Matthew Sellman

Chapter 1: Algorithms

1. Divide and Conquer: Subproblems:

A. Creating the 1D objects – creating the Rabbit and the Cheetah as two small objects. I decided to make them a C and an R on the grid inside the console.

B. Creating the 1D grid – I will be using the console in order to create the grid and will have help from the internet in order to create the program.

C. Getting the rabbit to move away from the cheetah, this would require the rabbit to move away from the cheetah or to move randomly.

D. Implementing the cheetah to move after the rabbit. This would require a pathfinding algorithm in order to make the cheetah move after the rabbit.

E. End the game when objects touch and their positions a line.

1 ½:

Algorithms I meant to use:

A* algorithm: source: `https://gist.github.com/oktapodi/5443979`

Dijkstra's algorithm: source: `http://stackoverflow.com/questions/17480022/java-find-shortest-path-between-2-points-in-a-distance-weighted-map`

2.

Data structure one: Arraylists: I have used Arraylists in order to store the number of walks in the program.

Data structure two: Queues: I used queues as another option to store the number of walks in the program.

Data structure three: Stacks: Stacks were used as a third option for storing the number of walks in the program.

3. Time complexity

Upper bound of algorithms used:

0(n) + 0(1) + 0(n) = 0(n):

Stack, queue, linked list = Time complexity:

Lower bound of algorithms:

0(1) + 0(n) + 0(1) = 0(1):

Stack, queue, linked list = Time complexity:

4.

Methods:

Chapter 2: Implementation

<u>My evaluation of major testing methods:</u>

The intellij debugger is particularly thoughtless and mundane in its approach, however, I managed to make my code work, even when I kept getting array index out of bounds exceptions whenever the cheetah reached the edge of my grid, rather than bounce from the side as it is meant to do.

Why I used particular data structures:

The data structures I used are: stacks, queues and linked lists. I used these data structures to make my life easier when making this program. My program gave the user a choice to choose any of those three data structures in order to store the number of moves that were made by the cheetah. I did this because it allowed me a real diversity of choices when I ran my program.

The advantages or disadvantages of using one block of code first:

The advantages of this is that you can easily access all aspects of your code. This allows you to easily edit and debug the code while not compromising the integrity of the program. Another advantage of this is that the user can see what they are doing.

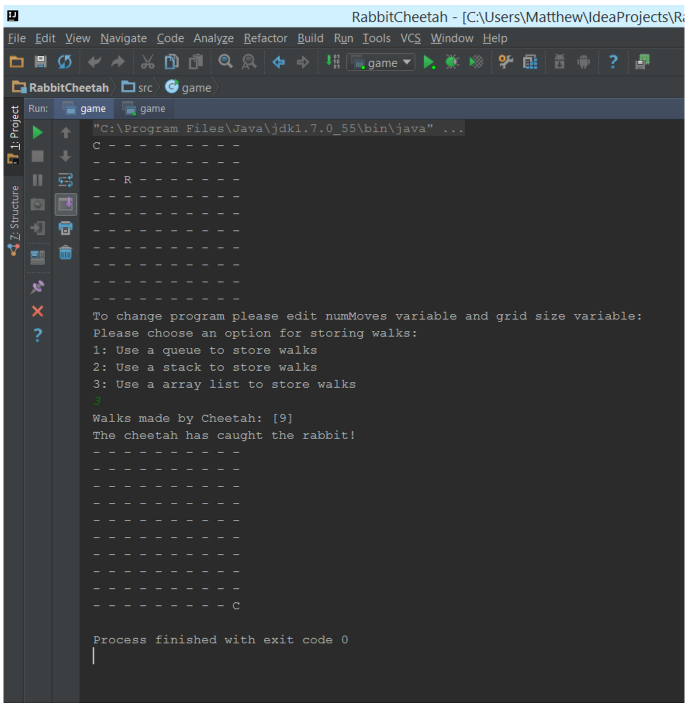
The disadvantages of using my program like this are that it can get muddled and confusing when creating a large scale program. Another disadvantage of using the program like this is that you need to make sure the code is separated out, so that you can read each individual section.

Matthew Sellman

Chapter 3: Demonstration

5.

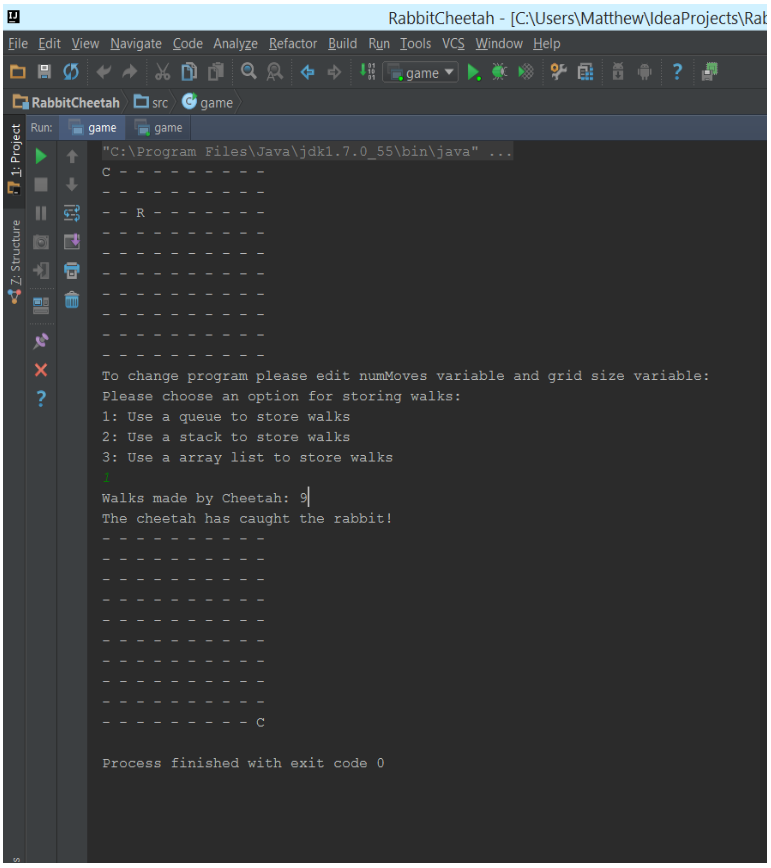
This is my working program. Printing out a new grid every time the cheetah or rabbit moved was far too awkward. Therefore I printed out a grid at the beginning and the end.


RabbitCheetah - [C:\Users\Matthew\IdeaProjects\R
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
game
RabbitCheetah
src
game
Run:
game
game
1: Project
Z: Structure

```
"C:\Program Files\Java\jdk1.7.0_55\bin\java" ...
C - - - - - - - - -
- - - - - - - - - -
- - R - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
To change program please edit numMoves variable and grid size variable:
Please choose an option for storing walks:
1: Use a queue to store walks
2: Use a stack to store walks
3: Use a array list to store walks
3
Walks made by Cheetah: [9]
The cheetah has caught the rabbit!
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - C

Process finished with exit code 0
```

This is my working program where I have used a queue to store the walks. The program above shows the same thing, but using an array list to store the number of walks made by the cheetah.


RabbitCheetah - [C:\Users\Matthew\IdeaProjects\Rat
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
game
RabbitCheetah
src
game
Run:
game
game
1: Project
Z: Structure

```
"C:\Program Files\Java\jdk1.7.0_55\bin\java" ...
C - - - - - - - - -
- - - - - - - - - -
- - R - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
To change program please edit numMoves variable and grid size variable:
Please choose an option for storing walks:
1: Use a queue to store walks
2: Use a stack to store walks
3: Use a array list to store walks
1
Walks made by Cheetah: 9
The cheetah has caught the rabbit!
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - -
- - - - - - - - - C

Process finished with exit code 0
```

Chapter 4: Discussion

Limitations: The main limitation that I found, early on in my program, was the grid. I did not know how I could make the rabbit or cheetah move across the grid, and then paint over them. In order to fix this problem I took to the internet, which was helpful as so far as providing me with similar programs, but none of these actually helped that much. In the end I had to change my entire program from using swing grids to using console.

Another limitation is that the program has to be done inside the console. At first I tried to use a swing grid but that was almost impossible to create. I was therefore forced to move onward and to make the program inside the console. I found this much easier than using the swing grid. Though it didn't look as clean, I realised that it was necessary to only print the grid twice, otherwise the console would be overloaded with different grids that changed slightly every time the cheetah or rabbit moved a square.

Successes:

The program worked, that is a major success because it allowed me to pass.

The switch from swing to using the console made everything so much easier, though this was difficult to accomplish in the time allowed.

What I have learnt:

I have learnt about swing grids and how they can be used. I have also learnt how to make programs or games in the console. This has taught me about algorithms and it has also shown me how I can apply an algorithm to my code.

Bibliography:

I was planning to use Dijkstra and A* algorithms, the sources I would have used for these are below:

A* algorithm: source: https://gist.github.com/oktapodi/5443979

Dijkstra's algorithm: source: http://stackoverflow.com/questions/17480022/java-find-shortest-path-between-2-points-in-a-distance-weighted-map