Project Pre-report:
Android Unlock Patterns Analysis

Motivation

Unlock gesture is a common used unlock pattern on Android smartphones. Normally, it contains 9 plots(3x3) and the user can use a specific gesture which contains some of the plots in a particular order to unlock the phone. Unlike the common unlock pattern using 4 digits as passwords, which has exactly 10000 different combinations, the security of the unlock gesture pattern is not sure for us. Therefore, we are going to figure out how many tries are needed when someone else wants to unlock the phone without knowing the gesture. Then we will also make comparisons about the performance between it other ways like PIN and password.

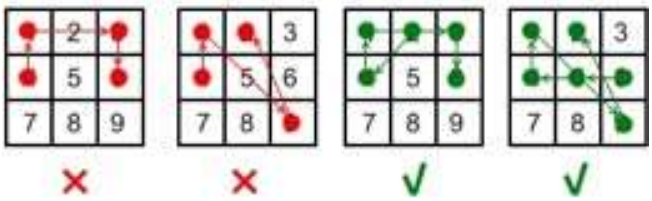
In this project, we will investigate on the Android Unlock Patterns to analyze patterns for unlocking an android smartphone and elaborate on different algorithms to compare the performance. The preliminary project plan consists these four parts:

1. Determine valid gestures;
2. Investigate algorithms that can solve the problem;
3. Compare and analyze performances of different algorithms;
4. Expand the problem to 4x4 and 5x5 situations.

Discussion

1. Determine Valid Gestures

Basically, given an Android 3x3 key lock screen, a valid unlock gesture must contain at least 2 but no more than 9 plots. The gesture must be consecutive and the keys must be distinct. In each valid gesture, the visited plot cannot be visited twice, except that it can be covered in a routine and get visited again. Additionally, every plot in the jumping curve must be marked as a visited plot. And the orders of plots visited matters. We give some valid and invalid gestures below for convenience.



Invalid move: 4 - 1 - 3 - 6

Line 1 - 3 passes through key 2 which had not been selected in the pattern.

Invalid move: 4 - 1 - 9 - 2

Line 1 - 9 passes through key 5 which had not been selected in the pattern.

Valid move: 2 - 4 - 1 - 3 - 6

Line 1 - 3 is valid because it passes through key 2, which had been selected in the pattern.

Valid move: 6 - 5 - 4 - 1 - 9 - 2

Line 1 - 9 is valid because it passes through key 5, which had been selected in the patter

2. The Available Algorithms

To solve this problem, the first idea is using DFS algorithm. Depth-first search (DFS) is an algorithm for traversing or searching tree or graph data structures. One starts at the root (selecting some arbitrary node as the root in the case of a graph) and explores as far as possible along each branch before backtracking. Firstly, we can define a two-dimensional array to store the information whether there is an intermediate plot in the curve between other two plots. Then we are going to define a one-dimensional array to record whether the plot has been visited or not in the gesture curve before. Unlike the enumeration method, we set some constraints when visiting the plots for bootstrap. We attempt to visit the next plot if it has not been visited before by calling the recursive function itself. And if there is an intermediate plot covered in the curve, we will visit it only if it has been not visited before. As for an optimization, we regard that it is equally to choose the plots on four corners of the 3X3 matrix as the starting points. Also, four plots in the middle of four sides of the 3X3 matrix are equal. Apart from these two scenarios, there is one more choice to select the plot at the very center of the 3X3 matrix as the origin. To sum up, we need to consider these three occasions when choosing the starting point.

3. Performance Analysis

For the problem, we will try our best to find more ideas on it. Then we will implement them and make comparisons on the performance of different algorithms. Analyze the order of the growth of running cost on average, best and worst cases. And we will attempt to find the optimization methods on it to reduce the algorithm complexity.

4. Expand

We will expand the scale of the problem to the 4X4 matrix and 5X5 matrix which need more consideration towards different scenarios, if possible.

Project Team:

Tong Wu(tw445)	tw445@scarletmail.rutgers.edu
Xun Tang(xt63)	xt63@scarletmail.rutgers.edu
Xinyu Lyu(xl422)	xl422@scarletmail.rutgers.edu