Algorithmik-Praktikum SS 2004

Aufgabe 3: Decrease-and-Conquer-Algorithmus *randomizedQuickSelect*

Späteste Abnahme: A, B: 05. Mai 2004 C, D: 12. Mai 2004

Name: ..... Matr-Nr: .....

Datum: …… Unterschrift des Dozenten (wenn bestanden): ……..……

Am Beispiel des Algorithmus *randomizedQuickSelect* soll eine Decrease-and-Conquer-Strategie (auch Prune-and-Conquer genannt) angewendet werden. Simulation eines schnellen nicht-deterministischen Algorithmus.

Aufgabe 1 (Verstehen von *randomizedQuickSelect*)

Gegeben ist eine unsortierte Folge von n Elementen, auf der eine Ordnungsrelation definiert ist. Die Aufgabe besteht darin, das k.-kleinste Element in der Folge zu finden. Ein zufallgesteuerter Algorithmus *randomizedQuickSelect* wird eine mittlere Laufzeit von O(n) haben, wobei der Algorithmus mögliche Entscheidungen zufallgesteuert trifft. Über die Verteilung der Eingabedaten muss keine statistische Annahme gemacht werden. Im Worst Case wird die Laufzeit $O(n^2)$ betragen. Mit dem zufallgesteuerten Algorithmus *randomizedQuickSelect* wird hier ein schneller „nicht-deterministischer" Algorithmus simuliert.

Idee des Algorithmus

Gegeben sei eine unsortierte Folge S von n vergleichbaren Elementen und eine Integerzahl $k \in [1, n]$. Wähle zufällig ein Element x aus S und benutze dieses als Pivot-Element, um S in die drei Subsequenzen L, E und G zu teilen. L speichert alle Elemente von S, die kleiner als x, E alle die gleich x und G alle die größer als x sind. Dieser Schritt heißt **Decrease**- oder auch **Prune**-Schritt. Anhand des Vergleichs von k mit der Kardinalität |L|, |E| und |G| der drei Mengen wird entschieden, mit welchen Parameterwerten für S und k der Algorithmus *randomizedQuickSelect* rekursiv aufgerufen wird. Der Pseudocode der Algorithmus-Idee sieht wie folgt aus:

Algorithmus randomizedQuickSelect(S, k):
  Input:Sequenz S von n vergleichbaren Elementen und Integer $k \in [1, n]$
  Output: Das k-kleinste Element von S
  if n = 1 **then**
    return das (erste) Element von S
  wähle ein Element x zufällig aus S aus
  entferne alle Elemente aus S und speichere sie in den Folgen:
    • L, speichert alle Elemente aus S kleiner als x
    • E, speichert alle Elemente aus S gleich x
    • G, speichert alle Elemente aus S größer als x

  if $k \le |L|$ **then** ... ?
  else if $k \le |L| + |E|$ **then** ... ?
  else randomizedQuickSelect(G, k - |L| - |E|)

Bearbeiten Sie folgende Teilaufgaben:

- Erklären Sie, warum die Brute-Force-Methode zur Lösung des Select-Problems $O(n^2)$ Zeit benötigt!

- Erklären Sie genau die Idee des beschriebenen Algorithmus *randomizedQuickSelect*, und vervollständigen Sie den Pseudocode an den mit Fragezeichen versehenen Stellen. Für jede der drei Fallunterscheidungen findet ein rekursiver Aufruf statt. Wie lauten die Parameter dieser Aufrufe?

- Worin besteht der nicht-deterministische Kern des Algorithmus *randomizedQuickSelect*? An welchen Stellen trifft der Algorithmus „selbst" Entscheidungen, die zu einer sehr schnellen Lösung des Select-Problems führen? Bitte genau erklären!!

- Führen Sie den Algorithmus von Hand auf Papier für die folgende Menge S und k = 9 aus. Untersuchen Sie jeweils für die Wahl von x die Fallunterscheidungen, und schreiben Sie jeden Rekursionsaufruf mit den geeigneten Parametern auf.

  S = { 3, 9, 15, 8, 5, 17, 10, 1, 13, 6, 20, 11, 15, 25, 29, 2, 3, 14, 4, 3}

  Zur Konstruktion der beiden rekursiven Aufrufe versuchen Sie, Antworten auf folgende zwei Fragen zu finden:

  - Welcher Zusammenhang besteht zwischen der Größe der Menge L und dem k.-kleinsten Element von S?
  - Was kann man über das k.-kleinste Element aussagen, wenn k = |L| + |E|?

- Diskutieren Sie die Laufzeit-Performance des Algorithmus *randomizedQuickSelect*.

Aufgabe 2 (Programmieren, Testen und Analysieren von *randomizedQuickSelect*)

Programmieren Sie den Algorithmus *randomizedQuickSelect* und geben Sie in jedem Rekursionsschritt die Mengen L, E und G sowie deren Kardinalität auf dem Bildschirm aus.

- Erzeugen Sie die Startmenge S mit Hilfe eines Zufallszahlengenerators. Folgende Parameter zur Erzeugung von S geben Sie interaktiv ein
  - die Größe von S
  - den Wertebereich von S, z.B. [10, 8888]
  - den Wert k
- Geben Sie das Pivotelement $x \in S$ vor jeder Aufteilung S in die Mengen L, E und G, d.h. in jedem Rekursionsschritt interaktiv ein

- Der Ablauf des Algorithmus *randomizedQuickSelect* soll Schritt für Schritt am Bildschirm verfolgt werden. Geben Sie dazu alle relevanten Daten in übersichtlicher Form aus. Halten Sie dazu den Algorithmus nach jedem Rekursionsschritt an und setzen Sie Verarbeitung durch eine Tastenbetätigung und nach Eingabe eines Wertes für x fort. Muss x ein Element aus S sein?

- Führen Sie einen experimentellen Ratio-Test für *randomizedQuickSelect* durch und zeichnen Sie die Laufzeiten mit einem Plot-Programm. Betrachten Sie dabei sowohl die nicht-deterministische Komponente des Algorithmus, als auch den gesamten Algorithmus mit allen (auch deterministischen) Methoden.

- Dokumentieren Sie Ihre Ergebnisse und Erkenntnisse ausführlich!