

CSCI 360 – Spring 2020 – Introduction to Artificial Intelligence Project 1

Due February 12, 2020



Description

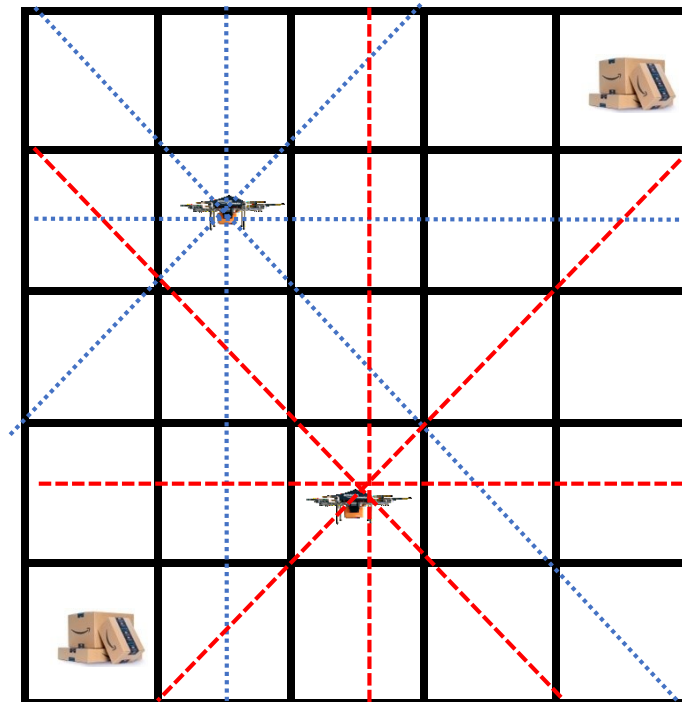
You are helping Amazon to place drone in a city to quickly deliver packages. According to the news, Amazon is developing a future delivery system to get packages delivered in less than 30 minutes via unmanned aerial vehicles (drones). To avoid unexpected collisions and interruptions from other drones, there are some rules that must be satisfied when choosing the positions (served regions) of these drones – a drone only flies within its own region. Here, we use a grid system to represent the different regions of a city.

The goal of your project is to place the drones in locations that do not conflict with each other, while maximizing the total number of packages delivered for the day. You must follow the following guidelines:

- Drones cannot be in the same square, same row, same column, or along the same diagonal. (Think of the 8 queens on a chess board)
- Packages get delivered when drones are in same square as packages. Each drone can take one or more packages.
- Once the drone's position gets assigned, it's not permitted to change.
- All drones must be allocated.
- The grid coordinate system will be indexed starting from the top-left corner. An example of a 5 by 5 grid is given below with each cell's coordinates:

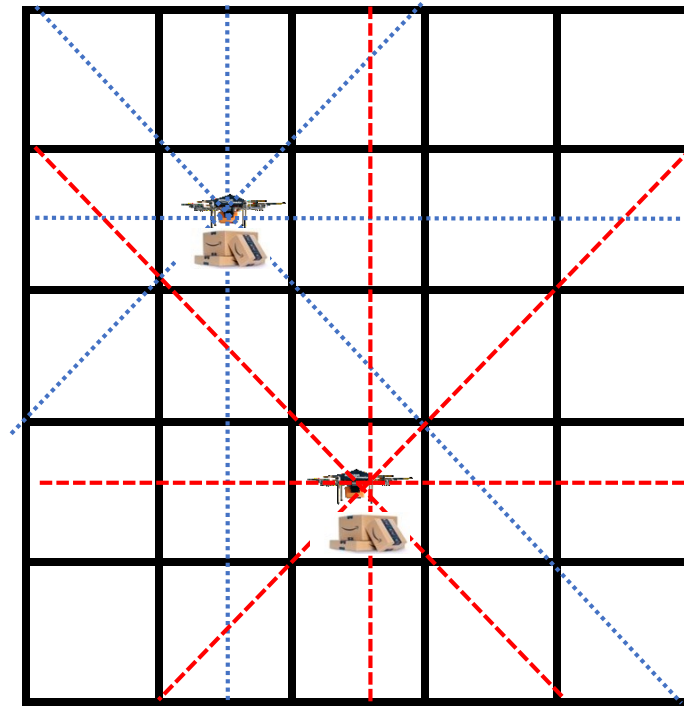
0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

Example 1



The two drones have been placed. The red and blue lines show the limitations on placing drones; no drones may be placed on the same row, column, or diagonal as another drone. Since no packages are in the same square as either of the drone, no value is gained in Example 1.

Example 2



Since both packages are in the same square as the drones, two packages get delivered in Example 2.

Guidelines

This is a programming assignment. Your program should read a text file named “input.txt” in the current file directory that contains a problem definition, and write your solution to a text file named “output.txt” to the current file directory. Since **each project is checked via an automated script**, your output should match the specified format *exactly*.

The grading script will

- Create an “input.txt” file, and delete any old “output.txt” file.
- Run your program, which should be named “project1cs360s2020.*”, where * is .cpp, .java, or .py.
- Check your “output.txt” file.

You should upload and test your program on Vocareum, and after you are done, you will submit it there. You will be provided sample inputs and outputs to check that you can correctly parse the problem definition and generate a correctly formatted output. The sample inputs are simple; although the format of grading cases will be identical, you should not assume that if your program works on the samples, it will work on all grading test cases. There will be more test cases used in grading and it is ***your task to make sure that your program will work correctly on any valid input.***

The inputs will always be valid inputs – you don't need to worry about exception handling.

Input: End-of-line character is LF (since Vocareum follows the Unix convention).

First line: strictly positive 32-bit integer n , the width/height of the $n \times n$ delivery area, $n \leq 15$

Second line: strictly positive 32-bit integer d , the number of drones, $d \leq n$

Third line: strictly positive 32-bit integer p , the number of packages, $p \leq 100$

Fourth line: algorithm to use, either *astar* for A* search or *dfs* for depth-first search

Next a lines: the list of package x,y coordinates, separated with the End-of-line character LF.

Multiple lines with the same x,y coordinates denote multiple packages with the same location.

Output:

Max number of delivered packages: strictly positive 32-bit integer m

Helpful Hints

1. **We will not give unsolvable inputs.** There won't be more drones than can be assigned without conflicts, or bad inputs that don't conform to the format described above.
2. **Think about representing the problem.**
 - a. What is a good representation of states and actions?
 - b. How will you evaluate the "score" of a state?
3. **Think about complexity.**
 - a. How does grid size affect the branching factor?
 - b. How does the number of drones affect the search tree depth?

Project Rules

1. Your program must finish each test case within **3 minutes** on Vocareum. Although upper bounds have been provided, test cases are created with this time limit in mind, and you don't need to expect the maximum size of all parameters in one test case.
2. You may use **ONLY C++, Java, Python 2.7, 3.x**, and any libraries **that are included in Vocareum**.
3. C++ and Java files will be compiled using the following commands (you may confirm this work by submitting your code for testing on the small submission test cases):
 1. `g++ -o project1cs360s2020 project1cs360s2020.cpp`
`timeout 180s ./project1cs360s2020`
 2. `javac project1cs360s2020.java`
`timeout 180s java project1cs360s2020`
4. Projects must be submitted through Vocareum. Please only upload your program code. **Don't create any subfolder or upload any other files.** Please refer to <http://help.vocareum.com/article/30-getting-started-students> to get started.
5. You are strongly encouraged to submit your program 24 hours ahead of the deadline to avoid any last-minute submission issues on Vocareum.
6. There is a grace period of 24 hours during which time your submission will be accepted, but with a 20% penalty. After 24 hours, your submission will not be accepted.

Academic Honesty and Integrity

All project material is checked vigorously for dishonesty. Violations of academic honesty are forwarded to the Office of Student Judicial Affairs. To be safe, you are urged to err on the side of caution. Do not copy work from another source. Sanctions for dishonesty will be reflected in *your permanent record* and will negatively impact your future success. As a general guide:

- **Do not copy** code or written material from another student.
Do not collaborate on this project. The project is to be solved individually.
Do not copy code off the web. This is easier to detect than you may think.
- **Do not share** any test cases that you may create to check your program's behavior in more complex scenarios with other students.
Do not copy code from past students.
- **Do** ask the CPs/TAs/Instructor in case you are unsure about if certain actions constitute dishonesty. It is better to be safe than sorry.