

# Algorithmik Blatt 3 Teil 2

Mtr.-Nr. 6329857

Universität Hamburg — 6. November 2019

## 1 Aufgabe 8

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

2  $A_{i,j}$  sei das Ereignis, dass die Bedingung  $A$  in Iteration  $(i, j)$  erfüllt ist. Wir nehmen an, dass die  
3 Ereignisse  $A_{i,j}$  für alle  $i, j$  unabhängig voneinander sind und interessieren uns für die Vereinigung  
4 aller Ereignisse, also für das Gesamtereignis, dass mindestens eines der Teilereignisse eintritt, wobei  
5  $n$  die Länge des Array ist:

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

6 Wir definieren die Indikatorfunktion für  $A(n)$ :

$$I\{A(n)\} = \begin{cases} 1 & \text{wenn } \bigcup_{i=1}^n \bigcup_{j=i+1}^n A_{i,j} \\ 0 & \text{sonst} \end{cases} \quad (2)$$

7 Und bestimmen nun den zugehörigen Erwartungswert:

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

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

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

10

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

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

## 12 Aufgabe 9

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

### 14 Beweis Richtung 1: Pfad-Existenz $\Rightarrow$ Knoten-Eigenschaft

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

### 26 Beweis Richtung 2: Pfad-Existenz $\Leftarrow$ Knoten-Eigenschaft

27 Wir erinnern uns, dass der Graph (a) streng zusammenhängend ist und (b) jeder Knoten genau so vie-  
28 le eingehende wie ausgehende Kanten hat und (c) mindestens je eine ausgehende und eingehende  
29 Kante hat.

30 Unter diesen Annahmen können wir einen kantendefinierten Pfad, der alle Kanten enthält, wie folgt  
31 konstruieren:

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

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

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

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