Eidgenössische
Technische Hochschule
Zürich

Ecole polytechnique fédérale de Zurich
Politecnico federale di Zurigo
Federal Institute of Technology at Zurich

Department Informatik
Markus Püschel
Peter Widmayer
Thomas Tschager
Tobias Pröger
Tomáš Gavenčiak

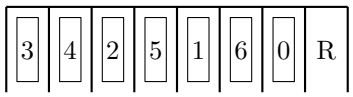
27. Oktober 2016

Datenstrukturen & Algorithmen Blatt P6 HS 16

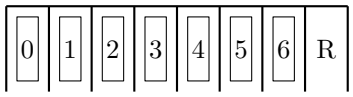
Abgabe: Bis zum 3. November 2016 um 10 Uhr auf dem Judge (ausschliesslich Quellcode).

Aufgabe P6.1 *Chinese wall parking.*

Entlang der Chinesische Mauer ist ein sehr langer Parkplatz, dessen Stellplätze ständig voll besetzt sind. Ein Reserveplatz am rechten Ende des Parkplatzes wird jedoch, wie im folgenden Bild dargestellt, ständig freigehalten.



Die n Autos sind mit den Zahlen 0 bis $n-1$ durchnummeriert und wurden offensichtlich an beliebigen Stellplätzen abgestellt. Behördliche Vorschriften verlangen jedoch, dass die Autos nach deren Nummer geordnet von links nach rechts abgestellt werden müssen, wobei der Reserveplatz frei bleiben muss (wie im folgenden Bild).



Sie sollen die Autos nun so umstellen, dass die Vorschriften erfüllt werden. Gegeben seien die Anzahl der Autos $n \geq 1$ und ein Array $a_0 \dots a_{n-1}$ mit den Nummern der Autos gemäss der ursprünglichen Parkposition von links nach rechts. Jede der Nummern $0 \dots (n-1)$ der Autos kommt genau einmal vor, d.h. Auto i soll immer an Position i geparkt werden (für i von 0 bis $n-1$).

Berechnen Sie die *minimale* Anzahl an Bewegungen der Autos, die nötig sind, um die Autos wie vorgeschrieben abzustellen. Mit *Bewegung* ist das Umparken eines einzigen Autos von seinem aktuellen Stellplatz zu einem freien Stellplatz gemeint (Sie dürfen auch den Reserveplatz benutzen). Beachten Sie, dass es nach einer Bewegung immer genau einen freien Stellplatz gibt. Es ist nicht erlaubt, das Auto während der Bewegung auf der Strasse vorübergehend abzustellen. Es ist nicht relevant wie weit ein Auto während einer Bewegung bewegt wird.

Eine mögliche kürzeste Abfolge von acht Bewegungen mit allen Zwischenschritten ist in folgender Abbildung gegeben. Das Zeichen _ bezeichnet den jeweils freien Stellplatz.

```
3 4 2 5 1 6 0 _   bewege Auto 3
_ 4 2 5 1 6 0 3   bewege Auto 0
0 4 2 5 1 6 _ 3   bewege Auto 6
0 4 2 5 1 _ 6 3   bewege Auto 5
```

```
0 4 2 _ 1 5 6 3   bewege Auto 3
0 4 2 3 1 5 6 _   bewege Auto 1
0 4 2 3 _ 5 6 1   bewege Auto 4
0 _ 2 3 4 5 6 1   bewege Auto 1
0 1 2 3 4 5 6 _   fertig
```

Beachten Sie, dass Sie nicht eine solche Abfolgen von Bewegungen konstruieren, sondern lediglich die minimale Anzahl an benötigten Bewegungen berechnen müssen. Es kann mehrere mögliche kürzeste Abfolgen von Bewegungen geben.

Eingabe Die Eingabe besteht aus zwei Zeilen. Die erste Zeile enthält lediglich die Ganzzahl n. Die zweite Zeile enthält n Ganzzahlen a_0 bis a_{n-1} (die Nummern der Autos), durch Leerzeichen getrennt.

Ausgabe Die Ausgabe soll lediglich eine Zahl, nämlich die minimale Anzahl an Bewegungen, um die Autos zu ordnen, enthalten.

Bonus Sie erhalten einen Bonuspunkt pro 100 Punkte auf dem Judge (abgerundet). Insgesamt können Sie bis zu 200 Punkte auf dem Judge erhalten. Damit alle Tests auf dem Judge erfolgreich sind, sollte die Laufzeit Ihres Programms in $\mathcal{O}(n)$ liegen (eine ausreichend effiziente Implementierung vorausgesetzt).

Senden Sie Ihr `Main.java` unter folgendem Link ein: `https://judge.inf.ethz.ch/team/websubmit.php?cid=18985&problem=DA_P6.1`. Das Passwort für die Einschreibung ist "`quicksort`".

Beispiele

Eingabe (Beispiel aus der Abbildung oben):

```
7
3 4 2 5 1 6 0
```

Ausgabe:

```
8
```

Eingabe (Autos bereits sortiert):

```
5
0 1 2 3 4
```

Ausgabe:

```
0
```

Hinweise Wir stellen für diese Aufgabe eine Programmvorlage als Eclipse Projektarchiv auf der Vorlesungswebseite zur Verfügung. In der Vorlage wird die Eingabe bereits eingelesen. Das Archiv enthält weitere Beispiele – Sie können diese als Eingabe für Ihr Programm verwenden und die Ausgabe überprüfen.