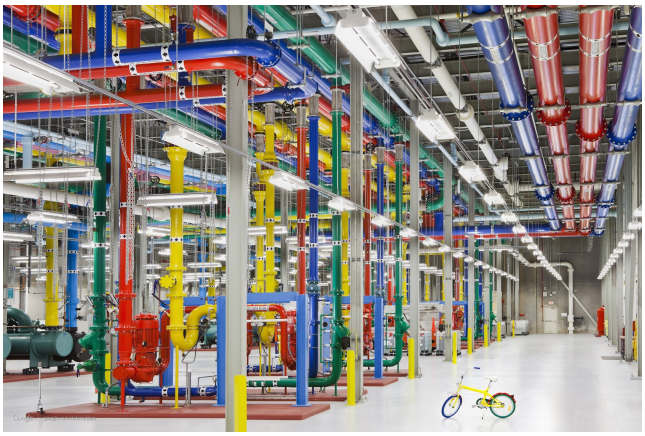
Hash Code

by Google France

EN

Problem statement for the Qualification Round, March 12th, 2015

Optimize a Data Center



Introduction

For over ten years, Google has been building data centers of its own design, deploying thousands of machines in locations around the globe. In each of these locations, batteries of servers are at work around the clock, running services we use every day, from Google Search and YouTube to the judge system of Hash Code.

Data center design is a multi-factor optimization problem. Surely, we want to get as much computing capacity as possible — the services need a lot of resources today and will be asking for more tomorrow. But maximizing the raw capacity is not the only goal: it's equally important to ensure that the computing capacity is provided reliably even in the face of inevitable hardware failures.

Task

Servers in a data center are **physically divided into rows**. Rows can share resources such as electric power. If such a shared resource fails, we assume that the entire row is lost and all servers in that row become **unavailable**.

Servers in a data center are also **logically divided into pools.** Each server belongs to exactly one pool, and provides it with some amount of computing resources, called **capacity**. The capacity of a pool is the sum of the capacities of the **available** servers in that pool.

To ensure reliability of a pool, it is therefore desirable to distribute its servers between different rows. Then, when a row fails, the pool can continue to operate on the servers from the remaining rows (albeit

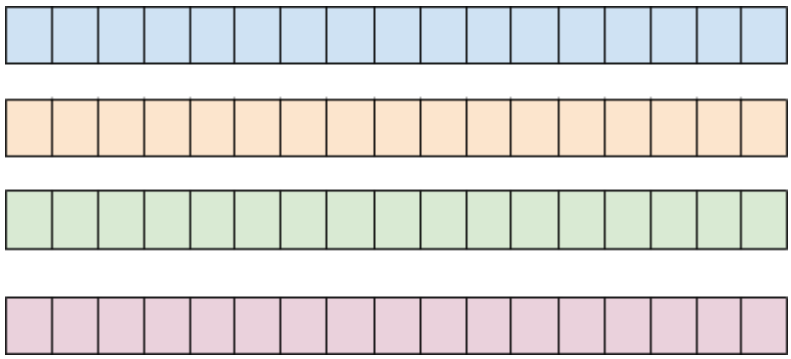
with a reduced capacity because of the unavailable servers). The **guaranteed capacity** of a pool is the minimum capacity it will have when at most one data center row goes down.

Given a schema of a data center and a list of available servers, your goal is to assign servers to **slots within the rows** and to **logical pools** so that the **lowest guaranteed capacity** of all pools is maximized.

Problem description

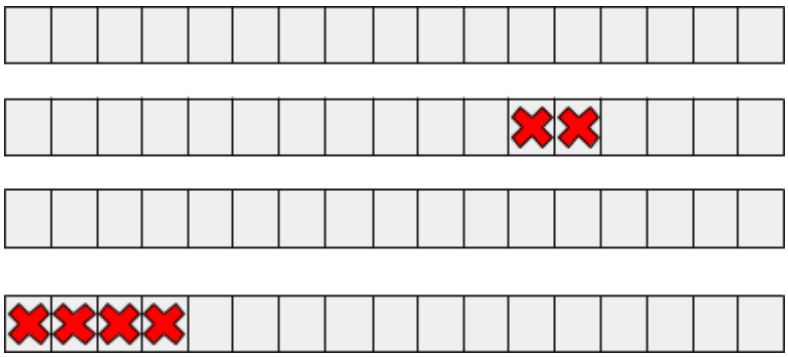
Slots

A data center is modeled as **rows** of **slots** in which servers can be placed.



A data center with four rows highlighted in different colors.

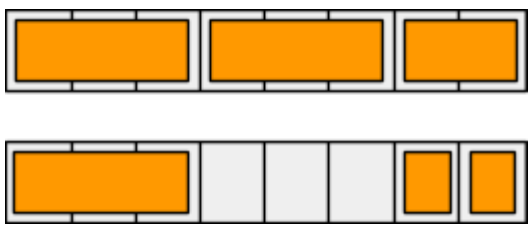
Some of the slots might be unavailable (e.g. because of other installations occupying some of the slots).



Six unavailable slots of the data center marked with red crosses.

Servers

Each server is characterized by its **size** and **capacity**. Size is the number of consecutive slots occupied by the machine. Capacity is the total amount of CPU resources of the machine (an integer value).



Servers of different sizes placed in two rows.

Input data

The input data is provided in a plain text file containing exclusively ASCII characters with lines terminated with a single '\n' character at the end of each line (UNIX-style line endings).

The file consists of:

- one line containing the following five natural numbers separated by single spaces:
  - ***R*** $(1 \leq R \leq 1000)$ denotes the number of rows in the data center,
  - ***S*** $(1 \leq S \leq 1000)$ denotes the number of slots in each row of the data center,
  - ***U*** $(0 \leq U \leq R \times S)$ denotes the number of unavailable slots,
  - ***P*** $(1 \leq P \leq 1000)$ denotes the number of pools to be created,
  - ***M*** $(1 \leq M \leq R \times S)$ denotes the number of servers to be allocated;
- ***U*** subsequent lines describing the unavailable slots of the data center. Each of these lines contains two natural numbers separated by a single space: r_i and s_i $(0 \leq r_i < R, 0 \leq s_i < S)$, denoting the row number (r_i) and the slot number within the row (s_i) of the particular unavailable slot;
- ***M*** subsequent lines describing the servers to be allocated. Each of these lines contains two natural numbers separated by a single space: z_i and c_i $(1 \leq z_i \leq S, 1 \leq c_i \leq 1000)$, denoting the size of the server (z_i), ie. the number of slots that it occupies, and the capacity (c_i) of the machine.

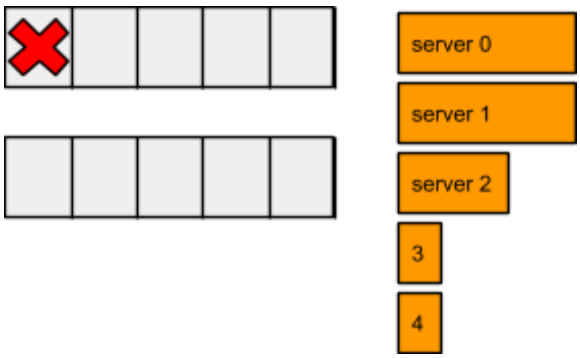
Example

An example input file could look as follows.

`2 5 1 2 5`	`2 rows of 5 slots each, 1 slot unavailable, 2 pools and 5 servers.`
`0 0`	`Coordinates of the first and only unavailable slot.`
`3 10`	`First server takes three slots and has a capacity of 10.`
`3 10`	`So does the second one.`
`2 5`	`The third one takes two slots and has a capacity of 5.`
`1 5`	`The fourth one takes just one slot and has a capacity of 5.`
`1 1`	`The fifth one takes just one slot too and has a capacity of 1.`

Example input file.

The file above describes a data center of two rows of five slots each and servers of different parameters.



Data center and servers described in the above example.

Submissions

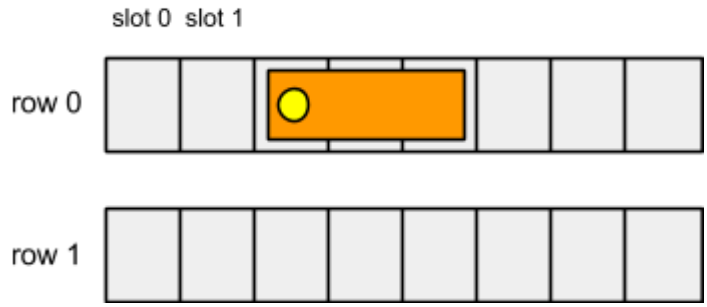
File format

A submission file has to be a plain text file containing exclusively ASCII characters with lines terminated with either a single '\n' character at the end of each line (UNIX-style line endings) or '\r\n' characters at the end of each line (Windows-style line endings).

The file has to consists of ***M*** lines describing the allocation of the individual servers, in the same order as they appeared in the input file. Each of these lines has to contain either:

- three natural numbers separated by single spaces: ar_i, as_i, ap_i $(0 \leq ar_i < R, 0 \leq as_i < S, 0 \leq ap_i < P)$ denoting the allocated row (ar_i) and slot within the row (as_i) for the server, and the allocated logical pool for it (ap_i);
- the single lowercase "x" letter, if the server is left unallocated.

For servers that occupy more than one slot, the slot of the lowest index (leftmost in the picture below) should be indicated as the position of the server.



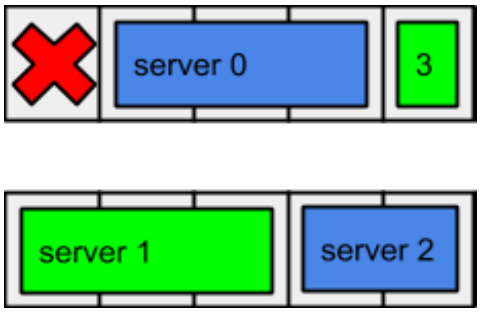
The position of the three-slot server in the picture should be described as $ar_i = 0$ and $as_i = 2$.

Example

The following example submission file corresponds to the example input file presented above.

`0 1 0`
`1 0 1`
`1 3 0`
`0 4 1`
`x`
	`Server 0 placed in row 0 at slot 1 and assigned to pool 0.`
	`Server 1 placed in row 1 at slot 0 and assigned to pool 1.`
	`Server 2 placed in row 1 at slot 3 and assigned to pool 0.`
	`Server 3 placed in row 0 at slot 4 and assigned to pool 1.`
	`Server 4 not allocated.`

Example submission file.



Server layout corresponding to the given example.

Validation

For the solution to be accepted, it has to meet the following criteria:

- the format of the file has to match the description above,
- each slot of the data center has to be occupied by at most one server,
- no server occupies any unavailable slot of the data center,
- no server extends beyond the slots of the row.

Score

For each pool, its **guaranteed capacity** is defined as the lowest total capacity of its running servers when exactly one of the data center rows is unavailable (over all possible rows). The score awarded to the submission is the **lowest** guaranteed capacity of all pools. The goal is to maximize this score.

Example

In the example submission given above, the total score is **5**, as the guaranteed capacities of both pools is 5 and $min(5, 5) = 5$. See the figure for details.

	row 0	row 1	guaranteed capacity
pool 0	10	5	**5**
pool 1	5	10	**5**

Guaranteed capacities of each pool in the example submission.

Formally speaking

For ***i*** ($0 \leq i < P$), the **guaranteed capacity *gc*$_i$** can be defined as:

$$gc_i = min_{0 \leq r < R}\left(\sum_{k=0,\ server\ k\ in\ pool\ i}^{M-1} c_i - \sum_{k=0,\ server\ k\ in\ pool\ i,\ server\ k\ in\ row\ r}^{M-1} c_i\right)$$

and the total score can be defined as:

$$score = min_{0 \leq i < P} gc_i$$