Delivery

Problem statement for Online Qualification Round, Hash Code 2016

Introduction

The Internet has profoundly changed the way we buy things, but the online shopping of today is likely not the end of that change; after each purchase we still need to wait multiple days for physical goods to be carried to our doorstep.

This is where drones come in - autonomous, electric vehicles delivering online purchases. Flying, so never stuck in traffic. As drone technology improves every year, there remains a major issue: how do we manage and coordinate all those drones?

Task

Given a fleet of drones, a list of customer orders and availability of the individual products in warehouses, schedule the drone operations so that the orders are completed as soon as possible.

Problem description

Map

The simulation takes place on a two-dimensional grid. The grid is **not** cyclic and a drone cannot fly outside of the grid. The drones can fly over all cells within the grid.

Each cell is identified by a pair of integer coordinates [***r***, ***c***] ($0 \leq r < row\ count$, $0 \leq c < column\ count$).

Products

There are a number of product types available for order. Each product type has one or more product items available in warehouses. Each product type has a fixed product weight, identical for all product items. Every product weight is guaranteed to be smaller or equal to the maximum payload that a drone can carry.

Warehouses

Product items are stored in several warehouses. Each warehouse is located in one particular cell of the grid, different for each warehouse. Each warehouse initially stocks a known number of product items of each product type. No new product items beyond the initial availability will be stocked in the warehouses during the simulation, but the drones can transport product items between the warehouses. Any warehouse does not necessarily need to have every product type available.

Orders

Each order specifies the product items purchased by the customer. The product items in an order can be of one or multiple product types and can contain multiple product items of the same product type.

Each order specifies the cell in the grid where the product items have to be delivered. It is possible to have multiple orders with the same delivery cell. No order has the delivery cell that is a location of a warehouse.

The order is considered fulfilled when all of the ordered product items are delivered. Individual product items can be delivered in multiple steps, in any order. It is valid to deliver the individual product items of an order using multiple drones, including using different drones at the very same time.

It is guaranteed that for each product type, the total number of product items in all orders is **not greater** than the total availability of product items of this product type in all warehouses.

It is **not** required to deliver all orders (see the Scoring section at the end).

Drones

Drones transport product items from warehouses to customers and between the warehouses.

The drones always use the shortest path to fly from one cell in the grid to another. The distance from cell $[r_a, c_a]$ to cell $[r_b, c_b]$ is calculated as $\sqrt{|r_a - r_b|^2 + |c_a - c_b|^2}$ (two-dimensional Euclidean distance). We define a drone flight as a single movement of the drone that happens between two subsequent commands for the given drone (as described below). Each drone flight takes **one turn** per one unit of distance between the start and the destination cell, **rounded up** to the next integer.

Multiple drones can be in the same cell at any moment - they never collide, as they can fly at different altitudes.

For **example**, if the distance to be flown is 2.9 distance units, the duration of the flight is 3 turns. If the distance to be flown is exactly 4 distance units, the duration of the flight is 4 turns.

At the beginning of the simulation, all drones are at the first warehouse (warehouse with id 0).

Commands

Each drone can be given the following basic commands:

- **Load**: Moves the specified number of items of the specified product type from a warehouse to the drone's inventory. If the drone isn't at the warehouse it will fly there using the shortest path before loading the product items. The requested number of items of the specified product type must be available in the warehouse. The total weight of the items in the drone's inventory after the load cannot be bigger than the drone's maximum load.
- **Deliver**: Delivers the specified number of items of the specified product type to a customer. If the drone isn't at the destination it will fly there using the shortest path before delivering the product items. The drone must have the requested number of items of the specified product type in its inventory.

Each drone can also be given the following advanced commands. These commands are **not** necessary to solve the problem, but you can use them to further improve your solution.

- **Unload**: Moves the specified number of items of the specified product type from drone's inventory to a warehouse. If the drone isn't at the warehouse it will fly there using the shortest path before unloading the product items. The drone must have the requested number of items of the specified product type in its inventory.
- **Wait**: Waits the specified number of turns in the drone's current location.

Simulation

The simulation proceeds in ***T*** turns, from 0 to $T-1$. A drone executes the commands issued to it in the order in which they are specified, one by one. The first command issued to the drone starts in turn 0.

The duration of a command depends on its type:

- Each **Load**/**Deliver**/**Unload** command takes $d+1$ turns, where d is the distance travelled by the drone to perform the requested action (d can be 0 if the drone is already in the required location). When the command starts, the drone flies to the required location in d turns. Then, the *actual action* (loading, delivery or unloading) takes place and takes 1 turn.
- Each **Wait** command takes w turns, where w is the specified number of turns.

For **example**, let's assume that at the beginning of the simulation (at the beginning of turn 0) a drone is in warehouse 0 at the cell [1,1], warehouse 1 is in cell [1, 3], customer order 0 has to be delivered to [1,4], and p and q are some product types, the items of which are part of order 0.

Let's consider the following commands:

- **Load one item of product type p at warehouse 0.** This will take 1 turn: 0 turns to get to the warehouse and 1 turn to load the item.
- **Load five items of product type q at warehouse 1.** This will take 3 turns: 2 turns to get from [1, 1] to the warehouse at [1,3] and 1 turn to load the items.
- **Deliver one item of product type p for customer order 0**. This will take 2 turns: 1 turn to get from [1, 3] to the delivery cell at [1, 4] and 1 turn to deliver the item.
- **Deliver five items of product type q for customer order 0**. This will take 1 turn: 0 turns to get to the delivery cell and 1 turn to deliver the items.

If there are multiple *actual actions* (see above) taking place in the same turn in the same warehouse, all unloading commands are processed **before** all loading commands.

For **example**, let's assume that in one turn, three different drones are all located at warehouse 0 and ordered to perform the following actions:

- Load one item of product type p at warehouse 0. (first drone)
- Load one item of product type p at warehouse 0. (second drone)
- Unload two items of type p at warehouse 0. (third drone)

Because unloading takes precedence over loading, all commands will succeed even if there were no items of product type p at the warehouse before this turn.

Input data set

The input data is provided as a data set file - 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).

Product types, warehouses and orders are referenced by integer IDs. There are ***P*** product types numbered from 0 to $P-1$, ***W*** warehouses numbered from 0 to $W-1$ and ***C*** orders numbered from 0 to $C-1$.

File format

The first section of the file describes the **parameters of the simulation**. This section contains a single line containing the following natural numbers separated by single spaces:

- number of rows in the area of the simulation $(1 \leq number\ of\ rows \leq 10000)$
- number of columns in the area of the simulation $(1 \leq number\ of\ columns \leq 10000)$
- ***D*** - number of drones available $(1 \leq D \leq 1000)$
- deadline of the simulation $(1 \leq deadline\ of\ the\ simulation \leq 1000000)$
- maximum load of a drone $(1 \leq maximum\ load\ of\ a\ drone \leq 10000)$

The next section of the file describes the **weights of the products available for orders**. This section contains:

- a line containing the following natural number:
  - ***P*** - the number of different product types available in warehouses $(1 \leq P \leq 10000)$
- a line containing ***P*** natural numbers separated by single spaces denoting weights of subsequent products types, from product type 0 to product type ***P*** - 1. For each weight, $1 \leq weight \leq maximum\ load\ of\ a\ drone$.

The next section of the file describes the **warehouses and availability of individual product types** at each warehouse. This section contains:

- a line containing the following natural number:
  - ***W*** - the number of warehouses $(1 \leq W \leq 10000)$
- two lines for each warehouse, each two lines describing the subsequent warehouses from warehouse 0 to warehouse $W - 1$:
  - a line containing two natural numbers separated by a single space: the row and the column in which the warehouse is located
    $(0 \leq row < number\ of\ rows;\ 0 \leq column < number\ of\ columns)$
  - a line containing **P** natural numbers separated by single spaces: number of items of the subsequent product types available at the warehouse, from product type 0 to product type $P - 1$. For each product type, $0 \leq number\ of\ items \leq 10000$ holds.

The next section of the file describes the **customer orders**. This section contains:

- a line containing the following natural number:
  - ***C*** - the number of customer orders $(1 \leq C \leq 10000)$
- three lines for each order, each three lines describing the subsequent orders from order 0 to ***C*** - 1:
  - a line containing two natural numbers separated by a single space: the row of the delivery cell and the column of the delivery cell
    $(0 \leq row < number\ of\ rows;\ 0 \leq column < number\ of\ columns)$
  - a line containing one natural number $\mathbf{L_i}$ - the number of the ordered product items $(1 \leq L_i < 10000)$
  - a line containing $\mathbf{L_i}$ integers separated by single spaces: the product types of the individual product items. For each of the product types, $0 \leq product\ type < P$ holds.

Example

In the comments, “u” is an abbreviation for units of weight.

`100 100 3 50 500`
`3`
`100 5 450`
`2`
`0 0`
`5 1 0`
`5 5`
`0 10 2`
`3`
`1 1`
`2`
`2 0`
`3 3`
`3`
`0 0 0`
`5 6`
`1`
`2`
	`100 rows, 100 columns, 3 drones, 50 turns, max payload is 500u`
	`There are 3 different product types.`
	`The product types weigh: 100u, 5u, 450u.`
	`There are 2 warehouses.`
	`First warehouse is located at [0, 0].`
	`It stores 5 items of product 0 and 1 of product 1.`
	`Second warehouse is located at [5, 5].`
	`It stores 10 items of product 1 and 2 items of product 2.`
	`There are 3 orders.`
	`First order to be delivered to [1, 1].`
	`First order contains 2 items.`
	`Items of product types: 2, 0.`
	`Second order to be delivered to [3, 3].`
	`Second order contains 3 items.`
	`Items of product types: 0, 0, 0.`
	`Third order to be delivered to [5, 6]`
	`Third order contains 1 item.`
	`Items of product types: 2.`

Example input file.

Submissions

File format

The first line of the output file contains a single integer **Q**, the number of drone commands ($0 \leq Q \leq D \times T$), where **D** is the number of drones and **T** is the duration of the simulation.

The rest of the output file should describe the individual commands to the drones, each command in a separate line. Commands for different drones can be intertwined but commands for any given drone must come in chronological order.

Drones are referenced by integer IDs, from 0 to $D - 1$, where **D** is the number of drones given in the input file.

Each **Load** or **Unload** command is described by a line with the following elements separated by single spaces:

- the ID of the drone that the command is for
- the command tag - a single character, either ‘L’ (for load) or ‘U’ (for unload),
- the ID of the warehouse from which we load items / to which we unload items
- the ID of the product type
- the number of items of the product type to be loaded or unloaded - a **positive** integer

Example command: `0 L 1 2 3` (Command to drone 0, load at warehouse 1 products of product type 2, three of them.

Each **Deliver** command is described by a line with the following elements separated by single spaces:

- the ID of the drone that the command is for
- the command tag - single character ‘D’
- the ID of the customer order we are delivering items for
- the ID of the product type
- the number of items of the product type to be delivered - a **positive** integer

Example command: `0 D 1 2 3` (Command to drone 0, deliver for order 1 items of product type 2, three of them.

Each **Wait** command is described by a line with the following elements separated by single spaces:

- the ID of the drone that the command is for
- the command tag - single character ‘W’
- the number of turns for which the drone needs to wait - a **positive** integer

Example command: `0 W 3` (Command to drone 0, wait for three turns)

Example

9 0 L 0 0 1 0 L 0 1 1 0 D 0 0 1 0 L 1 2 1 0 D 0 2 1 1 L 1 2 1 1 D 2 2 1 1 L 0 0 1 1 D 1 0 1	9 commands in total. Drone 0: **load** one product 0 at warehouse 0. Drone 0: **load** one product 1 at warehouse 0. Drone 0: fly to customer 0 and **deliver** one product 0. Drone 0: fly to warehouse 1 and **load** one product 2. Drone 0: fly to customer 0 and **deliver** one product 2. Drone 1: fly to warehouse 1 and **load** one product 2. Drone 1: fly to customer 2 and **deliver** one product 2. Drone 1: fly to warehouse 0 and **load** one product 0. Drone 1: fly to customer 1 and **deliver** one product 0.

Example submission file.

Validation

The output file is considered valid if it meets the following criteria:

- The format of the output file matches the description above
- All commands are valid with regard to the requirements in the Commands section
- No order receives more product items of any type than the number of product items of this type that is specified in the order.
- All commands for any given drone take at most ***T*** turns in total, where ***T*** is the number of turns of the simulation.

Scoring

Each completed order will earn between 1 and 100 points, depending on the turn in which it is completed. The order is completed in the first turn at the end of which all items in the order are delivered.

For an order completed in turn ***t*** and a simulation taking ***T*** turns in total, the score for the order is calculated as $(T - t) / T \times 100$, rounded up to the next integer.

For **example**, if the simulation takes 160 turns (***T*** = 160), and an order consists of three items, delivered at turns 5, 15 and 15, then the order is considered completed at t = 15, and the score is calculated as $(160 - 15)/160 = 0.90625$, multiplied by 100 and rounded up to **91 points**.

For an order completed in the last turn of the simulation (***t*** = 159, ***T*** = 160), the score would be $(160 - 159)/160 = 0.00625$, multiplied by 100 and rounded up to **1 point**.

The score for a single data set will be the sum of the scores of all completed orders.

Note that there are multiple data sets representing separate instances of the problem. The final score for your team will be the sum of your best scores on the individual data sets.

Scoring example

Score for the example solution given above is calculated as follows.

Drone 0:

- **loads** item 0 from warehouse 0 **in turn 0**
- **loads** item 1 from warehouse 0 **in turn 1**
- **flies** to order 0 **in turns 2 and 3**
- **delivers** item 0 to order 0 **in turn 4**
- **flies** to warehouse 1 **in turns 5 to 10**
- **loads** item 2 from warehouse 1 **in turn 11**
- **flies** to order 0 **in turns 12 to 17**
- **delivers** item 2 to order 0 **in turn 18**. Order 0 is now fulfilled, **scoring** $\frac{(50 - 18)}{50} \cdot 100 = 64$ **points**

Drone 1:

- **flies** to warehouse 1 **in turns 0 to 7**
- **loads** item 2 from warehouse 1 **in turn 8**
- **flies** to order 2 **in turn 9**
- **delivers** item 2 to order 2 **in turn 10**. Order 2 is now fulfilled, **scoring** $\frac{(50 - 10)}{50} \cdot 100 = 80$ **points**
- **flies** to warehouse 0 **in turns 11 to 18**
- **loads** item 0 from warehouse 0 **in turn 19**
- **flies** to order 1 **in turns 20 to 24**
- **delivers** item 0 to order 1 **in turn 25**. Order 1 is now fulfilled, **scoring** $\frac{(50 - 25)}{50} \cdot 100 = 50$ **points**

Drone 2 isn't used at all and there are no other orders.

The score of the example submission is $64 + 50 + 80 = 194$ points.