Algorithmik Blatt 3 Teil 2

Mtr.-Nr. 6329857

Universität Hamburg — 5. November 2019

Aufgabe 8

```
procedure FindPairs(A):
    for i=1 to A.length do
        for j=i+1 to A.length do
            if A[i] > A[j] then // Bedingung A
                output (A[i], A[j]);
```

$A_{i,j}$ sei dass Ereignis, dass die Bedingung A in Iteration (i, j) erfüllt ist. Wir nehmen an, dass die Ereignisse $A_{i,j}$ für alle i, j unabhängig voneinander sind und interessieren uns für die Vereinigung aller Ereignisse, wobei n die Länge des Array ist:

$$A(n) = \bigcup_{i=1}^{n} \bigcup_{j=i+1}^{n} A_{i,j} \tag{1}$$

Indikatorfunktion für Ereignis $A_{i,j}$:

$$I\{A_{i,j}\} = \begin{cases} 1 & \text{wenn } A[i] > A[j] \\ 0 & \text{cases} \end{cases} \tag{2}$$

$$\begin{aligned} E\left[A_{i,j}\right] &= Pr(A_{i,j}) \\ E\left[A(n)\right] &= E\left[\bigcup_{i=1}^{n} \bigcup_{j=i+1}^{n} A_{i,j}\right] = \sum_{i=1}^{n} \sum_{j=i+1}^{n} E\left[A_{i,j}\right] \end{aligned} \tag{3}$$

Da die Werte im Array paarweise verschieden und zufällig verteilt sind, können wir für $Pr(A_{i,j}) = \frac{1}{2}$ annehmen.

$$\begin{aligned} E\left[A(n)\right] &= \sum_{i=1}^{n} \sum_{j=i+1}^{n} \frac{1}{2} \\ &= \sum_{i=1}^{n} \left(\sum_{j=1}^{n} \frac{1}{2} - \sum_{j=1}^{i} \frac{1}{2} \right) \\ &= \sum_{i=1}^{n} \left(\frac{1}{2} \cdot n - \frac{1}{2} \cdot i \right) \end{aligned} \tag{4}$$

$$
\begin{aligned}
E[A(n)] &= \frac{1}{2} \cdot \sum_{i=1}^{n} (n-i) \\
&= \frac{1}{2} \cdot \sum_{i=1}^{n} (n-i) \\
&= \frac{1}{2} \cdot \left(\sum_{i=1}^{n} n - \sum_{i=1}^{n} i \right) \\
&= \frac{1}{2} \cdot \left(n^2 - \frac{n^2+n}{2} \right) \\
&= \frac{n^2-2n}{4}
\end{aligned}
\tag{5}
$$

Die Erwartete Anzahl an Ausgaben bei einem Array der Länge n ist $\frac{n^2-2n}{4}$

Aufgabe 9

Es ist eine gdw-Aussage zu zeigen, also sind beide Richtungen zu zeigen.

Beweis Richtung 1: Pfad-Existenz ⇒ Knoten-Eigenschaft

Jeder Knoten, der als Zielknoten einer Kante im Pfad vorkommt, muss in der darauf folgenden Kante als Ursprungsknoten vorkommen. Der Zielknoten der letzten Kante muss als Ursprungsknoten der ersten Kante vorkommen. Daraus folgt, dass jeder Knoten genau so oft als Ursprungs wie als Zielknoten in den Kanten des Pfades vorkommen muss. Zusammen mit der Forderung, dass der Pfad genau so viele Kanten wie der gesamte Graph enhält und keine Kanten doppelt enthalten darf, ergibt sich, dass er genau alle Kanten enthält. Da somit in der Menge aller Kanten jeder Knoten genau so oft als Ursprungs wie als Zielknoten vorkommt, muss für alle Knoten v gelten, dass $\text{outdeg}(v) = \text{indeg}(v)$ ist. Aus der Forderung, dass der Graph zusammenhängend ist, folgt dass für alle Knoten v $deg(v) \geq 1$ ist, denn sobald ein Pfad existiert (der Graph ist weder leer noch besteht er aus nur einem kantenlosen Knoten), muss jeder Knoten erreichbar sein, also mindestens eine eingehende Kante besitzen.

Beweis Richtung 2: Pfad-Existenz ⇐ Knoten-Eigenschaft

Wir erinnern uns, dass der Graph (a) streng zusammenhängen ist und (b) jeder Knoten genau so viele eingehende wie ausgehende Kanten hat und (c) mindestens je eine ausgehende und eingehende Kante hat.

Unter diesem Annahmen können wir einen Kantendefinierten Pfad, der alle Kanten enthält wie folgt konstruieren:

1. Wähle einen beliebigen Startknoten
2. Wähle eine ausgehende Kante (wir wissen wegen (c), dass eine existiert), gehe über sie zum Nachbarknoten und markiere sie als benutzt.
3. Gehe so von Knoten zu Knoten ohne eine bereits benutzte Kante erneut zu verwenden.

Wir wissen durch (b), dass wir jeden Knoten, den wir erreichen, auch wieder verlassen können. Wir wissen auch, dass wir den Startknoten wieder erreichen können, weil der Graph streng zusammenhängend ist und der Startknoten für jedes mal, dass wir ihn verlassen auch noch eine unbenutzte Eingangskante für die Rückkehr besitzt. An jedem Knoten an dem wir die Wahl zwischen mehreren Ausgangskanten haben, wissen wir, dass wenn wir eine wählen, auch wieder zurückkehren können um später die andere zu wählen, weil es zu dieser auch eine Eingangskante geben muss, um b zu erfüllen.

Es kann passieren, dass wir die Kanten so wählen, dass wir zum Startknoten zurückgelangen ohne alle Kanten besucht zu haben und dabei auch schon alle seine Ausgangskanten verbraucht zu haben. Auf Grund von (a) wissen wir aber, dass die nicht besuchten Kanten prinzipiell vom Startknoten aus erreichbar gewesen wären. Wir hätten also an einem der besuchten Knoten k eine andere Kante wählen können. Hätten wir das getan, könnten wir auf Grund von (a) und (b) auch wieder zu k zurückkehren. Somit lägen alle nicht erreichten Kanten auf Zyklen die wir prinzipiell hätten mitnehmen können.

Somit ist per Konstruktion gezeigt, dass wenn der Graph die geforderten Eigenschaften erfüllt, der gefragte Pfad konstruieren kann.