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

15. Dezember 2016

# Algorithmen & Datenstrukturen — Blatt P13 — HS 16

**Abgabe:** Bis zum 22. Dezember 2016 um 10 Uhr auf dem Judge (ausschliesslich Quellcode).

**Aufgabe P13.1** *Dschungel.*

*Mitten im dichten Dschungel schauen Sie nervös auf Ihre Landkarte. Ein Sturm kommt auf, und Sie müssen das Gebiet so schnell wie möglich verlassen...*

Auf Ihrer Landkarte ist der Dschungel in ein quadratisches Raster mit $n \times n$ Feldern unterteilt. Für jedes Feld mit Koordinaten $0 \leq i, j < n$ ist in der Landkarte die Zeit $t_{i,j}$ eingezeichnet, die man benötigt um sich zu diesem Feld zu bewegen. (Der Einfachheit halber benötigt man nur eine gewisse Zeit, um das Feld *zu betreten* und die Richtung beeinflusst die Zeit nicht). Ihr Startfeld ist durch $t_{i_0,j_0} = 0$ definiert und es gibt genau ein Feld mit dieser Zeit. Für alle anderen Felder gilt, dass $1 \leq t_i \leq 10\,000$. Die Grösse der Landkarte ist $1 \leq n \leq 10\,000$.

Sie können sich von jedem Feld in vier verschiedene Richtungen zu einem benachbarten Felder bewegen. Wenn Sie zu einem Feld gelangen, dass am Rand der Landkarte ist (d.h. an mindestens einem der vier Ränder kein benachbartes Feld auf der Landkarte hat), können Sie den Dschungel sofort verlassen und Ihre Reise endet. Die Zeit, die Sie für Ihre gesamte Reise benötigen ist die Summe der Zeiten, die Sie benötigen, um die Felder entlang Ihrer Route zu betreten. Für das Startfeld benötigen Sie Zeit 0.

Ihre Aufgabe besteht darin, die *kleinstmögliche benötigte Zeit* zu berechnen, um den Dschungel ausgehend vom Startfeld zu verlassen. Die Anzahl der Felder, die Sie besuchen, ist nicht relevant; es zählt lediglich die benötigte Zeit. Beachten Sie, dass es mehrere schnellste Routen mit derselben benötigten Zeit geben kann. Wir interessieren uns lediglich für die Zeit, die für eine solche schnellste Route benötigt wird.

**Beispiel** In den Beispielen unten sind zwei Landkarten mit $n = 8$ und $n = 12$ dargestellt. In grau ist jeweils ein schnellster Pfad dargestellt mit benötigter Zeit 10 im linken und 236 im rechten Beispiel.

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 9 | 1 | 1 | 9 | 1 | 1 | 9 | 1 |
| 9 | 1 | 9 | 9 | 9 | 9 | 1 | 9 |
| 9 | 1 | 1 | 9 | 1 | 1 | 1 | 9 |
| 9 | 1 | 1 | 9 | 1 | 1 | 9 | 9 |
| 9 | 1 | 9 | 1 | 1 | 0 | 9 | 9 |
| 9 | 1 | 1 | 1 | 9 | 1 | 9 | 9 |
| 9 | 1 | 9 | 1 | 9 | 9 | 9 | 1 |
| 9 | 9 | 1 | 9 | 1 | 9 | 1 | 42 |

| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 65 | 81 | 23 | 82 | 98 | 41 | 68 | 48 | 85 | 96 | 17 | 22 |
| 71 | 10 | 34 | 65 | 29 | 39 | 28 | 21 | 50 | 66 | 14 | 86 |
| 41 | 71 | 44 | 79 | 13 | 78 | 66 | 56 | 12 | 13 | 36 | 29 |
| 85 | 50 | 99 | 87 | 10 | 94 | 79 | 31 | 14 | 20 | 65 | 91 |
| 35 | 98 | 41 | 27 | 55 | 23 | 26 | 18 | 63 | 41 | 77 | 50 |
| 53 | 21 | 83 | 47 | 71 | 10 | 91 | 38 | 60 | 41 | 19 | 68 |
| 74 | 93 | 58 | 70 | 43 | 24 | 0 | 92 | 96 | 38 | 33 | 93 |
| 86 | 26 | 51 | 63 | 64 | 44 | 13 | 91 | 57 | 39 | 27 | 63 |
| 73 | 39 | 48 | 27 | 85 | 50 | 72 | 72 | 70 | 67 | 52 | 35 |
| 34 | 14 | 81 | 95 | 61 | 80 | 68 | 65 | 17 | 96 | 11 | 31 |
| 49 | 72 | 22 | 96 | 40 | 99 | 94 | 38 | 15 | 37 | 74 | 50 |
| 97 | 58 | 80 | 35 | 77 | 72 | 71 | 41 | 78 | 27 | 19 | 38 |

**Eingabe** Die Eingabe besteht aus mehreren Tests. Die erste Zeile enthält die Anzahl der Tests, die folgen.

Jeder Test beginnt mit $n$ in einer eigenen Zeile. Die $n$ folgenden Zeilen enthalten je $n$ Zahlen. Die $i$-te Zeile enthält die Ganzzahlen $t_{i,j}$ für $j = 0, \ldots, n-1$, durch Leerzeichen getrennt.

**Ausgabe** Geben Sie für jeden Test in einer separaten Zeile aus, wie schnell man den Dschungel verlassen kann.

**Beispiel**

*Eingabe (die Beispiele oben)*

```
2
8
9 1 1 9 1 1 9 1
9 1 9 9 9 9 1 9
9 1 1 9 1 1 1 9
9 1 1 9 1 1 9 9
9 1 9 1 1 0 9 9
9 1 1 1 9 1 9 9
9 1 9 1 9 9 9 1
9 9 1 9 1 9 1 42
12
65 81 23 82 98 41 68 48 85 96 17 22
71 10 34 65 29 39 28 21 50 66 14 86
41 71 44 79 13 78 66 56 12 13 36 29
85 50 99 87 10 94 79 31 14 20 65 91
35 98 41 27 55 23 26 18 63 41 77 50
53 21 83 47 71 10 91 38 60 41 19 68
74 93 58 70 43 24 0 92 96 38 33 93
86 26 51 63 64 44 13 91 57 39 27 63
73 39 48 27 85 50 72 72 70 67 52 35
34 14 81 95 61 80 68 65 17 96 11 31
49 72 22 96 40 99 94 38 15 37 74 50
97 58 80 35 77 72 71 41 78 27 19 38
```

*Ausgabe*

```
10
236
```

**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 $\mathcal{O}(n^2 \log n)$ liegen. Falls Sie einen Heap verwenden wollen, können Sie ihn entweder selbst implementieren oder beispielsweise `java.util.TreeSet<>` oder `java.util.PriorityQueue<>`[1] benutzen.

Senden Sie Ihr `Main.java` unter folgendem Link ein: `https://judge.inf.ethz.ch/team/websubmit.php?cid=18985&problem=DA_P13.1`. Das Passwort für die Einschreibung ist "`quicksort`".

**Hinweise** Wir stellen für diese Aufgabe eine Programmvorlage als Eclipse Projektarchiv auf der Vorlesungswebseite zur Verfügung. In der Vorlage wird die Eingabe bereits eingelesen.

[1] Für diese Bibliothek müssen Sie eine eigene `class` für die Heap-Elemente erstellen und die Klasse entweder als `Comparable` deklarieren oder einen `Comparator` implementieren. Beachten Sie, dass `PriorityQueue` langsam beim Entfernen von einem Element ist, das sich nicht am Anfang der Queue befindet. Sie sollten die Reihenfolge der Elemente nur durch Einfügen und Entfernen der Elemente verändern. Implementieren Sie Ihren eigenen, massgeschneiderten Heap, falls Sie mit den Details zur Verwendung der Bibliotheken nicht vertraut sind.

Das Archiv enthält weitere Testdaten, damit Sie lokal testen können. Ausserdem stellen wir zusätzlich ein `Judge.java` Programm zur Verfügung, das Ihr Programm `Main.java` mit allen verfügbaren Testdaten testet – öffnen und starten sie dazu einfach `Judge.java` in derselben Weise wie Sie `Main.java` starten würden.