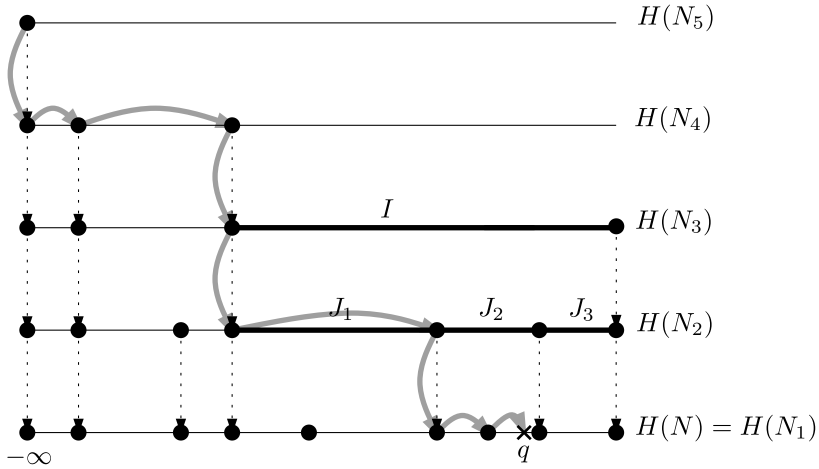
2. Übungsblatt

Abgabe: Montag, 8. November 2004, *zu Beginn* der Vorlesung

Aufgabe 1



Ein Anfragepunkt q soll in der Partition $H(N)$ lokalisiert werden. Dazu steht folgende Datenstruktur zur Verfügung: Die Partition $H(N) = H(N_1)$ besteht aus den Intervallen, die durch die Menge der paarweise verschiedenen Eingabepunkte N sowie einem künstlichen Punkt $-\infty$ induziert werden. Die Partition $H(N_2)$ entsteht aus $H(N_1)$ indem für jeden echten Punkt aus N_1 eine faire Münze geworfen wird. Ist das Ergebnis dieses Wurfs 'Zahl', so ist der Punkt auch in N_2. Jeder Punkt der Partition $H(N_2)$ erhält einen Zeiger auf den ihm entsprechenden Punkt in $H(N_1)$. Die Partition $H(N_3)$ geht dann dementsprechend aus $H(N_2)$ hervor, *usw*. $H(N_r)$ bezeichne die Partition, die nur noch $-\infty$ enthält während $H(N_{r-1})$ noch mindestens einen echten Punkt enthält. Nehmen Sie an, dass jede Partition als geordnete verkettete Liste vorliegt. Eine Suchanfrage lokalisiert nun für jede Partition $H(N_i)$ das Intervall I_i, in dem q liegt. Für $H(N_r)$ ist dies trivial, da $H(N_r)$ nur aus einem Intervall besteht. Ist I_i gefunden, wird vom Anfangspunkt von I_i in die Partition $H(N_{i-1})$ abgestiegen und per inkrementeller Suche in der verketteten Liste von $H(N_{i-1})$ das Intervall I_{i-1} bestimmt. Die Antwort auf die Suchanfrage ist schliesslich I_1.

Als *Kinder* eines Intervalls $I \in H(N_i)$ bezeichnen wir gerade die Intervalle "unter" I in der Partition $H(N_{i-1})$. In der Abbildung sind J_1,J_2 und J_3 die Kinder von I.

Zeigen Sie:

(a) Der erwartete Speicherplatz zum Speichern aller Partitionen ist linear in $n = |N|$.

(b) Die Anzahl r der Partitionen ist in $\tilde{O}(\log n)$.

(c) Die erwartete Anzahl Kinder jedes Intervalls ist konstant.

(d) Eine Suchanfrage benötigt $O(\log n)$ erwartete Zeit. (Tatsächlich ist eine Suchanfrage sogar in $\tilde{O}(\log n)$.)

Aufgabe 2

Gegeben sei ein roter Korb und $k-1$ grüne Körbe. Sie werfen nacheinander Bälle (zufällig und gleichverteilt) in die Körbe.

(a) Wieviele Bälle (Erwartungswert) befinden sich nach n Würfen im roten Korb?

(b) Nach wievielen Würfen ist der Erwartungswert für die Anzahl der Bälle im roten Korb mindestens eins?

(c) Wie oft müssen sie werfen (Erwartungswert), bis ein Ball im roten Korb liegt?
Hinweis: Sei X die Zufallsvariable für die Anzahl der Würfe bis der erste Ball im roten Korb gelandet ist. Berechnen Sie $\Pr(X = i)$ und $E(X)$. Formulieren sie dabei den Erwartungswert zunächst in Abhängigkeit von $\sum_{i=1}^{\infty} \left(\frac{k-1}{k}\right)^i$.

Aufgabe 3

Sei R ein Array mit n paarweise verschiedenen Einträgen. Für $i \in \{1, \ldots, n\}$ bezeichne $R[i]$ den i-ten Eintrag von R. Betrachten Sie den folgenden Algorithmus RANDOMSORT.

Algorithmus 1 RANDOMSORT(R, n)

$S = \{1, \ldots, n\}$, Liste $L = \emptyset$
i = random(S), $S = S \setminus \{i\}$
L.push_back$(R[i])$
solange $S \neq \emptyset$ **führe aus**
  i = random(S), $S = S \setminus \{i\}$
  falls $R[i] < L$.first() **dann** $L = L$.push_front$(R[i])$
  falls $R[i] > L$.last() **dann** $L = L$.push_back$(R[i])$
  falls L.first() $< R[i] < L$.last() **dann gib** 'kein Ergebnis' **zurück**
ende
gib L **zurück**

Wir nehmen an, dass die Bereitstellung einer Zufallszahl und das Verwalten der Menge S in konstanter Zeit möglich ist. Der Algorithmus gibt entweder die sortierte Liste zurück oder liefert kein Ergebnis.

(a) Berechnen Sie die Wahrscheinlichkeit, dass RANDOMSORT tatsächlich die sortierte Liste zurückgibt.

(b) Bestimmen Sie die erwartete Anzahl der Durchläufe der *solange*-Schleife.

(c) Mit dem Ergebnis aus (b) hat RANDOMSORT eine erwartete Laufzeit von $O(1)$. Damit hat n-maliges Ausführen des Algorithmus (n-RANDOMSORT) eine erwartete Laufzeit von $O(n)$. Bestimmen Sie die Wahrscheinlichkeit, dass n-RANDOMSORT wenigstens einmal erfolgreich sortiert. Zeigen Sie die Ineffizienz des Algorithmus, indem Sie das größte n angeben, für das n-RANDOMSORT noch mit einer Wahrscheinlichkeit von grösser als 50% (1%) ein Ergebnis liefert.