

# ALGORITHMIC THIEF

## OVERVIEW:-

"Algorithmic Thief" is an Android Application which is targeted on most of the versions of Android operating system.

This Application is a perfect companion for a thief to smuggle the goods from a house.

Application lets user to select the items which are having particular price and weight and also asks for the sack (Bag) size which the user has with him. Then it will process and gives out the most valued items which can fit into his/her Bag.

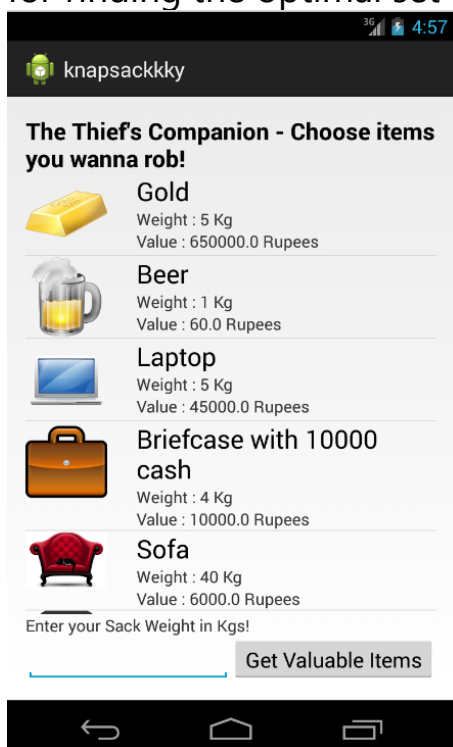
Thief can use this application when he goes out to a house for stealing items but doesn't know what to take which will give him more money in his bag.

"Algorithmic Thief" is made to tackle this problem by implementing knapsack algorithm using top-down approach of dynamic programming.

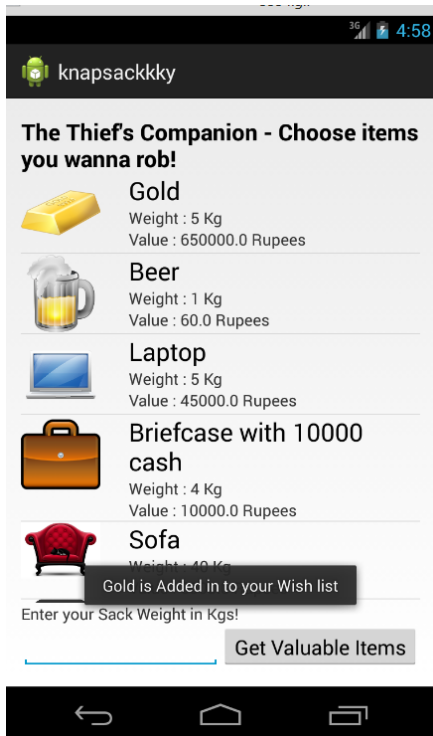
## INTERFACE TO THE USER:

When this application is opened, all the items which are most probable of stealing in a house are listed. Thief has to select the items which he wants to steal from this list.

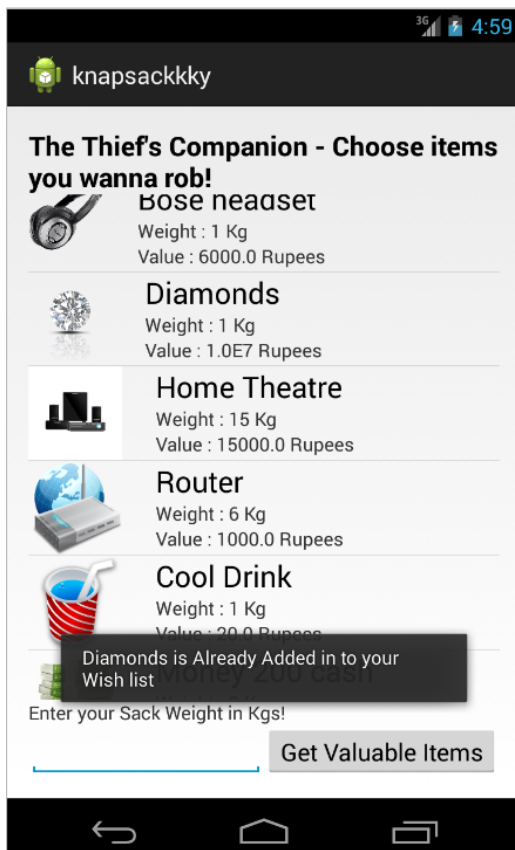
He can only select an item once, as I have used bounded knapsack algorithm for finding the optimal set of items.



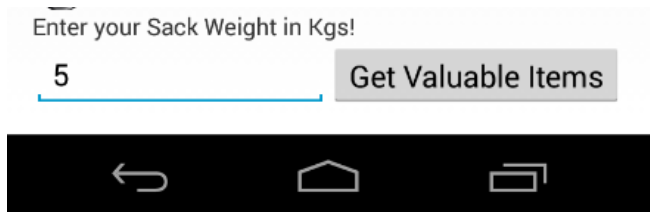
He can scroll down the list for more items in the house and can select by clicking on the item. Toast notification will be displayed upon each click by the user on the item.



Selecting item more than once displays a message "item already added in you wish list"



Next, he has to enter his sack weight which he will use to carry the items home.



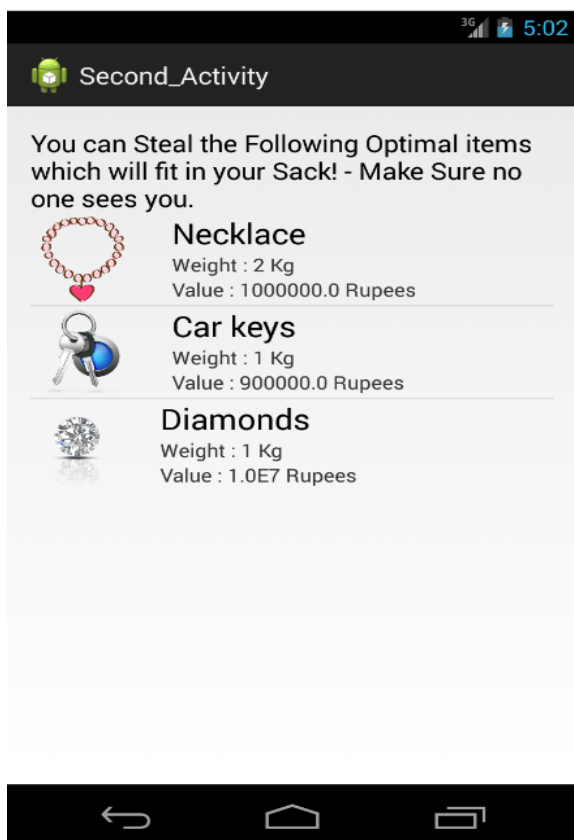
Enter your Sack Weight in Kgs!

5

Get Valuable Items

This screenshot shows a mobile application interface for entering sack weight. It features a text input field with the number '5' and a button labeled 'Get Valuable Items'. The title bar above the input field says 'Enter your Sack Weight in Kgs!'. At the bottom, there is a standard Android navigation bar with back, home, and recent apps icons.

After entering the sack size and clicking the button, the knapsack algorithm will run in the backend and it will compute the best items among the items selected by the user for a given sack size and displays it.



Now Thief can take these items home which would fill his pocket with more money.

### **APPLICATION STRUCTURE:-**

This application solves the classical computer science problem called knapsack.

The application is divided into 4 important classes.

- 1) Main Activity
- 2) Second Activity
- 3) Knapsack class
- 4) Item class

**Main Activity:-**

This class is the layout class which is created by the eclipse and is modified to add items such as ListView, TextView. This controls the layout part of the start screen where the items are displayed.

**Second Activity:-**

This is the class which is used to create the layout and functions of the second activity where the results are displayed

**Item Class:-**

Item class is the building class for the structure of data; it has fields such as name of item, value, weight. As the user selects the item in the user interface appropriate event will generate the item object and will add it to the list

**Knapsack Class:-**

Knapsack class has the top-down recursive code which uses dynamic programming and populates the table.

It has functions like getKnapsack() which takes the input ArrayList<item> which are selected by the user and gives out ArrayList<item> which are optimal list of items.

**WORKING:-**

Initially when the application opens, all the items which are already stored in the list are displayed out to the listView with their values, images, weights. As the user selects the items, the selected item list gets populated and the user gives the weight of his sack which gets stored in the class variable for future use.

Now when the user submits the items by pressing "get optimal items" button, it passes the list of selected items to a knapsack class function getKnapsack(). This function processes the list and returns a new list which has only optimal item references.

This returned list is stored in a static class for future use of it in the other activity.

Now, the activity is changed to second activity by transition and the optimal list is displayed similarly by using listView.

**CONCLUSION:**

This is the starter Application built by me in this platform, I have used my Algorithmic skills to tackle a real life problem which takes non-polynomial time when the data set is large, But by using the Dynamic programming which I have used in my project, we can solve it in  $O(n^2)$  time which is pretty good for an algorithm like knapsack.

This app doesn't exactly meant for thief's, but I took a running example to show how critical the problem is in real life situations and can be adopted to many situations where optimal selection is needed from large data set.