DESIGN AND ANALYSIS OF ALGORITHMS LABORATORY WITH MINI PROJECT [19IS4DLADA]

Presentation

on

" *LOST DORA* "

Helping Dora out in Acapulco

Presented by:

Pankaj Garg	[1DS19IS066]
Rakshitha K	[1DS19IS076]
Pooja Madhav	[1DS19IS068]
Nishanth M Prasad	[1DS19IS065]

LAB IN-CHARGE

Mrs. Bindu Bhargavi S M
Mrs. Shilpa Shree S
Mrs. Sreevidya B S

Department of ISE, DSCE

Story Line

We've all grown up watching Dora. So, we're all well aware about the fact that her sense of direction is not that great. Living in the town of Acapulco, Dora still has a lot trouble getting around her own hometown therefore, why not help her out?

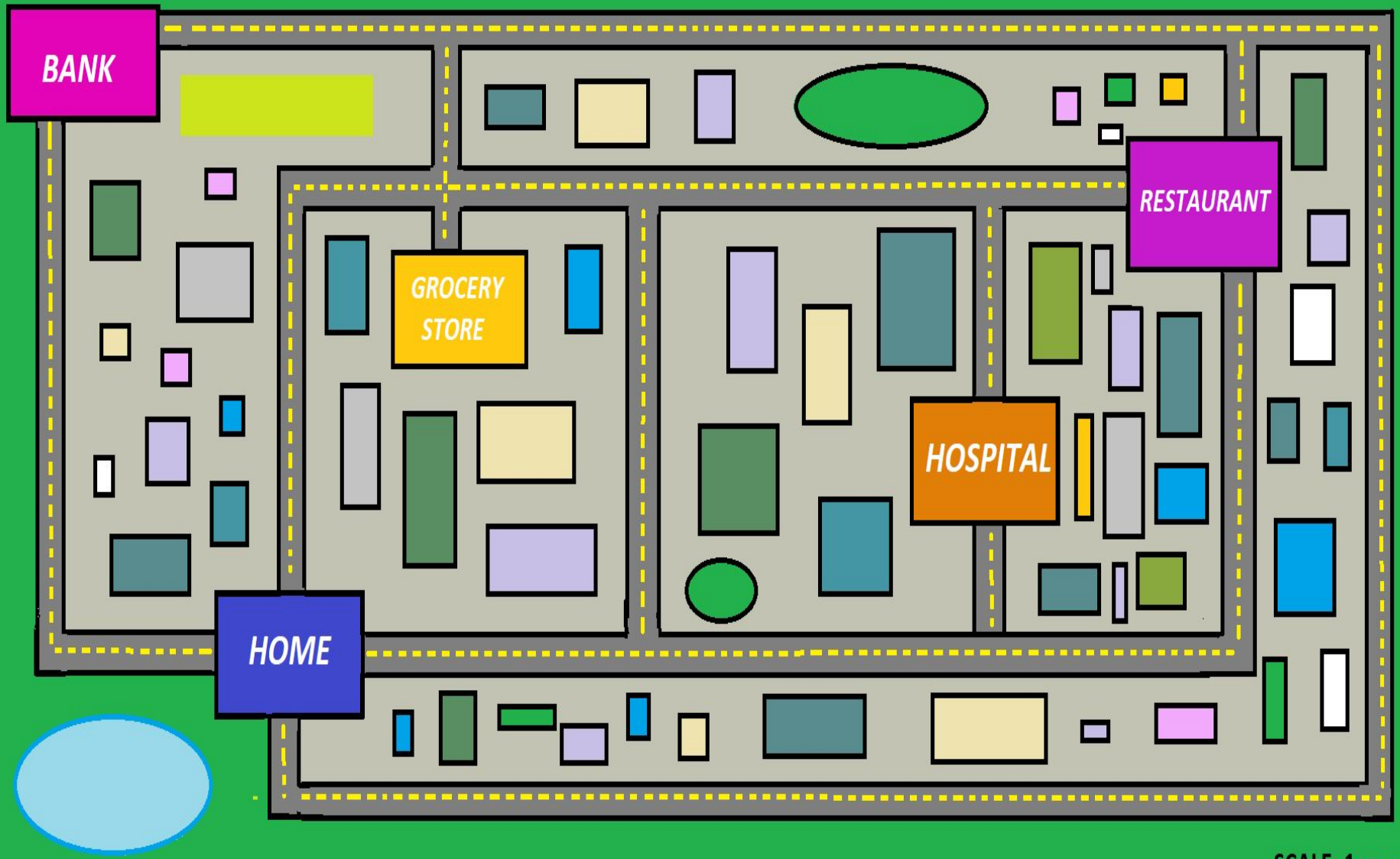
One could say that, we're only returning Dora a favour for all those years of her teaching us alphabets, numbers, colours and even basic manners.

System Requirement:

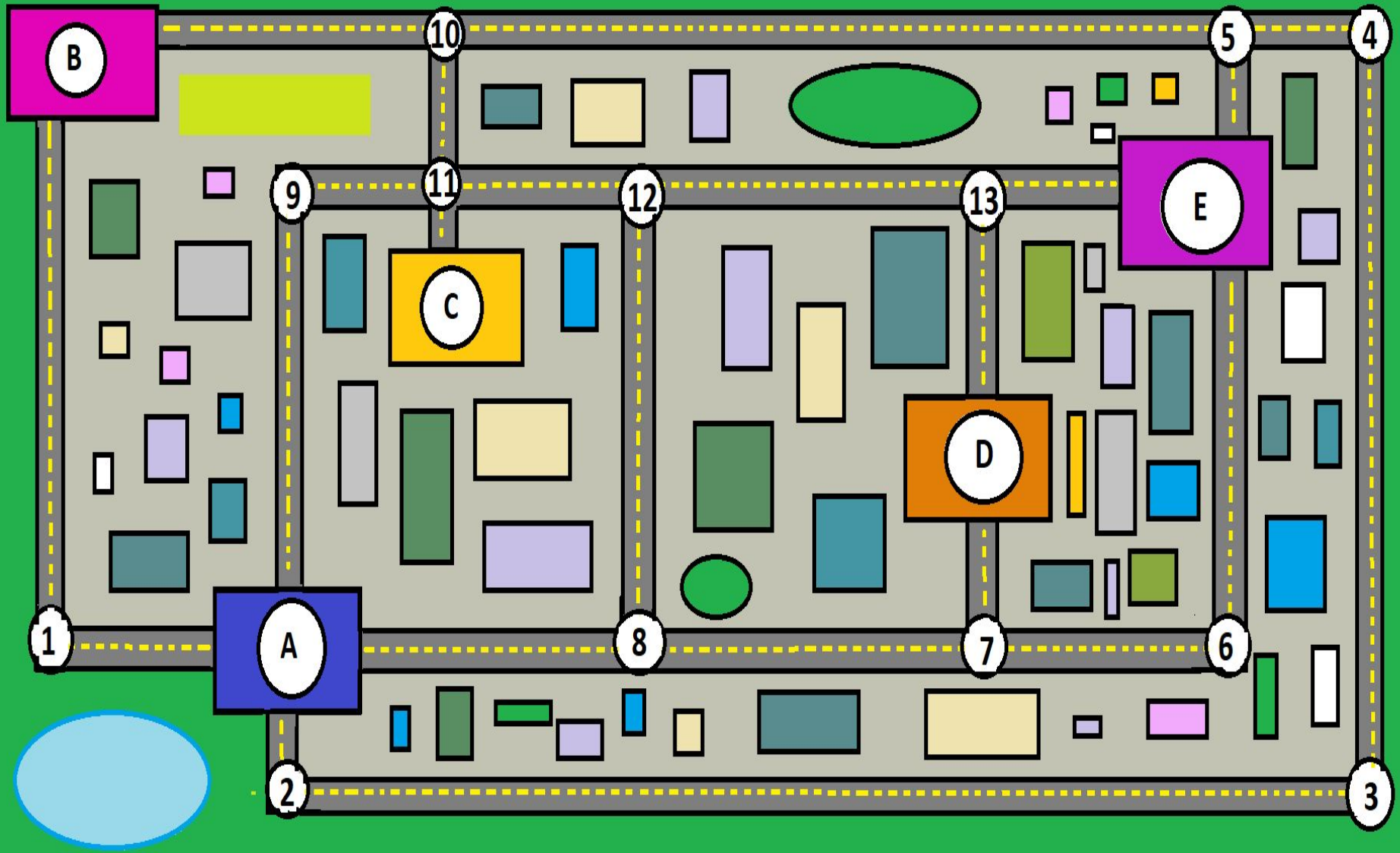
This code will be compatible with any system having any IDE and any c++ compiler.

Pc: Except Turbo C++. 

Acapulco's Map


BANK
RESTAURANT
GROCERY STORE
HOSPITAL
HOME
SCALE: 1cm = 3km

Let’s see the same map with respect to the undirected graph data structure by representing each building and turn on the road as a node and each path between node as an edge.


B
10
5
4
9
11
12
13
E
C
D
1
A
8
7
6
2
3

Inputs in form of adjacency matrix :

	A	B	C	D	E	1	2	3	4	5	6	7	8	9	10	11	12	13
A	999	999	999	999	999	10.5	4.5	999	999	999	999	999	18	17.4	999	999	999	999
B	999	999	999	999	999	24	999	999	999	999	999	999	999	999	18	999	999	999
C	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	2.4	999	999
D	999	999	999	999	999	999	999	999	999	999	999	6	999	999	999	999	999	7.8
E	999	999	999	999	999	999	999	999	999	4.2	18	999	999	999	999	999	999	9
1	10.5	24	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999	999
2	4.5	999	999	999	999	999	999	72	999	999	999	999	999	999	999	999	999	999
3	999	999	999	999	999	999	72	999	34.5	999	999	999	999	999	999	999	999	999
4	999	999	999	999	999	999	999	34.5	999	9	999	999	999	999	999	999	999	999
5	999	999	999	999	4.2	999	999	999	9	999	999	999	999	999	52.5	999	999	999
6	999	999	999	999	18	999	999	999	999	999	999	16.5	999	999	999	999	999	999
7	999	999	999	6	999	999	999	999	999	999	16.5	999	24	999	999	999	999	999
8	18	999	999	999	999	999	999	999	999	999	999	24	999	999	999	999	19.5	999
9	17.4	999	999	999	999	999	999	999	999	999	999	999	999	999	999	9	999	999
10	999	18	999	999	999	999	999	999	999	52.5	999	999	999	999	999	6	999	999
11	999	999	2.4	999	999	999	999	999	999	999	999	999	999	9	6	999	13.5	999
12	999	999	999	999	999	999	999	999	999	999	999	999	19.5	999	999	13.5	999	22.5
13	999	999	999	7.8	9	999	999	999	999	999	999	999	999	999	999	999	22.5	999

Algorithms used :

Dijkstra algorithm- Shortest Path in the Map

Change Making algorithm- Bank

Binary search- Hospital

Bubble Sort- Restaurant

Knapsack algorithm- Grocery Store

Dijkstra's Algorithm

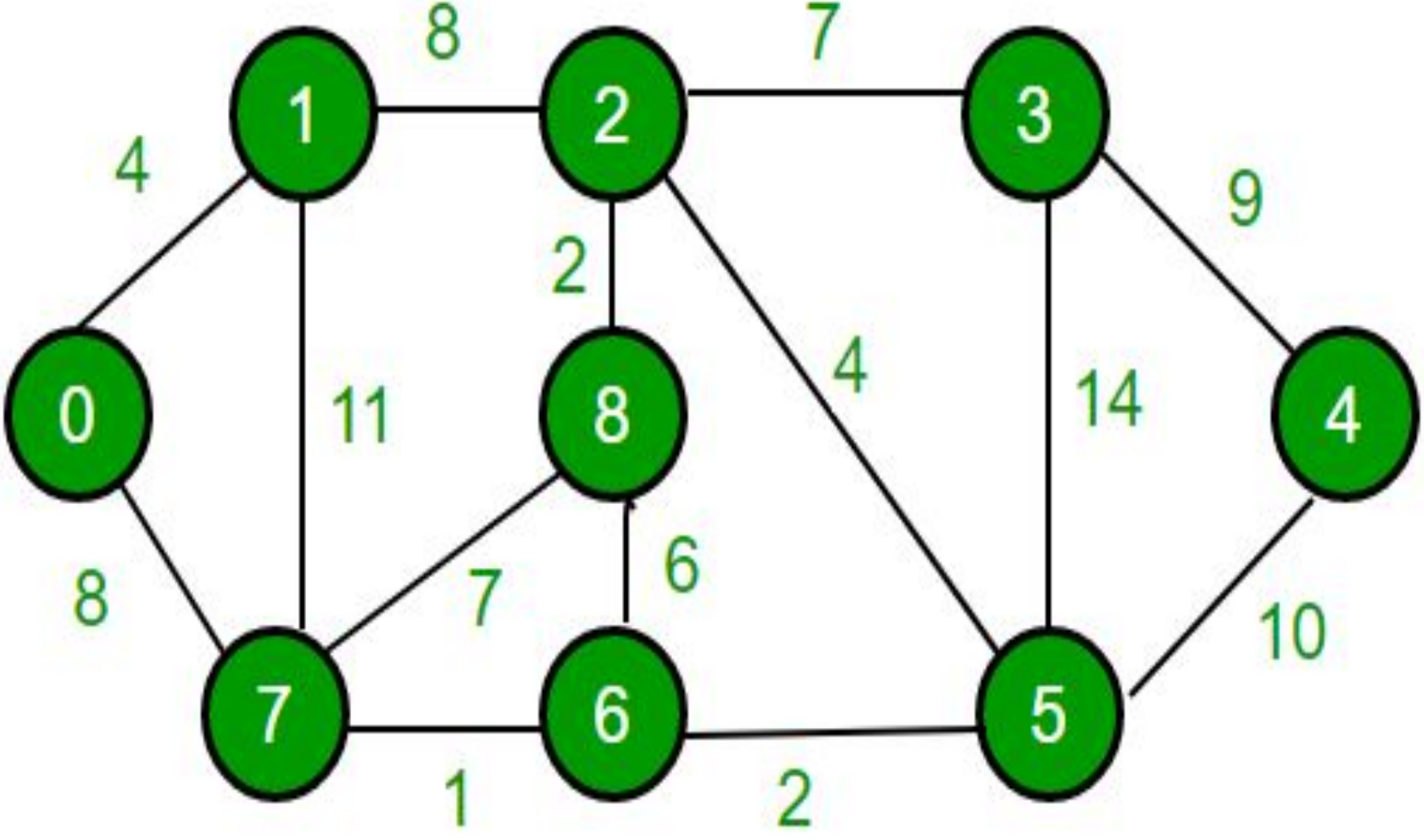
An algorithm comes under the principle of Greedy Technique .

It is used for finding the shortest path from a source node to a target node in a weighted graph.

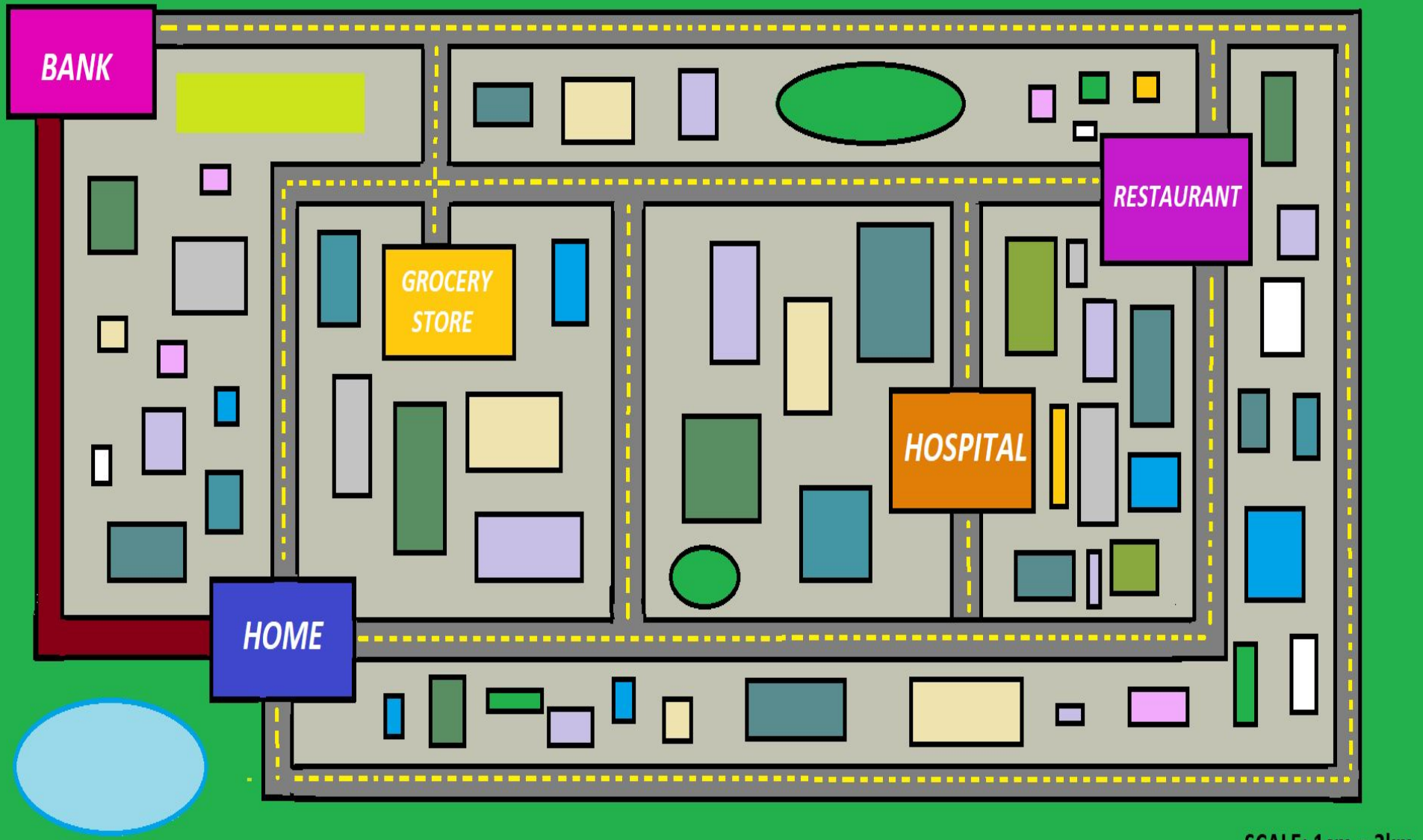
The algorithm creates a tree of shortest paths from the starting vertex, the source, to all other points in the graph.

Steps/Algorithm

1. Mark all nodes as unvisited.
2. Assign a distance value to every node : Set our source node to and all other nodes as infinity.
3. For the current node, consider all of its unvisited neighbours and calculate their distances through the current node.
4. Compare the pre assigned value to the current value and assign the smaller one.
5. When we are done considering all of the unvisited neighbours of the current node, mark the current node as visited, a visited node will never be checked again.
6. Repeat the above steps for every node.
7. The distance calculated for each node after performing all the above steps will give us the shortest distance from source node to that particular node.


0
1
2
3
4
5
6
7
8
4
8
8
11
7
2
4
7
6
14
9
10
1
2


BANK


BANK
RESTAURANT
GROCERY STORE
HOSPITAL
HOME
SCALE: 1cm = 3km

Change Making Problem

The change-making problem addresses the question of finding the minimum number of coins (of certain denominations) that add up to a given amount of money.

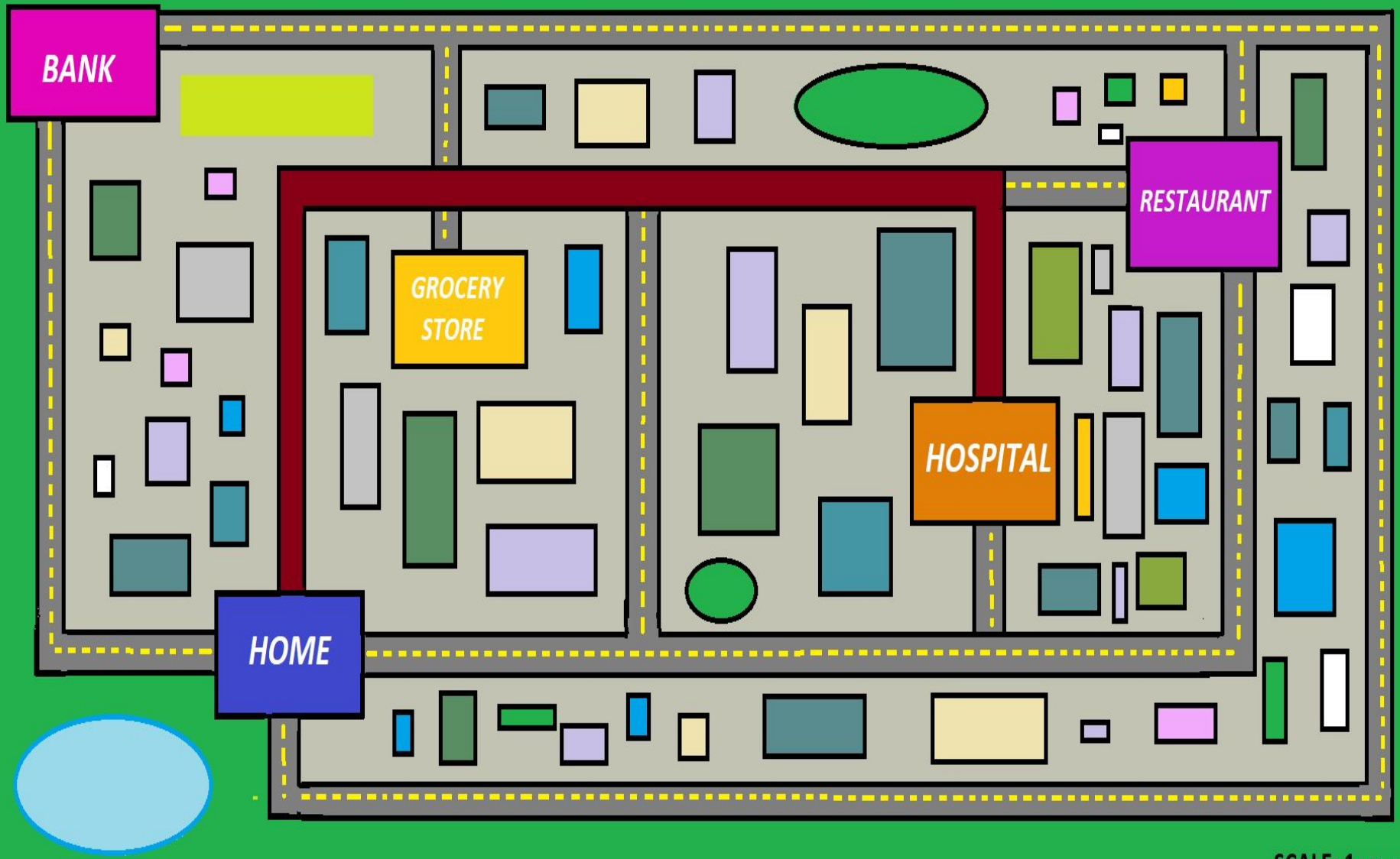
GREEDY APPROACH:

A common intuition would be to take coins with greater value first. This can reduce the total number of coins needed. Start from the largest possible denomination and keep adding denominations while the remaining value is greater than 0.

Algorithm:

1. Sort the array of coins in decreasing order.
2. Initialize result as empty.
3. Find the largest denomination that is smaller than current amount.
4. Add found denomination to result. Subtract value of found denomination from amount.
5. If amount becomes 0, then print result.
6. Else repeat steps 3 and 4 for new value of V.


HOSPITAL


BANK
RESTAURANT
GROCERY STORE
HOSPITAL
HOME
SCALE: 1cm = 3km

Binary Search

Binary search is an efficient algorithm for finding an item from a sorted list of items.

This search algorithm works on the principle of divide and conquer.

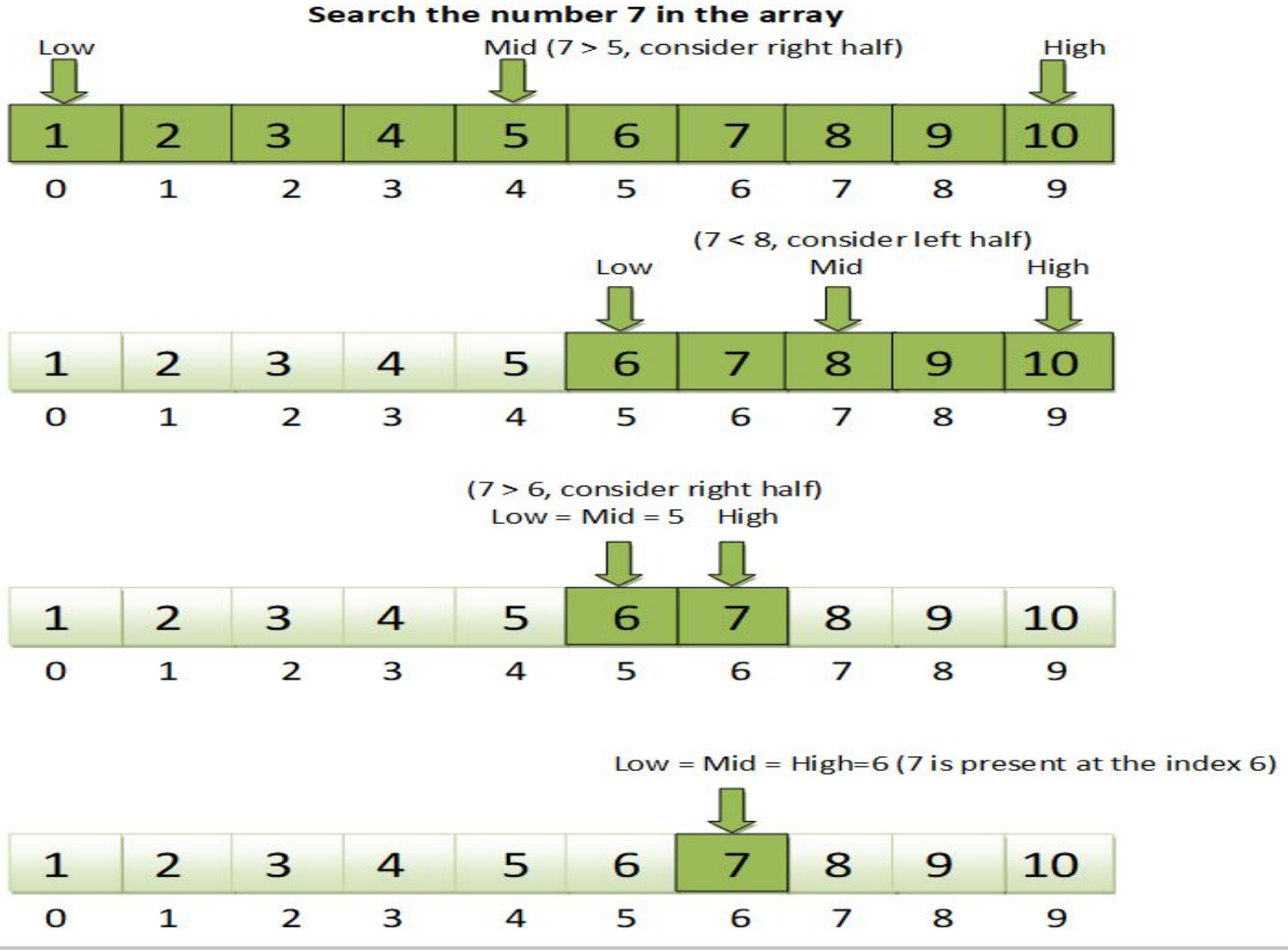
Logic/Algorithm:

Search a sorted array by repeatedly dividing the search interval in half.

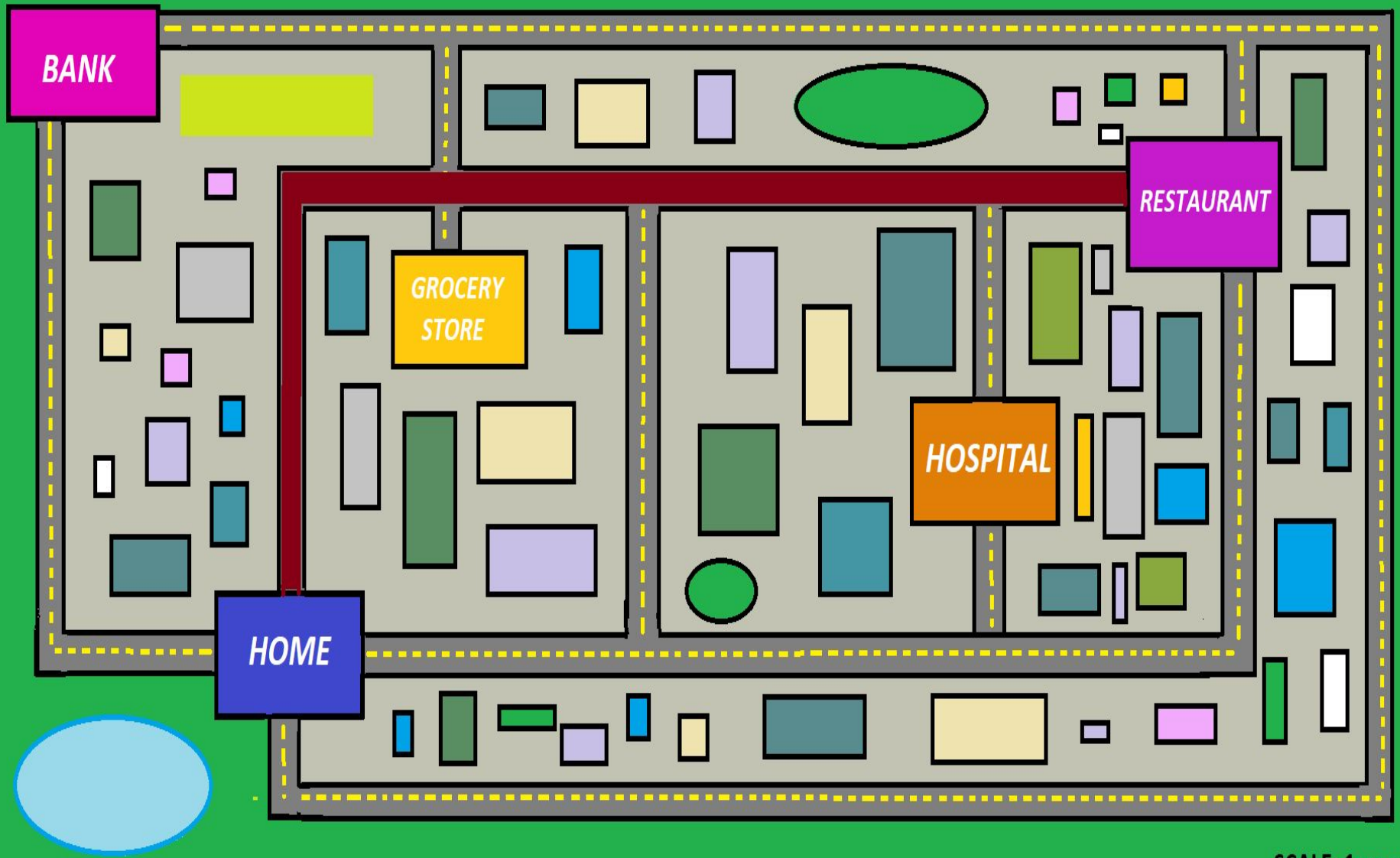
Begin with an interval covering the whole array.

If the value of the search key is less than the item in the middle of the interval,narrow the interval to the lower half. Otherwise, narrow it to the upper half repeatedly until the middle element points to the search key.

Binary Search




RESTAURANT

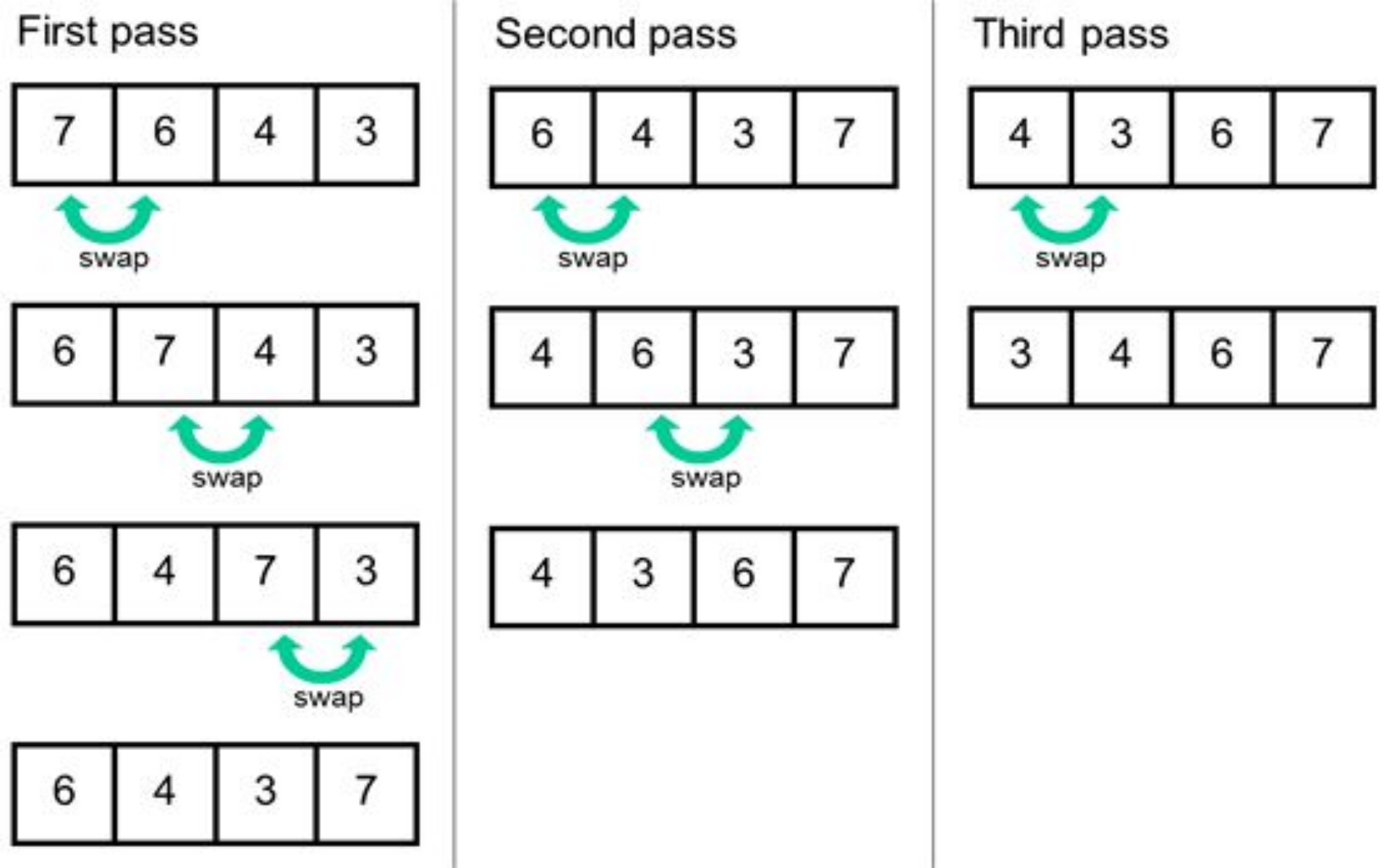

BANK
RESTAURANT
GROCERY STORE
HOSPITAL
HOME
SCALE: 1cm = 3km

Bubble Sort

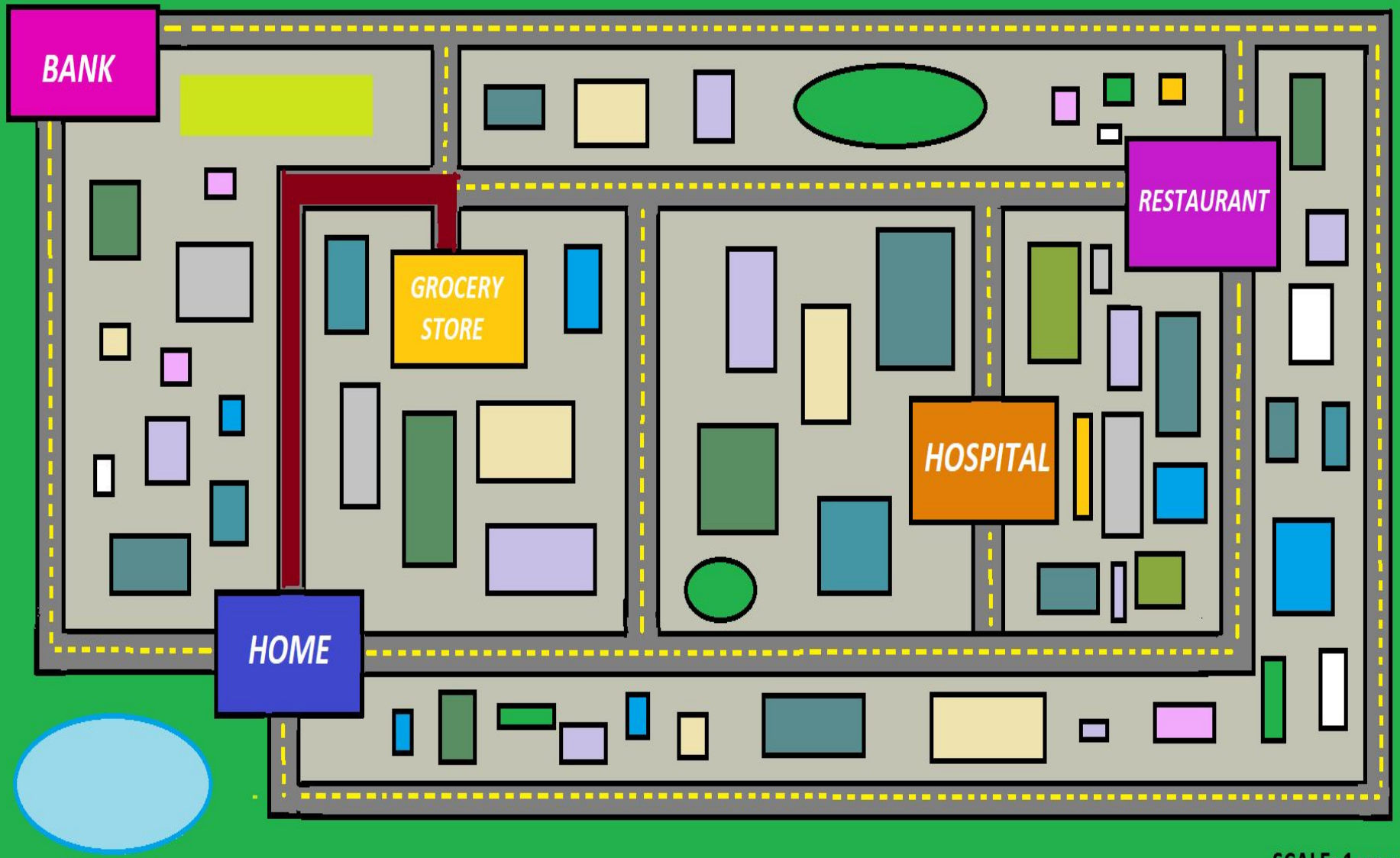
Bubble sort is one of the simple sorting algorithms and also popularly known as a **Brute Force Approach.**

The logic of the algorithm is very simple as it works by repeatedly iterating through a list of elements, comparing two elements at a time and swapping them if necessary until all the elements are swapped to an order.

The greatest number is bubbled to the last after each iteration.


First pass
7 6 4 3
swap
6 7 4 3
swap
6 4 7 3
swap
6 4 3 7
Second pass
6 4 3 7
swap
4 6 3 7
swap
4 3 6 7
Third pass
4 3 6 7
swap
3 4 6 7


GROCERY
OPEN
FRESH
BREAD


BANK
RESTAURANT
GROCERY STORE
HOSPITAL
HOME
SCALE: 1cm = 3km

KnapSack Algorithm

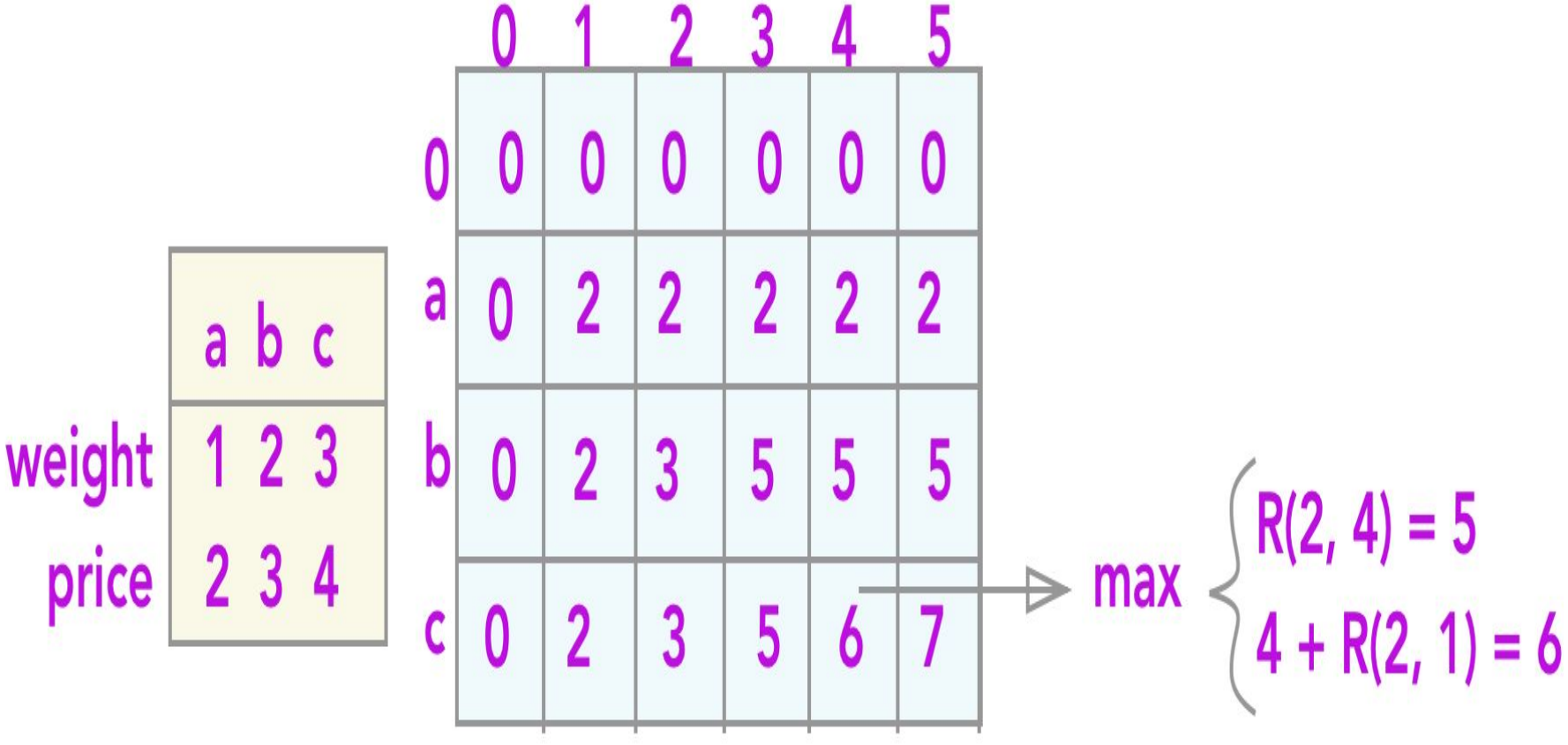
The knapsack problem is an optimization problem.

Given a set of items, each with a weight and a value, we need to determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.

0-1 knapsack problem follows dynamic programming technique .

In 0-1 knapsack item is either taker or not taken, fractional values can not be considered.

$$V[i, w] = \max(V[i-1, w], v_i + V[i-1, w-w_i])$$



Time Complexity :

Dijkstra algorithm:

Adjacency list: O(E logV) **Adjacency matrix : O(v^2)**

Change making algorithm:

Best: O(1) **Worst: O(n)**

Binary search:

Best: O(1) **Worst: O(log n)**

Efficient Bubble sort:

Best: O(n) **Worst: O(n^2)**

The overall Time Complexity of the entire algorithm will be O(n^2).

THANK YOU

ANY QUESTIONS ?