CSCI 360 – Spring 2019 – Introduction to Artificial Intelligence Project 1

Due February 8, 2019

https://www.wwf.org.uk/conservationtechnology/camera-trap.html

Description

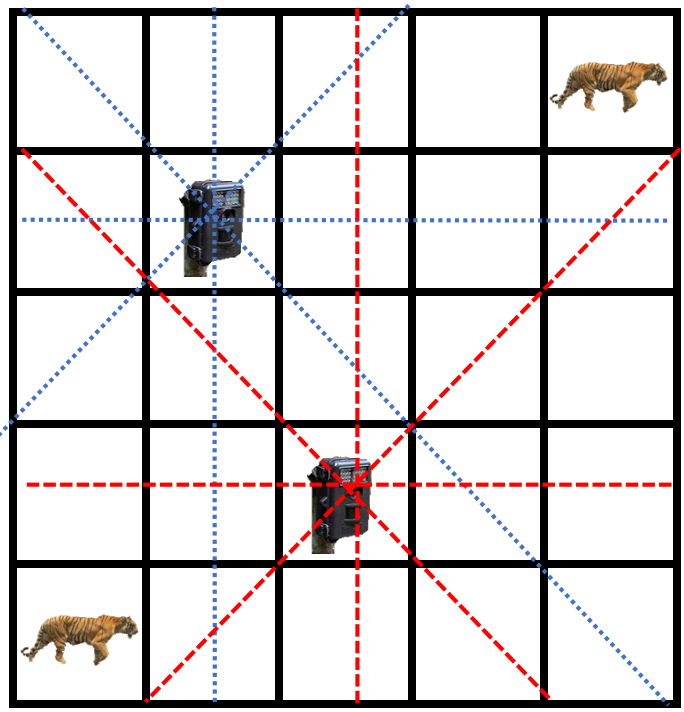
You are helping the World Wildlife Fund (WWF) to place camera traps in a forest for tiger conservation purposes. According to WWF, "camera traps provide data on exactly where species are, what they are doing, and how large their populations are… This is improving our understanding of human impacts on wildlife, and helping land managers make better decisions at both small and large scales." You may read more at the link above.

The goal of your project is to place the camera traps in locations that do not conflict with each other, while maximizing the total number of animals recorded for the day. We must follow these guidelines:

- Camera traps cannot be in the same square, same row, same column, or along the same diagonal. (Think of queens on a chess board)
- Camera traps cannot move.
- Animal images are collected when camera traps are in same square as animals. One point per each image.
- All camera traps 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:

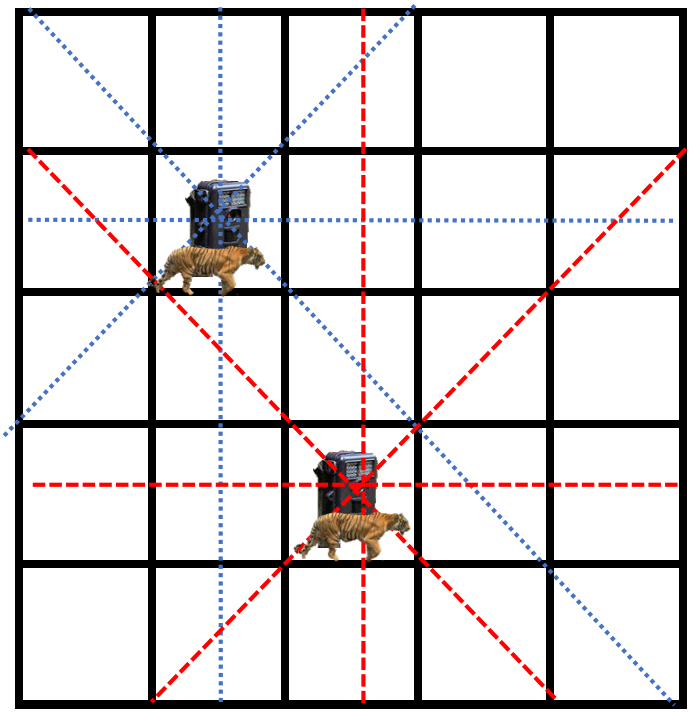
0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

Example 1



The two camera traps have been placed and never move. The red and blue lines show the limitations on placing camera traps; no camera trap may be placed on the same row, column, or diagonal as another camera trap. Since no animals are in the same square as either of the camera traps, no value is gained in Example 1.

Example 2



Since both animals are in the same square as the camera traps, two images are gained in Example 2.

Guidelines

This is a programming assignment. Your program should read a text file named “input.txt” in the current directory that contains a problem definition, and write your solution to a text file named “output.txt” to the current 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 “project1cs360s2019.*”, 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 samples 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 50 test cases used in grading and it is ***your task to make sure that your program will work correctly on any valid input***.

Input: End-of-line character is LF (since Vocareum follows the Unix convention).
First line: strictly positive 32-bit integer ***n***, the width and height of the ***n*** x ***n*** park area, ***n*** <= 15
Second line: strictly positive 32-bit integer ***c***, the number of camera traps, ***c*** <= ***n***
Third line: strictly positive 32-bit integer ***a***, the number of animals, ***a*** <= 100
Fourth line: algorithm to use, either ***astar*** for A* search or ***dfs*** for depth-first search
Next ***a*** lines: the list of animal x,y coordinates, separated with the End-of-line character LF.

Output:
Max number of images: strictly positive 32-bit integer ***m***

Helpful Hints

1. **We will not give unsolvable inputs.** There won't be more cameras 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 camera traps 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**, and any libraries **that are included in Vocareum.**
3. C++ and Java files will be compiled using the following commands (you may confirm this will work by submitting your code for testing on the small submission test cases):
    1. g++ -o project1cs360s2019 project1cs360s2019.cpp
       timeout 180s ./project1cs360s2019
    2. javac project1cs360s2019.java
       timeout 180s java project1cs360s2019
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/Graders/TAs/professors if you are unsure about whether certain actions constitute dishonesty. It is better to be safe than sorry.