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

10th November 2016

Datenstrukturen & Algorithmen Exercise Sheet P8 AS 16

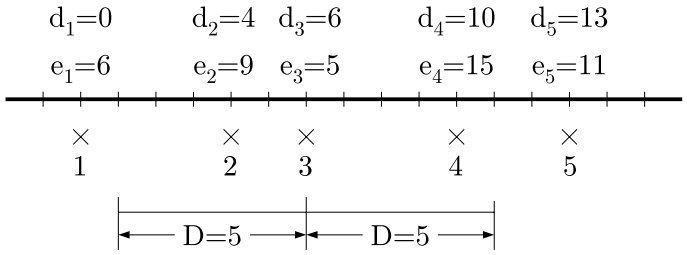
Hand-in: Before Thursday, 17th November 2016 10:00 via the online judge (source code only).

Exercise P8.1 *Wind Turbines.*

We want to place wind turbines along one road out of our town to produce energy. Due to the local conditions, n different positions are possible, some better suited than others. The laws prescribe that the distance between two wind turbines has to be at least D meters.

The n possible positions are given as the distances $d_1, \ldots, d_n$ in meters from the start of the road, e.g. as coordinates on a line, such that $0 \leq d_i < d_{i+1}$ for all $i \in \{1, \ldots, n-1\}$. When a wind turbine is installed at location i it produces energy e_i, where all the values of $e_1, \ldots, e_n$ are also given.

The task is to determine where to build the wind turbines to maximizes the total energy yield (the sum of energy produced). The number of turbines to build is limited only by the number n and the minimal distance constraint.



Example The image above shows a situation for $n = 5$ possible positions. For example, if a wind turbine is installed at position 3, no wind turbines can be installed at the positions 2 or 4. When the wind turbines are placed on the positions 1, 3 and 5, they produce $6 + 5 + 11 = 22$ units of energy. This solution is not optimal: An installation of wind turbines on the positions 2 and 4 produces $9 + 15 = 24$ units of energy.

Multiple cases As will be usual from now on, your program has to solve several (usually independent) *cases* in one *test run*[1] to get the points for that run – see input description and example below. The template will help you with loading the data.

Note that to get the points for a test run, your program has to match the expected output exactly, that is solve all the C cases correctly. You can generally expect some of the grouped cases to test your program on small corner-cases and special (but valid) inputs. For example a

[1]The *test runs* are called `example`, `judge1`, `judge2`, ... The *cases* are the sub-tasks in these tests.

program that would solve a random large input but fail for some specific small input will not get any points for that test run.

Input The input consists of several cases. The first line contains the integers $C > 0$, the number of cases to follow.

Each case is independent of the others and consists of three lines: The first line contains the integers $n > 0$ and $D > 0$ separated by a space. The second line contains n integers d_1 to d_n separated by spaces. The third line contains n integers e_1 to e_n separated by spaces.

Output For every case, the output should contain one integer number on a separate line – the maximal total energy yield of that case.

Grading You will get 1 bonus point for every 100 judge points, rounded down. Your may get up to 200 judge points. The program should be reasonably efficient and work in $O(n)$ time or similar to get full points.

Submit your `Main.java` at `https://judge.inf.ethz.ch/team/websubmit.php?cid=18985&problem=DA_P8.1`, enroll password is "`quicksort`".

Examples

Input (as in the figure above plus one more case)

```
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
```

Output (using turbines 2 and 4 in first, all in the second case)

```
24
15
```

Notes For this exercise, we provide a program template as an Eclipse project archive on the lecture website, which will load the input for you. The archive also contains more test data for you convenience – you can copy&paste the data into your running program.

When thinking about complexity, note that D and the values d_i and e_i can be very large even for small n. Technically, we only guarantee that the sum of all e_i in one case fits within a Java `int`.