
DRONE KEEPER
UNIVERSITY CUP
ENTELECT
CHALLENGE
2025

Introduction

Welcome to Megazoo Inc. One of the world's leading zoo owners, animal rehabilitation experts, and proud record holder of “Biggest Zoo on the Planet”. As you might imagine, when working with zoos that have up to 15 000 enclosures (yes, that’s right), it can be quite tricky to feed all the animals with the limited number of zookeepers.

Your IT department was carefully selected to create a program that feeds the animals with a pre-programmed drone. Fully automated, efficient, and convenient. This would allow the zookeepers to focus on more important tasks within the zoo.

For this, you will need to create a program which generates the optimal paths that a drone might follow to feed the animals without running out of battery.

You will be awarded points based on which animals you feed. The zoo has evaluated the importance of feeding each animal, and this importance factor counts as a points multiplier.

Good luck!

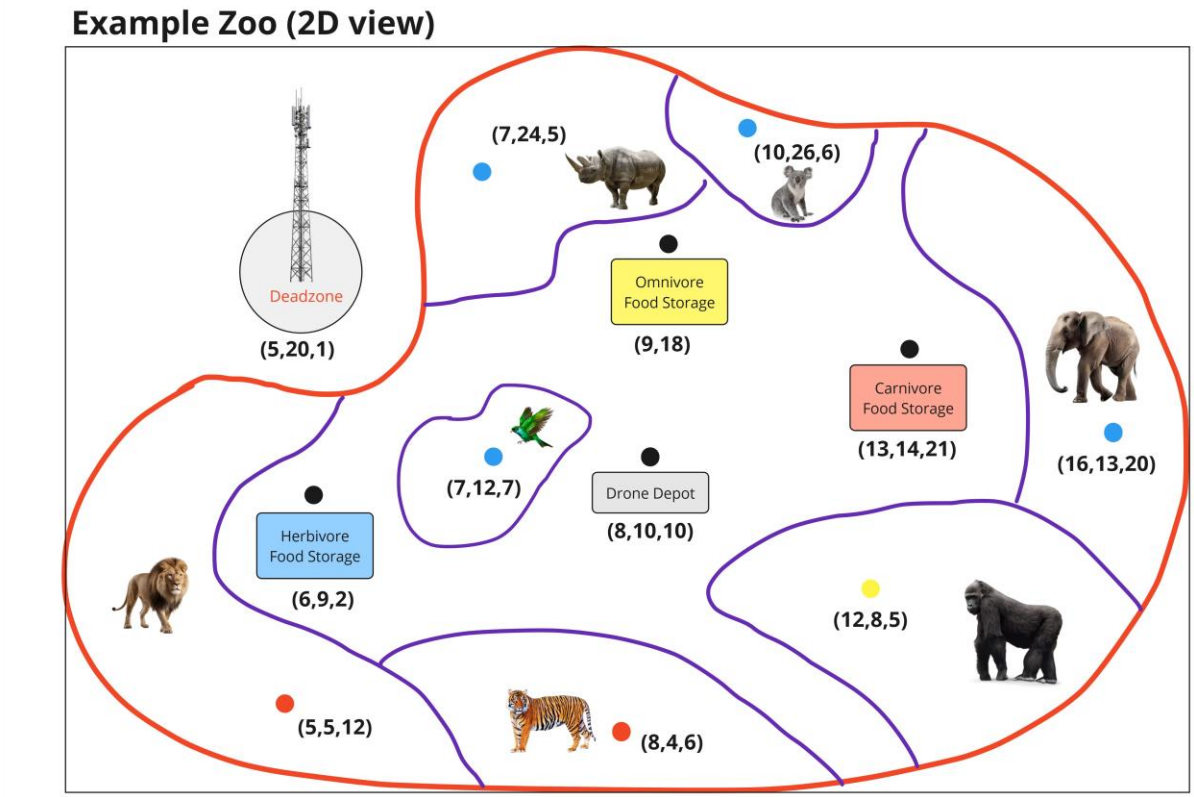
Goal

Your goal is to write a program that finds optimal paths for feeding the zoo animals that will result in most points that you can gather within the given number of drone battery swaps. Your submission must be a list of coordinates that the drone visits, per battery swop.

The Zoo

Each zoo is 3D, meaning that it has x, y, and z coordinates. The coordinate space is all integers. A zoo consists of a **Drone Depot**, **Food Storages** for the 3 respective diets (Carnivore, Herbivore, Omnivore), the **Animal Enclosures** themselves, and **Deadzones**.

For the enclosures, only the feeding point within the enclosure is provided as a coordinate. The exact dimensions of the enclosures are not relevant.



This is a 2D representation of the Example Zoo. As you can see, the Drone Depot is near the center of the zoo. This means that the starting and ending point of all drone runs will be (8,10,10).

Zoo Dimensions

The zoo is based inside of a rectangular prism. From a 2D perspective, the zoo levels are all squares that increase in size, however, the height maximum or ceiling of the zoo remains at 50m for elevation. This means a zoo can be **200m x 200m x 50m** or even **5000m x 5000m x 50m**.

Zoo Text File Representation

Provided for each level is a text file containing the zoo specification. The first step of your program would be to read this text file into your program's memory, using the appropriate data structures. Each attribute is provided in one line as follows:

```
ZOO DIMENSIONS (x,y,z)
DRONE DEPOT (x,y,z)
BATTERY DISTANCE CAPACITY (m)
FOOD STORAGES COORDINATES [(x,y,z,diet),...]
ENCLOSURES [(x,y,z,importance,diet),...]
DEADZONES [(x,y,r),...]
```

Diet

Diet is denoted by a single character: ***c*** for carnivores, ***h*** for herbivores, and ***o*** for omnivores.

Importance

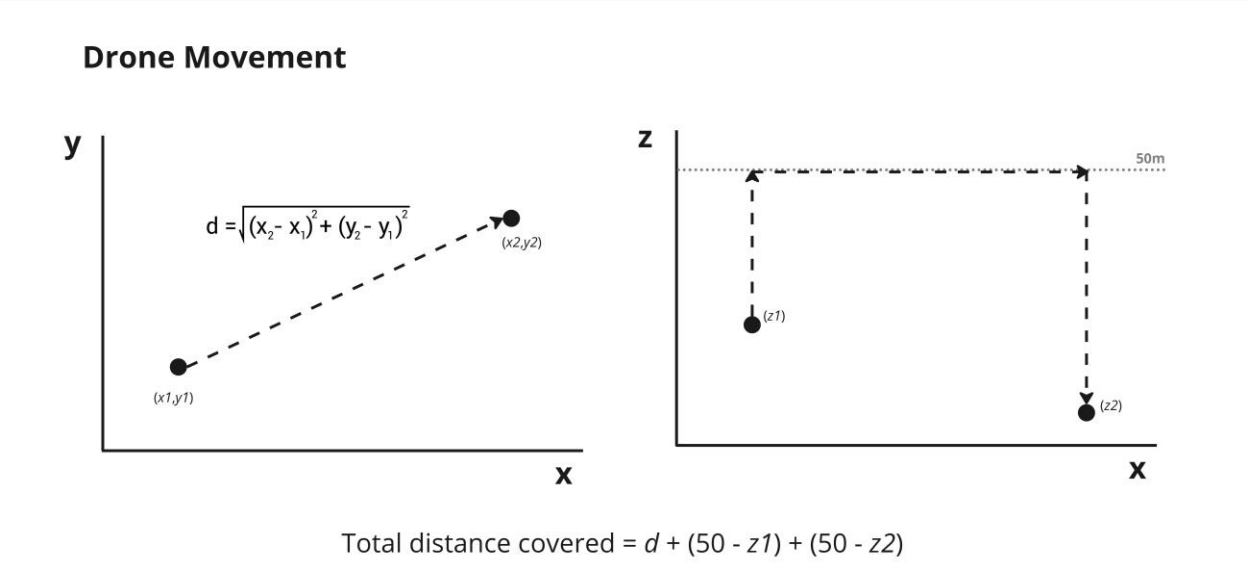
Importance is a multiplier which indicates the urgency with which some animals need to be fed. In theory, some animals can survive a few days without being fed, but if we do not feed the lions and tigers, they might feel more inclined to escape. Importance is denoted as a float value rounded to two decimal places.

Drone Mechanics

Drone mechanics are straightforward. A drone must lift off from the Drone Depot, collect food from a Food Storage, deliver the food to enclosures, and then return to the Drone Depot for a battery swap. The cycle can then be repeated for however many battery swaps are available for that level.

Movement

The drone must take off from the Drone Depot and reach drone flight height (z = 50 meters), before it can move on the horizontal plane. Diagonal movement is allowed when the drone is at flight height, but only vertical movement is allowed for takeoff and landing.



A drone is allowed to move to any (x, y) coordinate within the zoo bounds, even if it is not a Drone Depot, Food Storage, or Enclosure.

Food Storages

When a drone visits a Food Storage, it has unlimited food for that food type on board. However, only one food type can be carried at a time. When the next coordinate of a drone is that of a Food Storage, if the drone carries no- or a different food type than that of the Food Storage, it will **always** pick up the new food type. If the drone carries the same food type as the Food Storage, it will not land.

Enclosures

When a drone visits an enclosure, and the food type it is carrying is the same as that of the animal in the enclosure's diet, it will land and deliver the food. If the drone visits an enclosure but the food types do not match, the drone will not land. An enclosure can only be fed once. Thus, if an enclosure has already been fed, the drone will not land if its next point is that enclosure.

Battery Capacity

For each level, a drone is allocated batteries which can power the drone for a certain distance. If a drone battery allows for 250 meters of movement, and the proposed path is 251 meters long, the drone will cut out and fall from the sky. This will result in all points gathered for that drone run to be rendered **zero**. This means that the drone must complete a full run and return to the Depot if any points are to be awarded.

Deadzones

If a drone flies into a Deadzone, it will also cut out, and you will lose all points collected during that run. Please see the following article on how the math can be done: https://www.superprof.co.uk/resources/academic/maths/analytical-geometry/conics/circle-line-intersection.html

Levels

Each level is unique, and levels are increasing in size. Please see the following table for the parameters of each:

	Level 1	**Level 2**	**Level 3**	**Level 4**
Size (m)	100x100x50	250x250x50	700x700x50	2500x2500x50
Enclosures	20	100	1000	15000
Battery Swaps	0	10	50	250
Battery Capacity (m)	999999	1125	2750	9250
Deadzones	0	0	5	15
Food Storages	1 of each	3 of each	5 of each	10 of each

Level 1 has 999,999m battery capacity to make it easier to start writing an algorithm, since you do not have to worry about having multiple drone runs.

Submission

Your submission will be a text file containing a list of drone runs, which are lists of 2D coordinates. Your drone run must start at the Drone Depot, and end at the Drone Depot, denoted by (x_0, y_0). The *z*-axis calculations are done by the submission engine, so no need to include any *z*-axis coordinates. The submission must adhere to the following format:

$$[r_0: [(x_0, y_0), (x_1, y_1), \dots, (x_0, y_0)], \dots, r_n: [(x_0, y_0), (x_1, y_1), \dots, (x_0, y_0)]]$$

The r_n represents the drone run, but it should not be included in the submission. An example of your submission would look something like this:

```
[[(8,10),(6,9),(16,3),(8,10)],[(8,10),(13,14),(8,4),(5,5),(8,10)]]
```

Scoring

Each of your drone runs are scored individually and then added up to obtain a final score. The score is calculated by the following formula:

$$score = \sum_{i}^{n} p_i \cdot 1000 \; - \; TD$$

n Is the number of enclosures fed per run.	p_i Is the priority multiplier for the enclosure.
TD Is the total distance in meters that the drone has moved.	

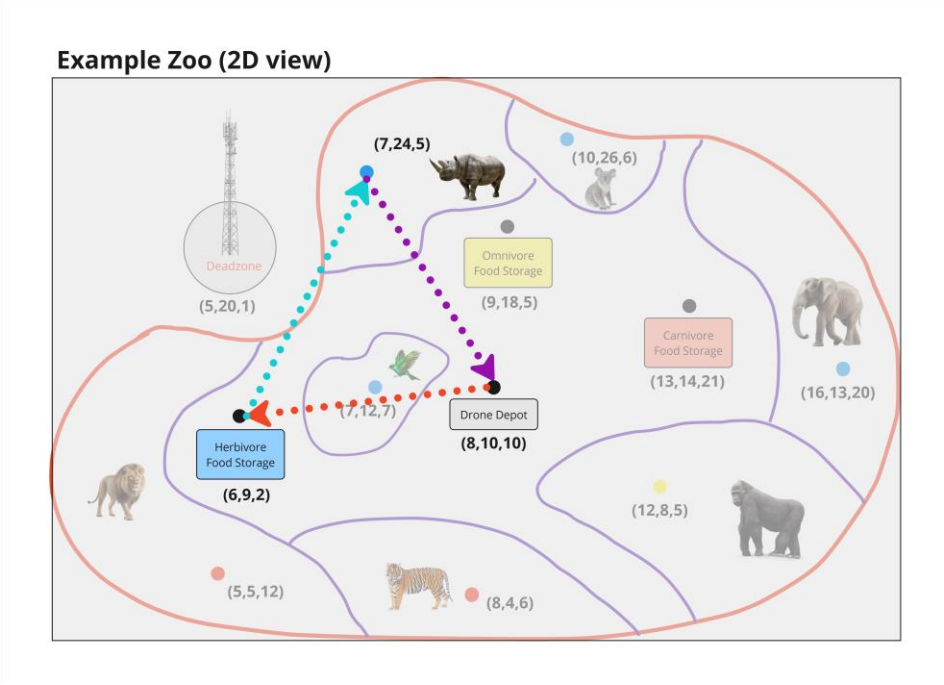
Example

For the Example Zoo, shown on page 3, the text representation would be as follows:

```
(30,30,50)
(8,10,10)
60
[(6,9,2,h),(9,18,5,o),(13,14,21,c)]
[(5,5,12,1.70,c),(8,4,6,2.50,c),(7,12,7,0.50,h),(7,24,5,5.00,h),(10,26,6,3.40,h),
(12,8,5,2.30,o),(16,13,20,4.0,h)]
[(5,20,1)]
```

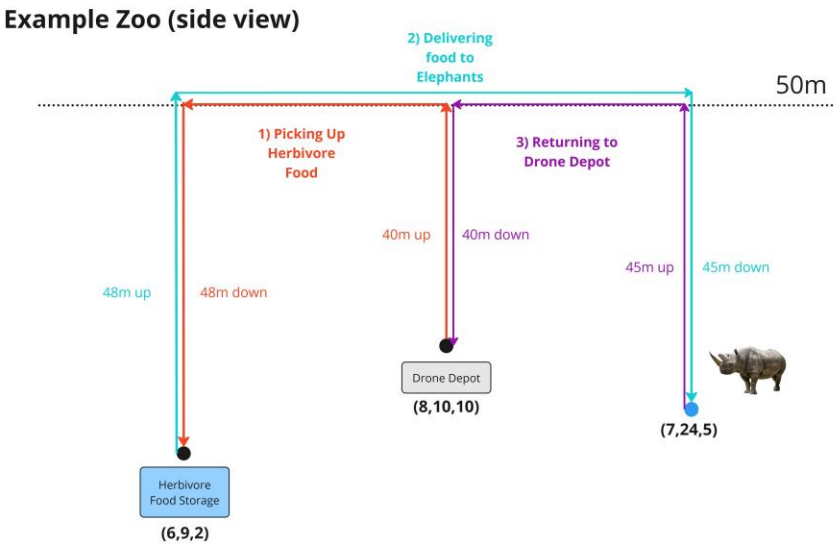
Suppose for our first drone run, we would like to feed the Elephants. First, we would need to take off from the Drone Depot, then get Herbivore food from the Food Storage, and then deliver it to the Elephant enclosure. After which we will return to the Drone Depot to complete our run.

From a 2D perspective, this is what the drone path would look like:



Our Drone Path for this run would be

```
[(8,10),(6,9),(7,24),(8,10)]
```



Now let's say that the Priority of the Rhino is 5.50. To calculate the score of this run we would need to add up the following:

For distance travelled:

```
50 - 10 = 40 (taking off from the Depot)
√((6 − 8)² + (9 − 10)²) = 2.26 (flight to herbivore food storage)
(50 - 2) * 2 = 96 (landing and taking off
√((7 − 6)² + (24 − 9)²) = 15.03 (flight to Rhino)
(50 – 5) * 2 = 90 (landing and taking off)
√((8 − 7)² + (10 − 24)²) = 14.04 (flight back to the Depot)
50 – 10 = 40 (landing at the Depot)
```

The total distance travelled is then ≈ 161m.

The score calculation is then as follows:

5.50 * 1000 – 161 = **5339 points**