**VISVESVARAYA TECHNOLOGICAL UNIVERSITY**

**“Jnana Sangama”, Belagavi-560018, Karnataka**



**DESIGN AND ANALYSIS OF ALGORITHMS LABORATORY**

**WITH MINI PROJECT-18IS4DLADA**

**REPORT**

**On**

**“Intelligent Parking System”**

**BACHELOR OF ENGINEERING**

In

**INFORMATION SCIENCE AND ENGINEERING**

Submitted by-

**Chiranthana.M (1DS18IS124)**

**Yash Agarwal (1DS18IS119)**

**Surya .R (1DS19IS410)**

**LAB IN-CHARGE**

Dr. Chandrakala B M

Mrs. RadhikaT.V

Mr. Dheemanth Urs R



**2019-20**

**Department Of Information Science and Engineering**

**DAYANANDA SAGAR COLLEGE OF ENGINEERING**

**SHAVIGE MALLESHWARA HILLS, KUMARASWAMY LAYOUT**

**Bengaluru-560078**

**DAYANANDA SAGAR COLLEGE OF ENGINEERING**

**Shavige Malleshwara Hills, Kumaraswamy Layout-560078**

**Department of Information Science and Engineering**



**2019-20**

**CERTIFICATE**

This is to certify that Design and Analysis of Algorithms Laboratory with Mini Project Work entitled **“ Intelligent Parking System ”** is a bonafide work carried out by **Chiranthana.M(1DS18IS124), Yash Agarwal(1DS18IS119), Surya.R(1DS19IS410)** in the partial fulfillment for the 4th semester of **Bachelor of Engineering in Information Science and** **Engineering** of theVisvesvaraya Technological University, Belgaviduring the year 2019-20.

------------------------------------- ------------------------------ ---------------------------

|  |  |  |
| --- | --- | --- |
| Signature of Lab-Incharge | Signature of Lab-Incharge | Signature of Lab-Incharge |
| [ Dr.Chandrakala B M] | [Mrs.Radhika T V] | [Mr.Dheemanth Urs. R] |

Name of the Examiners Signature with Date

1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Sr. No** | **Topics** | **Page No.** |
|  | Introduction | 4 |
|  | System Requirements | 5 |
|  | Algorithm | 5-6 |
|  | Implementation |  |
|  | 1. Pseudocode | 6-9 |
|  | 1. Snapshots of Results | 9-11 |
|  | Time Complexity | 12 |

**Introduction:**

As urban centers are becoming increasingly more populated and congested, the management of public parking places, according to local mobility policies, is a key strategic issue with high economic impact. Academia and industry are well aware of the potential gains that can be obtained taking advantage of information and communication technologies applied to this field. An efficient smart parking system will favor drivers, by reducing the time to find a parking place and by optimizing parking spaces’ utilization rate, thus increasing the revenues for the urban parking operators.​

The importance of applying the adequate strategy for the management of parking places can also be found in the estimate that during the day a typical vehicle stays in idle 95% of the time.​

The driver’s behavior, when choosing the type and the location of the parking place as well as making the final decision, depends on many factors and represents a considerably more complex process, which should be paid much more attention than before. The usage of advanced technologies is unavoidable in achieving a selected combination of strategies.​

A simple coding and design pattern has been used to automate the browsing through the parking lots, checking out empty spots, finding out whether it is available for advance booking, and fare.

​

Features -​

Login Register: Users and parking lot owners can register into the portal and will be allowed to browse through the list of places-> parking lots-> parking spots. ​

Current Booking: Users can check out the availability of empty spots currently and book the spot immediately.​

Future Booking: If a user needs an advance booking, he can do so by booking the car/bike from a fixed time to a fixed time. ​

**System Requirements:**

* Min Processor Type - Intel 386 or higher
* Min RAM Size - 4 MB
* Min Hard Drive Space - 25 MB
* OS Required - Microsoft DOS, Microsoft Windows 3.1 or later, PC DOS
* OS Family - Windows
* Min Processor Type - Intel 386 or higher

**Algorithm used :**

**Distance of nearest cell:-**​

Given a binary matrix of N x M, containing at least a value 0. The task is to find the distance of the nearest 1 in the matrix for each cell. The distance is calculated as |i1 – i2| + |j1 – j2|, where i1, j1 are the row number and column number of the current cell and i2, j2 are the row number and column number of the nearest cell having value 0.​

Approach:

The idea is to traverse the matrix for each cell and find the minimum distance, To find the minimum distance traverse the matrix and find the cell which contains 0 and calculates the distance between two cells and store the minimum distance.​

STEP 1. Traverse the matrix from start to end (using two nested loops)​

STEP 2. For every element find the closest element which contains 0. To find the closest element traverse the matrix and find the minimum distance.​

STEP 3. Fill the token number in the matrix.​

Sparse Matrix:-​

A matrix is a two-dimensional data object made of m rows and n columns, therefore having a total m x n values.​

A 2D array is used to represent a sparse matrix in which there are three rows named as :​

Row: Index of the row, where the non-zero element is located​

Column: Index of the column, where a non-zero element is located​

Value: Value of the non zero element located at index – ( row, column )​

**Implementation**

**Pseudo Code:**

view(n,m)

print(Parking Lot)

print([0 denotes empty slot]

for(i<-0 to n) do

for(j<- to m) do

print(Matrix(i,j))

exit(n,m)

if(current=0)

print(No vehicle in parking lot)

else

print(Enter token number)

input(token)

for(i<-0 to n) do

for(j<-0 to m) do

if(token=Matrix(i,j))

a=1

current<-currnet-1

Matrix(i,j)

xecord=n-i

yecord=m-j

print(Total units to exit = xecord+yecord+n+1)

print(Directions for exit)

if(a=0)

print(Wrong token number)

entry(n,m)

if((n\*m)==current)

print(Parking Lot is full)

else

c++;

current++;

print("##\*\*##\*\*##\*\*##\*\*##\*\*##\*\*##\*\*##")

int s=9999,total,xcord,ycord;

for(i<-0;i<n;i++)

for(j<-0;j<m;j++)

if(Matrix(i,j)==0)

total = i+j;

if(total<=s)

s=total;

xcord=i;

ycord=j;

print("Your TOKEN NUMBER is:")

print(Total units to your slot = )

print(Your Parking slot cordinates is: go "

print(ycord)

print(units straight, and)

print(go)

print(xcord+1)

print(units right)

Matrix[xcord][ycord]=c;

sparsematrix(n,m)

print(Empty positions in parking lot)

for(i<-0 to n) do

for(j<-0 to m) do

if(Matrix(i,j)=0)

size=size+1

k<-0

for(i<-0 to n) do

for(j<-0 to m) do

if(Matrix(i,j)=0)

sMatrix(0,k)=i

sMatrix(1,k)=j

sMatrix(2,k)=Matrix(i,j)

k<-k+1

for(i<-0 to 3)

if(i=0)

print(Row:)

else if(i=1)

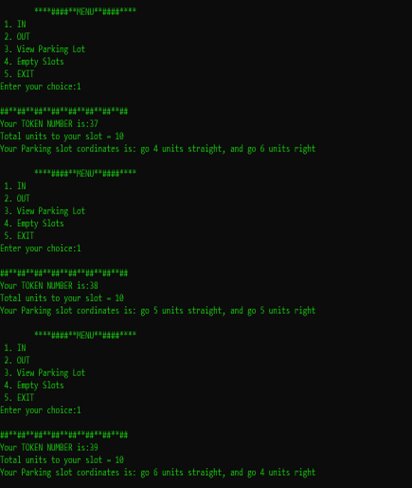
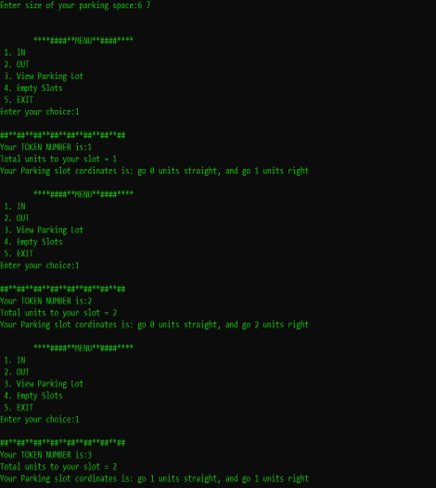
print(Column:)

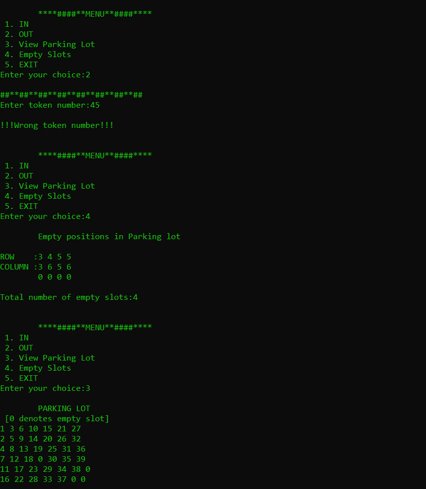
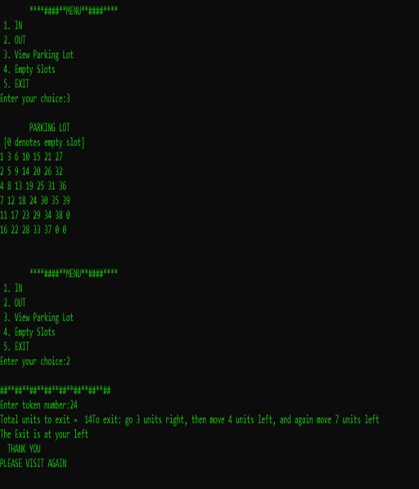
for(j<-0 to k) do

print(sMatrix(i,j))

print(Total number of empty slots = (n\*m)-current)

**Snapshots of Results :**



**Time Complexity:**

Time complexity is calculated during the entry of a vehicle for searching the nearest empty slot for parking.​

​n=number of rows in the parking lot​

m=number of columns in the parking lot​

​

Worst time complexity: O(n\*m) :- last slot is empty ​

​

Average time complexity: any slot between first and last position is empty​

​

Best time complexity: O(1) :- first slot is empty​

**Thank You**