob beschreiben, wie Sie sich die Realisierung denken und den Testaufbau gestalten wollen, z.B. welche Datentypen Sie verwenden wollen, wie und wo welche Zeit gemessen werden soll, wie die Messungen durchgeführt werden sollen etc., d.h. **ein Versuchsaufbau ist anzugeben**! Als erfolgreich wird eine Skizze bewertet, wenn Ihre Kenntnisse bzgl. der gestellten Aufgabe eine erfolgreiche Teilnahme an dem Praktikumstermin in Aussicht stellen.

Am Tag des Praktikums findet nach dem Eingangstest eine Befragung von Teams statt. Die Befragung muss erfolgreich absolviert werden, um weiter am Praktikum teilnehmen zu können. Ist die Befragung nicht erfolgreich, gilt die Aufgabe als nicht erfolgreich bearbeitet. Als erfolgreich wird die Befragung bewertet, wenn Ihre Kenntnisse bzgl. der gestellten Aufgabe zumindest ausreichend sind. Dazu gehört insbesodnere eine ausreichende Kenntnis über Ihren Code (und nicht die Kenntnis über den Kommentar im Code).

Abgabe: Unmittelbar am Ende des Praktikums ist von allen Teams der Code abzugeben. Zu dem Code gehören die Sourcedateien, die ggf. erzeugten *.log etc. Dateien, die während der Tests erzeugt wurden, und eine Readme.txt Datei, in der ausführlich beschrieben wird, wie die Software zu starten ist! Zudem ist der aktuelle Dokumentationskopf abzugeben. Die Abgabe gehört zu den PVL-Bedingungen und ist einzuhalten, terminlich wie auch inhaltlich!

Wird eine Aufgabe nicht erfolgreich bearbeitet gilt die PVL als nicht bestanden. Damit eine Aufgabe als erfolgreich gewertet wird, müssen die Skizze, die Befragung sowie die Abgabe als erfolgreich gewertet werden. Alle gesetzten Termine sind einzuhalten.


Thu 20 Jun 2013 22:53:48
Gratis Counter by GOWEB