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

10. November 2016

Datenstrukturen & Algorithmen — Blatt P8 — HS 16

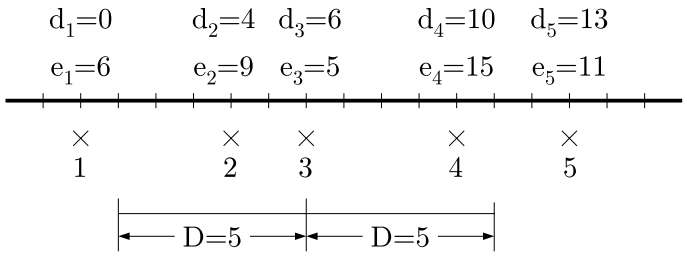
Abgabe: Bis zum 17. November 2016 um 10 Uhr auf dem Judge (ausschliesslich Quellcode).

Aufgabe P8.1 *Windräder.*

Entlang einer Strasse sollen Windräder zur Stromgewinnung aufgestellt werden. Aufgrund geographischer Gegebenheiten kommen n verschiedene Positionen in Betracht. Gesetze schreiben vor, dass zwischen zwei Windrädern ein Mindestabstand von D Metern eingehalten werden muss.

Die n möglichen Positionen sind als Distanzen $d_1, \ldots, d_n$ vom Anfang der Strasse in Meter gegeben, d.h. als Koordinaten auf einer Linie, sodass $0 \leq d_i < d_{i+1}$ für alle $i \in \{1, \ldots, n-1\}$. Wenn ein Windrad an der Position i aufgestellt wird, dann produziert es Energie e_i, wobei alle Werte $e_1, \ldots, e_n$ gegeben sind.

Die Aufgabe besteht nun darin, eine Positionierung von Windrädern mit maximaler gesamter Energieausbeute (Summe der produzierten Energie) zu finden. Die Anzahl der Windräder, die gebaut werden können, ist nur durch die möglichen Positionen n und den Mindestabstand beschränkt.



Beispiel In der obigen Abbildung ist eine Situation für $n = 5$ mögliche Positionen dargestellt. Wird beispielsweise ein Windrad an der Position 3 aufgestellt, dann können an den Positionen 2 und 4 keine Windräder aufgestellt werden. Werden die Windräder an den Positionen 1, 3 und 5 aufgestellt, dann werden $6 + 5 + 11 = 22$ Einheiten Energie produziert. Dies ist aber nicht die optimale Lösung: Eine Installation auf den Positionen 2 und 4 erzeugt $9 + 15 = 24$ Einheiten Energie.

Mehrere Tests Von nun an wird Ihr Programm üblicherweise mehrere unabhängige *Tests* innerhalb eines *Testsets*[1] lösen müssen, damit Sie für dieses Testset Bonuspunkte bekommen. Beachten Sie die Beschreibung der Eingabe und das Beispiel unten. Die Code-Vorlage wird Ihnen das Einlesen der Daten erleichtern.

[1] Die *Testsets* sind wie folgt benannt: `example`, `judge1`, `judge2`, ... Ein *Test* ist eine Teilaufgabe eines solchen Testsets.

Beachten Sie, dass Ihr Programm *alle C* Tests eines Testsets lösen muss, damit Sie Bonuspunkte für dieses Testset bekommen. Sie können davon ausgehen, dass einige der Tests, die zu einem Testset zusammengefasst sind, Randfälle und spezielle (aber gültige) Eingaben testen werden. Beispielsweise würde ein Programm, das grosse zufällige Eingaben lösen kann, aber einige spezielle kleine Eingaben nicht korrekt lösen kann, keine Punkte für das entsprechende Testset bekommen.

Eingabe Die Eingabe besteht aus verschiedenen Tests. Die erste Zeile enthält lediglich die Ganzzahl $C > 0$ der Tests, die folgen.

Jeder Test ist unabhängig von den anderen und besteht aus drei Eingabezeilen: Die erste Zeile enthält die Ganzzahlen $n > 0$ und $D > 0$, durch Leerzeichen getrennt. Die zweite Zeile enthält n Ganzzahlen d_1 bis d_n, durch Leerzeichen getrennt. Die dritte Zeile enthält n Ganzzahlen e_1 bis e_n, durch Leerzeichen getrennt.

Ausgabe Für jeden Test soll in einer separaten Zeile eine Ganzzahl, nämlich die maximale gesamte Energieausbeute, ausgegeben werden.

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_P8.1`. Das Passwort für die Einschreibung ist "`quicksort`".

Beispiele

Eingabe (wie im obigen Beispiel und ein weiterer Test)

```
2
5 5
0 4 6 10 13
6 9 5 15 11
5 1
1 2 3 5 6
3 2 4 5 1
```

Ausgabe (mit den Windrädern 2 und 4 im ersten Test und allen Windrädern im zweiten Test)

```
24
15
```

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.

Beachten Sie bezüglich der Komplexität, dass D und die Werte d_i und e_i selbst für kleine n sehr gross werden können. Wir garantieren lediglich, dass die Summe aller e_i in einem Test kleiner als der maximale Wert des Java-Datentyps `int` ist.