TEAM CHALLENGERS
BATCH NUMBER: 20
AIE

CRYPTOCURRENCY BEST VALUE OPTIMIZATION

A REPORT

Submitted by

JASWANTH M	BL.EN.U4AIE19038
NKL NARAYANA	BL.EN.U4AIE19046
S RAHUL	BL.EN.U4AIE19061

for the course

19AIE212 -Design and Analysis of Algorithms

AMRITA SCHOOL OF ENGINEERING, BANGALORE

AMRITA VISHWA VIDHYAPEETHAM

BANGALORE-560 035

TEAM CHALLENGERS
BATCH NUMBER: 20
AIE

TABLE OF CONTENTS

TEAM CHALLENGERS
BATCH NUMBER: 20
AIE

A REPORT ABOUT CRYPTOCURRENCY BEST VALUE OPTIMIZATION

ABSTRACT:

One of the greatest innovations in this era of digital age is that
of cryptocurrencies. Consider the scenario of the crypto currencies and the cryptocurreny markets. Everyday, a huge number of people trade equities and commodities on various exchanges across the globe. These processes are being technically equipped, and a majority of the functions are being automated. Over the years, cryptocurrencies have earned immense popularity from all across the globe. Cryptocurrencies are designed to function as money. They are considered to be an alternative to the fiat currencies of the world. Trading in the cryptocurrency market comes as no big a surprise. A lot of people engage themselves in this market depending upon their risk-taking capability. The growing acceptance of cryptocurrencies both as an investment opportunity and also as a medium of exchange is all the more a strong reason to talk about them in today's technology-rich world. In this report, we will examine the ability of forecasting Cryptocurrencies price trends. Observing a long period of time that includes volatile periods with strong up and downtrends introduces challenges to any forecasting system. With a given list of historical prices over a specific period, we will compute the maximum possible profit from the historical prices. One can perform an action for buying or selling at a given timeline. The total number of buying and selling operations that can be executed is not limited. We will use the divide and conquer paradigm to solve the maximization problem. The divide and conquer method recursively breaks down the problem into smaller sub-problems until they can be solved directly. The solutions to these sub-problems are then combined to give a solution to the original problem. We use divide and conquer algorithm for optimization of cryptocurrency to find the best strategies.

TEAM CHALLENGERS
BATCH NUMBER: 20
AIE

INTRODUCTION:

A cryptocurrency is a virtual currency that is protected by cryptography, which prevents it to be double-spent and counterfeited. Many cryptocurrencies are decentralized networks based on the blockchain technology which relies on distributed ledgers enforced by a diverse network of computers. This ensures that transactions made with cryptocurrencies are anonymous and untraceable, which enables both parties involved in the transaction to not share their sensitive data with third parties such as banks , other transaction providers. Day trading is one of the most commonly used trading strategies used in both – the stock market as well as the cryptocurrency market. Those who indulge in day trading employ with in the day strategies to earn profits from the volatile market. A lot of techniques are used to earn from intraday trading. Day trading is a strategy that revolves around the opening and closing positions several times during a single day. Another important feature of day trading is that the trader closes all positions at the end of a given day.We will be using a set of Day trading values during implementation. Algorithmic trading represents one such area of interest for brokerage firms for cryptocurrency. Algorithmic trading involves coming up with new algorithms to predict the future price of crypto currencies. Algorithmic Trading is used to the maximum possible profit. we will be using Divide and Conquer method and predict the best time to buy crypto currencies based on previous data.

GUI USED :

For this project, we used PyQt5 GUI framework.PyQt is a GUI widgets toolkit. It is a Python interface for Qt, one of the most powerful GUI library. PyQt is a blend of Python programming language and the Qt library. This introductory tutorial will assist you in creating graphical applications with the help of PyQt.PyQt5 is set of cross-platform libraries that implement high-level APIs for accessing many aspects of modern desktop and mobile systems. These include location and positioning services, multimedia, NFC and Bluetooth connectivity, a Chromium based web browser, as well as traditional UI development. PyQt5 is a comprehensive set of bindings for Qt v5. It is implemented as more than 35 extension modules and enables IDE's to be used as an alternative application development language on all supported platforms including iOS and Android.PyQt5 may also be embedded in applications to allow users of those applications to configure , enhance the functionality of those applications.

IMPLEMENTATION PROCESS:

We will use the divide and conquer paradigm to solve the cryptocurrency maximization problem. The divide and conquer method recursively breaks down the problem into smaller sub-problems until they can be solved directly. The solutions to these sub-problems are then combined to give a solution to the original problem. We also discussed the advantages and disadvantages of the divide and conquer paradigm. we are using gui platform pyqt5 to make our algorithm user friendly. we take the inputs as dates from the user and get the data of prices of cryptocurrency using web scrapping. Web scrapping is the process of using bots to extract content and data from a website. Unlike screen scraping, which only copies pixels displayed onscreen, web scraping extracts underlying HTML code and, with it, data stored in a database. The scraper can then replicate entire website content elsewhere. now with the data obtained from web, we process the above mentioned algorithm to get maximum profit from the prices of the cryptocurrency.

CODE:

```
from PyQt5 import QtCore, QtGui, QtWidgets
from PyQt5.QtGui import *
from PyQt5.QtCore import *
import csv
import requests
from bs4 import BeautifulSoup
from urllib.request import Request, urlopen
from pyqtgraph import PlotWidget, plot
import pyqtgraph as pg
import cryptocompare
from datetime import date


class Ui_MainWindow(object):

    def popup(self):
        self.calendarWidget.setVisible(True)
        self.calendarWidget.raise_()


    def calendar(self):
        dateselected = self.calendarWidget.selectedDate()
        date_in_string = str(dateselected.toPyDate())
        self.label_4.setText(date_in_string)
        self.calendarWidget.setVisible(False)
        self.date_in_string_to = date_in_string
```

```
    def popup_1(self):
        self.calendarWidget_1.setVisible(True)
        self.calendarWidget_1.raise_()

    def calendar_1(self):
        dateselected_1 = self.calendarWidget_1.selectedDate()
        date_in_string_1 = str(dateselected_1.toPyDate())
        self.label_5.setText(date_in_string_1)
        self.calendarWidget_1.setVisible(False)

        self.date_in_string_from = date_in_string_1

    def converter(self,text):

        MONTHS = ['January', 'February', 'Match', 'April', 'May', 'June','July', 'August', 'September',
'October', 'November', 'December']
        y = text[:4]
        m = int(text[5:-3])
        d = text[8:]
        month_name = MONTHS[m-1]
        result= month_name + ' ' + d + ' '  + y
        return result

    def scrape_data_tab_1(self):
        self.line.clear()
        for_graph=[]
        if(self.r6.isChecked()):
```

```
    self.url="https://in.investing.com/crypto/bitcoin/historical-data"
if(self.r7.isChecked()):
    self.url="https://in.investing.com/crypto/dogecoin/historical-data"
if(self.r8.isChecked()):
    self.url="https://in.investing.com/crypto/ethereum/historical-data"
if(self.r9.isChecked()):
    self.url="https://in.investing.com/crypto/cardano/historical-data"
if(self.r10.isChecked()):
    self.url="https://in.investing.com/crypto/tether/historical-data"
list_data=[]
r = Request(self.url, headers={"User-Agent": "Mozilla/5.0"})
html_content=urlopen(r).read()
soup = BeautifulSoup(html_content,"html.parser")
self.tableWidget_1.setRowCount(0)
table = soup.find_all('table')[0]
rows = table.select('tbody > tr')
header = table.select('thead > tr')
headers=[header[0].find_all('th')[i].text.rstrip() for i in range(len(header[0].find_all('th')))]
self.tableWidget_1.setHorizontalHeaderLabels(headers)
for row in rows:
    data = [th.text.rstrip() for th in row.find_all('td')]
    self.addTableRow(self.tableWidget_1,data)
    for x in range(len(data)):
        data[x]=data[x].replace(',',"")
    for_graph.append(data)
for_graph_price_list=[]
for_graph_date_list=[]
for i in range(len(for_graph)):
```

```
            for_graph_price_list.append(float(for_graph[i][2]))
            for_graph_date_list.append(len(for_graph)-i)
        for_graph_price_list=for_graph_price_list[::-1]
        for_graph_date_list=for_graph_date_list[::-1]
        self.line=self.graphWidget.plot(for_graph_date_list, for_graph_price_list)



    def pushButton_3_pressed(self):
        _translate = QtCore.QCoreApplication.translate
        self.label_8.clear()
        name0,done0 = QtWidgets.QInputDialog.getText(MainWindow, 'Input Dialog', 'Enter the
amount you want to invest')
        if(done0):

self.label_8.setText(_translate("MainWindow",str(round(self.stockBuySell(self.price_list,name0
)[1],4))))


    def coin_selected(self):


        if(self.date_in_string_from=='' or self.date_in_string_to==''):
            msg = QtWidgets.QMessageBox(MainWindow)
            msg.setIcon(QtWidgets.QMessageBox.Information)
            msg.setStandardButtons(QtWidgets.QMessageBox.Ok)
            msg.setText("Select Valid Dates")
            msg.exec()
        else:
            if(self.converter(self.date_in_string_from)==self.converter(self.date_in_string_to)):
                    msg = QtWidgets.QMessageBox(MainWindow)
```

```
            msg.setIcon(QtWidgets.QMessageBox.Warning)
            msg.setStandardButtons(QtWidgets.QMessageBox.Ok)
            msg.setText("Enter Different dates")
            msg.exec()
    else:
        if(self.r1.isChecked()):
            self.url="https://in.investing.com/crypto/bitcoin/historical-data"
            _translate = QtCore.QCoreApplication.translate
            self.price_list=self.scrape_data(self.url)[0]
            self.date_list=self.scrape_data(self.url)[1]
            records=self.stockBuySell(self.price_list)
            self.label_7.setText(_translate("MainWindow",str(self.maxProfit(self.price_list))))
        if(self.r2.isChecked()):
            self.url="https://in.investing.com/crypto/dogecoin/historical-data"
            _translate = QtCore.QCoreApplication.translate
            self.price_list=self.scrape_data(self.url)[0]
            self.date_list=self.scrape_data(self.url)[1]
            records=self.stockBuySell(self.price_list)
            self.label_7.setText(_translate("MainWindow",str(self.maxProfit(self.price_list))))
        if(self.r3.isChecked()):
            self.url="https://in.investing.com/crypto/ethereum/historical-data"
            _translate = QtCore.QCoreApplication.translate
            self.price_list=self.scrape_data(self.url)[0]
            self.date_list=self.scrape_data(self.url)[1]
            records=self.stockBuySell(self.price_list)
            self.label_7.setText(_translate("MainWindow",str(self.maxProfit(self.price_list))))
        if(self.r4.isChecked()):
            self.url="https://in.investing.com/crypto/cardano/historical-data"
```

```
            _translate = QtCore.QCoreApplication.translate
            self.price_list=self.scrape_data(self.url)[0]
            self.date_list=self.scrape_data(self.url)[1]
            records=self.stockBuySell(self.price_list)
            self.label_7.setText(_translate("MainWindow",str(self.maxProfit(self.price_list))))
        if(self.r5.isChecked()):
            self.url="https://in.investing.com/crypto/tether/historical-data"
            _translate = QtCore.QCoreApplication.translate
            self.price_list=self.scrape_data(self.url)[0]
            self.date_list=self.scrape_data(self.url)[1]
            records=self.stockBuySell(self.price_list)
            self.label_7.setText(_translate("MainWindow",str(self.maxProfit(self.price_list))))


def scrape_data(self,url):
    list_data=[]
    date_list=[]
    r = Request(url, headers={"User-Agent": "Mozilla/5.0"})
    html_content=urlopen(r).read()
    soup = BeautifulSoup(html_content,"html.parser")

    table = soup.find_all('table')[0]
    rows = table.select('tbody > tr')
    header = table.select('thead > tr')
    count_start=0
    count_end=0
    for row in rows:
        data = [th.text.rstrip() for th in row.find_all('td')]
        for x in range(len(data)):
```

```
            data[x]=data[x].replace(',',")
        list_data.append(data)



    for lst_1 in list_data:
        count_start=count_start+1


        if lst_1[0]==self.converter(self.date_in_string_from):
            break;
    for lst_1 in list_data:
        count_end=count_end+1
        if lst_1[0]==self.converter(self.date_in_string_to):
            break;
    price_list=[]
    for i in range(count_start-1,count_end):
        price_list.append(float(list_data[i][2]))
        date_list.append(list_data[i][0])
    price_list=price_list[::-1]
    date_list=date_list[::-1]
    return price_list,date_list


def stockBuySell(self,price,input_1=1000):
    n=len(price)
    buy_pricelist=[]
    sell_pricelist=[]
    buylist=[]
    buy_sell=[]
    input=int(input_1)
```

```
    selllist=[]
    if (n == 1):
        return
    i = 0
    while (i < (n - 1)):
        while ((i < (n - 1)) and
               (price[i + 1] <= price[i])):
            i += 1
        if (i == n - 1):
            break
        buy = i
        i += 1
        while ((i < n) and (price[i] >= price[i - 1])):
            i += 1
        sell = i - 1
        buylist.append(self.date_list[buy])
        buy_pricelist.append(price[buy])
        selllist.append(self.date_list[sell])
        sell_pricelist.append(price[sell])
        buy_sell=list(zip(buylist,buy_pricelist,selllist,sell_pricelist))
        input=input*price[sell]/price[buy]
    self.tableWidget.setRowCount(0)
    for listofbuysell in buy_sell:
        self.addTableRow(self.tableWidget, listofbuysell)
    return buy_sell,input
def addTableRow(self, table, row_data):
    row = table.rowCount()
    table.setRowCount(row+1)
```

```
    col = 0
    for item in row_data:
        cell = QtWidgets.QTableWidgetItem(str(item))
        table.setItem(row, col, cell)
        col += 1


def maxProfit(self,prices):
    n = len(prices)
    cost = 0
    maxcost = 0
    if (n == 0):
        return 0
    min_price = prices[0]
    for i in range(n):
        min_price = min(min_price, prices[i])
        cost = prices[i] - min_price
        maxcost = round(max(maxcost, cost),4)
    return maxcost


def setupUi(self, MainWindow):
    MainWindow.setObjectName("MainWindow")
    MainWindow.resize(1128, 732)
    MainWindow.setFixedSize(1128, 732)

    self.price_list=[]
    self.date_list=[]

    self.date_in_string_from="
```

```
self.date_in_string_to="
self.centralwidget = QtWidgets.QWidget(MainWindow)
self.centralwidget.setObjectName("centralwidget")
self.tabWidget = QtWidgets.QTabWidget(self.centralwidget)
self.tabWidget.setGeometry(QtCore.QRect(10, 70, 1108, 611))
self.tabWidget.setTabShape(QtWidgets.QTabWidget.Triangular)
self.tabWidget.setIconSize(QtCore.QSize(24, 24))
self.tabWidget.setElideMode(QtCore.Qt.ElideNone)
self.tabWidget.setUsesScrollButtons(False)
self.tabWidget.setDocumentMode(False)
self.tabWidget.setTabsClosable(False)
self.tabWidget.setTabBarAutoHide(True)
self.tabWidget.setObjectName("tabWidget")

self.url = "https://in.investing.com/crypto/bitcoin/historical-data"

self.tab = QtWidgets.QWidget()
self.tab.setObjectName("tab")
self.tabWidget.addTab(self.tab, "")

self.tab_2 = QtWidgets.QWidget()
self.tab_2.setObjectName("tab_2")

self.tab_3 = QtWidgets.QWidget()
self.tab_3.setObjectName("tab_3")

self.tabWidget.setStyleSheet("QTabWidget::tab-bar{alignment:center;}");
```

```
self.label_2 = QtWidgets.QLabel(self.tab_2)
self.label_2.setGeometry(QtCore.QRect(230, 60, 101, 31))
self.label_2.setObjectName("fromdate")

self.label_3 = QtWidgets.QLabel(self.tab_2)
self.label_3.setGeometry(QtCore.QRect(740, 60, 101, 31))
self.label_3.setObjectName("todate")

self.label_4 = QtWidgets.QLabel(self.tab_2)
self.label_4.setGeometry(QtCore.QRect(230, 90, 95, 31))
self.label_4.setObjectName("label")
self.label_4.setStyleSheet("border: 1px solid black;")

self.tabWidget.addTab(self.tab_2, "")

self.pushButton_1 = QtWidgets.QPushButton(self.tab_2)
self.pushButton_1.setGeometry(QtCore.QRect(325, 89, 31, 33))
self.pushButton_1.setObjectName("pushButton")
self.pushButton_1.clicked.connect(self.popup)
self.pushButton_1.setIcon(QIcon('25_dropdown_menu_down_arrow-512.png'))

current_date = date.today()
l_date = QDate(current_date.year, int(current_date.month)-1, 19)
u_date = QDate(current_date.year, current_date.month, 19)

self.calendarWidget = QtWidgets.QCalendarWidget(self.tab_2)
self.calendarWidget.setDateRange(l_date, u_date)
self.calendarWidget.setGeometry(QtCore.QRect(230, 125, 350, 250))
```

```
self.calendarWidget.setObjectName("calendarWidget")
self.calendarWidget.setVisible(False)
self.calendarWidget.clicked.connect(self.calendar)


self.label_5 = QtWidgets.QLabel(self.tab_2)
self.label_5.setGeometry(QtCore.QRect(740, 90, 95, 31))
self.label_5.setObjectName("label_5")
self.label_5.setStyleSheet("border: 1px solid black;")

self.label_6 = QtWidgets.QLabel(self.tab_2)
self.label_6.setGeometry(QtCore.QRect(360, 50, 375, 130))
self.label_6.setObjectName("label_6")
self.movie_1 = QMovie("gif_bitcoin.gif")
self.label_6.setMovie(self.movie_1)
self.movie_1.start()
self.label_6.setAlignment(Qt.AlignCenter)

self.label_7 = QtWidgets.QLabel(self.tab_2)
self.label_7.setGeometry(QtCore.QRect(770, 300, 95, 31))
self.label_7.setObjectName("label_7")
self.label_7.setStyleSheet("border: 1px solid black;")

self.label_8 = QtWidgets.QLabel(self.tab_2)
self.label_8.setGeometry(QtCore.QRect(770, 450, 150, 31))
self.label_8.setObjectName("label_8")
self.label_8.setStyleSheet("border: 1px solid black;")
```

```
self.profit = QtWidgets.QLabel(self.tab_2)
self.profit.setGeometry(QtCore.QRect(770, 250, 95, 31))
self.profit.setObjectName("profit")

self.profit_in = QtWidgets.QLabel(self.tab_2)
self.profit_in.setGeometry(QtCore.QRect(770, 400, 250, 31))
self.profit_in.setObjectName("profit_in")

self.pushButton_3 = QtWidgets.QPushButton(self.tab_2)
self.pushButton_3.setGeometry(QtCore.QRect(770, 350, 95, 31))
self.pushButton_3.setObjectName("pushButton")
self.pushButton_3.clicked.connect(self.pushButton_3_pressed)

self.pushButton_2 = QtWidgets.QPushButton(self.tab_2)
self.pushButton_2.setGeometry(QtCore.QRect(835, 89, 31, 33))
self.pushButton_2.setObjectName("pushButton")
self.pushButton_2.clicked.connect(self.popup_1)
self.pushButton_2.setIcon(QIcon('25_dropdown_menu_down_arrow-512.png'))

self.label = QtWidgets.QLabel(self.centralwidget)
self.label.setGeometry(QtCore.QRect(260, 20, 591, 41))
self.label.setAlignment(QtCore.Qt.AlignCenter)
self.label.setWordWrap(False)
self.label.setObjectName("label")
self.movie = QMovie("ezgif.com-gif-maker.gif")
self.label.setMovie(self.movie)
self.movie.start()
```

```
self.graphWidget = pg.PlotWidget(self.tab_3)
self.graphWidget.setStyleSheet("background: gray")
self.graphWidget.setGeometry(QtCore.QRect(20, 15, 1050, 540))
self.graphWidget.setCursor(QtGui.QCursor(QtCore.Qt.PointingHandCursor))

self.line=self.graphWidget.plot([0],[0])

self.tabWidget.addTab(self.tab_3, "")

self.tableWidget = QtWidgets.QTableWidget(self.tab_2)
self.tableWidget.setGeometry(QtCore.QRect(12, 250, 600, 300))
self.tableWidget.setColumnCount(4)
columns = ['Buy Date','Buy Price', 'Sell Date','Sell Price']
self.tableWidget.setHorizontalHeaderLabels(columns)

self.tableWidget_1 = QtWidgets.QTableWidget(self.tab)
self.tableWidget_1.setGeometry(QtCore.QRect(10, 15, 550, 550))
self.tableWidget_1.setColumnCount(7)



self.r1 = QtWidgets.QRadioButton(self.tab_2)
self.r1.setGeometry(630, 240, 131, 41)
self.r1.clicked.connect(self.coin_selected)

self.r2 = QtWidgets.QRadioButton(self.tab_2)
self.r2.setGeometry(630, 305, 131, 41)
self.r2.clicked.connect(self.coin_selected)
```

```
self.r3 = QtWidgets.QRadioButton(self.tab_2)
self.r3.setGeometry(630, 375, 131, 41)
self.r3.clicked.connect(self.coin_selected)

self.r4 = QtWidgets.QRadioButton(self.tab_2)
self.r4.setGeometry(630, 445, 131, 41)
self.r4.clicked.connect(self.coin_selected)

self.r5 = QtWidgets.QRadioButton(self.tab_2)
self.r5.setGeometry(630, 520, 131, 41)
self.r5.clicked.connect(self.coin_selected)

self.r6 = QtWidgets.QRadioButton(self.tab)
self.r6.setGeometry(570, 15, 131, 41)
self.r6.setChecked(True)

self.r7 = QtWidgets.QRadioButton(self.tab)
self.r7.setGeometry(701, 15, 131, 41)

self.r8 = QtWidgets.QRadioButton(self.tab)
self.r8.setGeometry(832, 15, 131, 41)

self.r9 = QtWidgets.QRadioButton(self.tab)
self.r9.setGeometry(963, 15, 131, 41)

self.r10 = QtWidgets.QRadioButton(self.tab)
self.r10.setGeometry(766, 70, 131, 41)
```

```
self.r6.clicked.connect(lambda:self.scrape_data_tab_1())
self.r7.clicked.connect(lambda:self.scrape_data_tab_1())
self.r8.clicked.connect(lambda:self.scrape_data_tab_1())
self.r9.clicked.connect(lambda:self.scrape_data_tab_1())
self.r10.clicked.connect(lambda:self.scrape_data_tab_1())

self.label_9 = QtWidgets.QLabel(self.tab)
self.label_9.setGeometry(QtCore.QRect(570, 195, 130, 105))
bit = QPixmap('bitcoin (1).png')
self.label_9.setPixmap(bit)
self.label_9.setToolTip('Bit Coin')
self.label_9.setAlignment(Qt.AlignCenter)
self.label_9.setObjectName("label_9")

self.label_10 = QtWidgets.QLabel(self.tab)
self.label_10.setGeometry(QtCore.QRect(750, 195, 130, 105))
bit_1 = QPixmap('ethereum (1).jpg')
self.label_10.setPixmap(bit_1)
self.label_10.setToolTip('ethereum Coin')
self.label_10.setAlignment(Qt.AlignCenter)
self.label_10.setObjectName("label_10")

self.label_11 = QtWidgets.QLabel(self.tab)
self.label_11.setGeometry(QtCore.QRect(930, 195, 130, 105))
bit_2 = QPixmap('cardano (1).jpg')
self.label_11.setPixmap(bit_2)
self.label_11.setToolTip('cardano Coin')
self.label_11.setAlignment(Qt.AlignCenter)
```

```
self.label_11.setObjectName("label_11")

self.label_12 = QtWidgets.QLabel(self.tab)
self.label_12.setGeometry(QtCore.QRect(655, 370, 130, 105))
bit_3 = QPixmap('tether (1).jpg')
self.label_12.setToolTip('tether Coin')
self.label_12.setPixmap(bit_3)
self.label_12.setAlignment(Qt.AlignCenter)
self.label_12.setObjectName("label_12")

self.label_13 = QtWidgets.QLabel(self.tab)
self.label_13.setGeometry(QtCore.QRect(830, 370, 130, 105))
bit_4 = QPixmap('dogecoin (1).jpg')
self.label_13.setToolTip('Doge Coin')
self.label_13.setPixmap(bit_4)
self.label_13.setAlignment(Qt.AlignCenter)
self.label_13.setObjectName("label_13")

self.label_14 = QtWidgets.QLabel(self.tab)
self.label_14.setGeometry(QtCore.QRect(570, 310, 130, 30))
self.label_14.setObjectName("label_14")
self.label_14.setAlignment(Qt.AlignCenter)
self.label_14.setStyleSheet("border: 1px solid black;")

self.label_15 = QtWidgets.QLabel(self.tab)
self.label_15.setGeometry(QtCore.QRect(750, 310, 130, 30))
self.label_15.setObjectName("label_15")
self.label_15.setAlignment(Qt.AlignCenter)
```

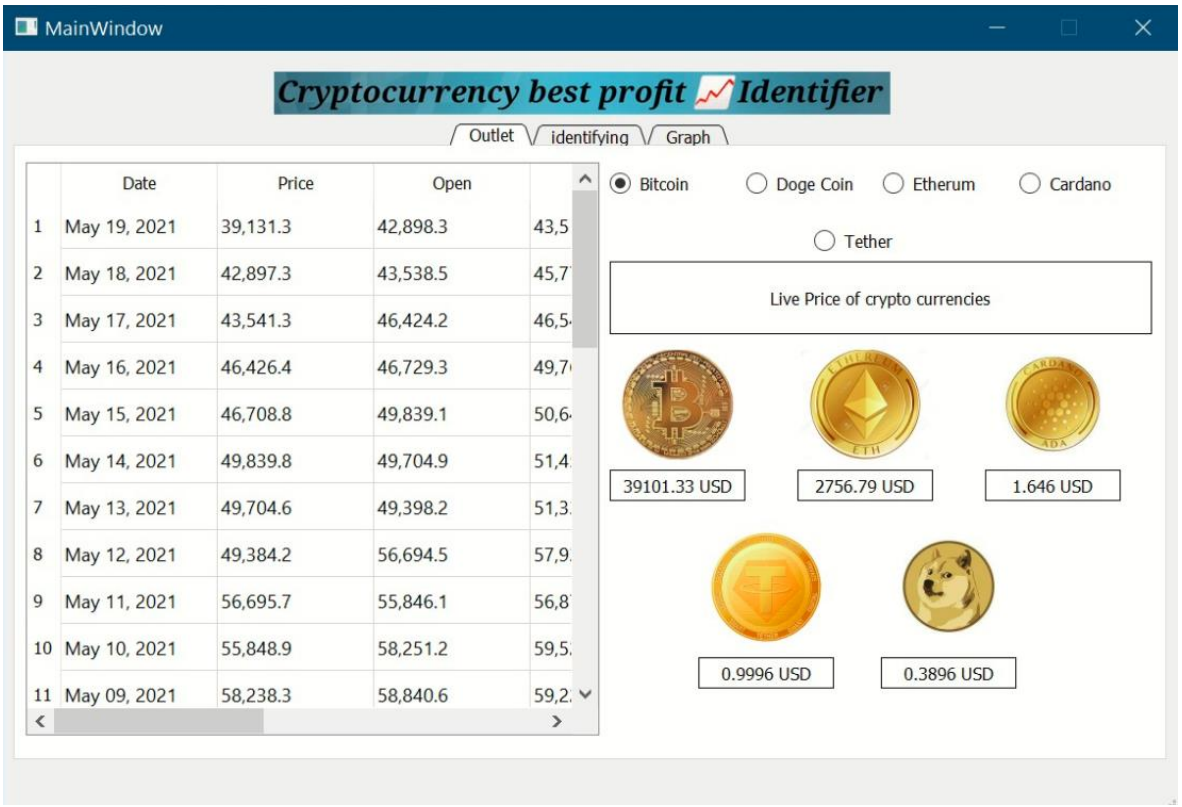
```
self.label_15.setStyleSheet("border: 1px solid black;")

self.label_16 = QtWidgets.QLabel(self.tab)
self.label_16.setGeometry(QtCore.QRect(930, 310, 130, 30))
self.label_16.setObjectName("label_16")
self.label_16.setAlignment(Qt.AlignCenter)
self.label_16.setStyleSheet("border: 1px solid black;")

self.label_17 = QtWidgets.QLabel(self.tab)
self.label_17.setGeometry(QtCore.QRect(655,490, 130, 30))
self.label_17.setAlignment(Qt.AlignCenter)
self.label_17.setObjectName("label_17")
self.label_17.setStyleSheet("border: 1px solid black;")

self.label_18 = QtWidgets.QLabel(self.tab)
self.label_18.setAlignment(Qt.AlignCenter)
self.label_18.setGeometry(QtCore.QRect(830, 490, 130, 30))
self.label_18.setObjectName("label_18")
self.label_18.setStyleSheet("border: 1px solid black;")

self.label_19 = QtWidgets.QLabel(self.tab)
self.label_19.setAlignment(Qt.AlignCenter)
self.label_19.setGeometry(QtCore.QRect(570, 110, 520, 70))
self.label_19.setObjectName("label_18")
self.label_19.setStyleSheet("border: 1px solid black;")

self.label_14.setText(str(cryptocompare.get_price('BTC', currency='USD')['BTC']['USD'])+'
'+'USD')
```

```
        self.label_15.setText(str(cryptocompare.get_price('ETH', currency='USD')['ETH']['USD'])+'
'+'USD')

        self.label_16.setText(str(cryptocompare.get_price('ADA',
currency='USD')['ADA']['USD'])+' '+'USD')

        self.label_17.setText(str(cryptocompare.get_price('USDT',
currency='USD')['USDT']['USD'])+' '+'USD')

        self.label_18.setText(str(cryptocompare.get_price('DOGE',
currency='USD')['DOGE']['USD'])+' '+'USD')


        self.calendarWidget_1 = QtWidgets.QCalendarWidget(self.tab_2)

        self.calendarWidget_1.setDateRange(l_date, u_date)

        self.calendarWidget_1.setGeometry(QtCore.QRect(740, 125, 350, 250))

        self.calendarWidget_1.setObjectName("calendarWidget")

        self.calendarWidget_1.setVisible(False)

        self.calendarWidget_1.clicked.connect(self.calendar_1)


#       backg = QPixmap('title_back.png')


#       self.picture_1 = QtWidgets.QLabel(self.tab)

#       self.picture_1.lower()

#       self.picture_1.setPixmap(backg)

#       self.picture_1.setGeometry(0,0, 1108, 611)


#       self.picture_2 = QtWidgets.QLabel(self.tab_2)

#       self.picture_2.lower()

#       self.picture_2.setPixmap(backg)

#       self.picture_2.setGeometry(0,0, 1108, 611)


#       self.picture_3 = QtWidgets.QLabel(self.tab_3)
```

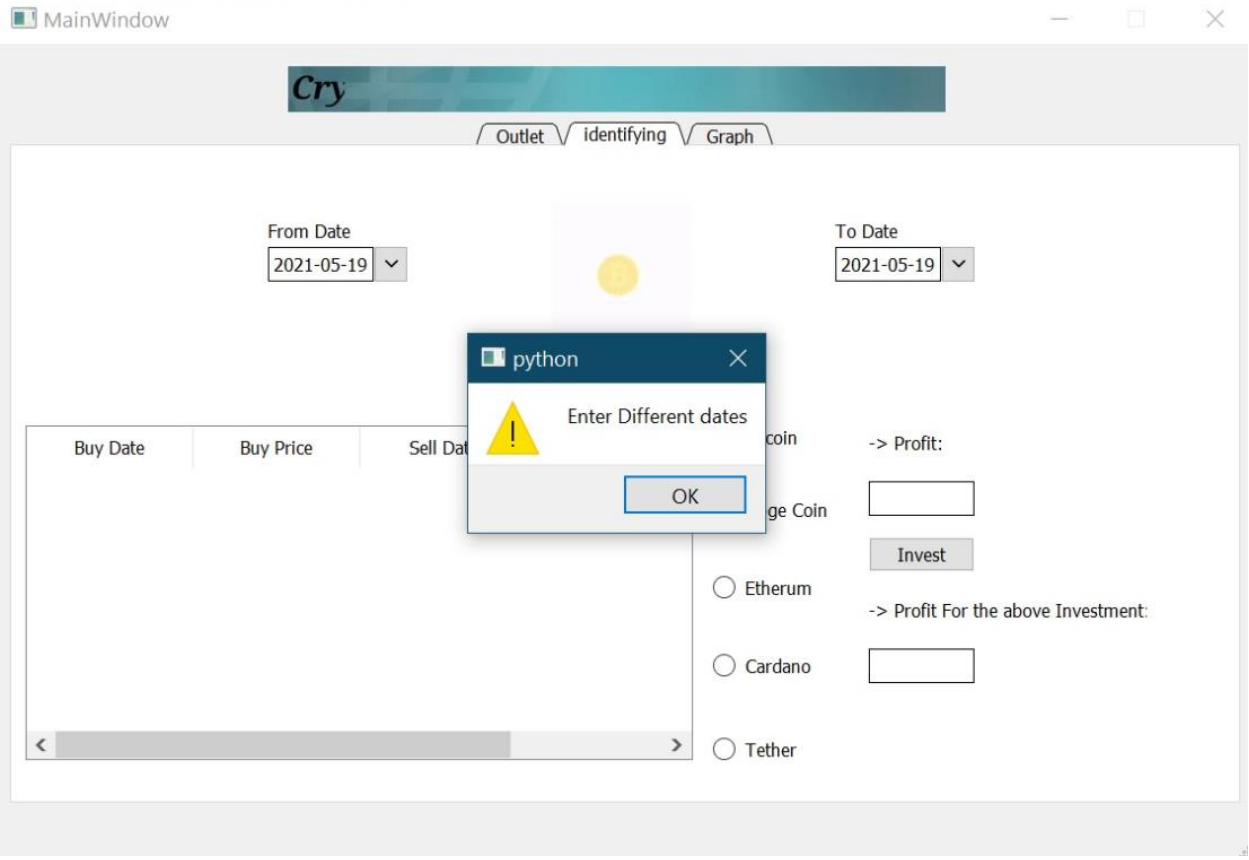
```
#         self.picture_3.lower()
#         self.picture_3.setPixmap(backg)
#         self.picture_3.setGeometry(0,0, 1108, 611)

#         self.picture_4 = QtWidgets.QLabel(self.centralwidget)
#         self.picture_4.lower()
#         self.picture_4.setPixmap(backg)
#         self.picture_4.setGeometry(0,0, 1108, 611)

        MainWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 1128, 31))
        self.menubar.setObjectName("menubar")
        MainWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainWindow)
        self.statusbar.setObjectName("statusbar")
        MainWindow.setStatusBar(self.statusbar)

        self.retranslateUi(MainWindow)
        self.tabWidget.setCurrentIndex(0)
        self.scrape_data_tab_1()
        QtCore.QMetaObject.connectSlotsByName(MainWindow)
    def retranslateUi(self, MainWindow):
        _translate = QtCore.QCoreApplication.translate
        MainWindow.setWindowTitle(_translate("MainWindow", "MainWindow"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), _translate("MainWindow",
"Outlet"))
        self.label_2.setText(_translate("MainWindow", "From Date"))
```

```
        self.label_3.setText(_translate("MainWindow", "To Date"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_2), _translate("MainWindow", "identifying"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab_3), _translate("MainWindow", "Graph"))
        self.label_4.setText(_translate("MainWindow", "Select Date"))
        self.label_5.setText(_translate("MainWindow", "Select Date"))
        self.r1.setText(_translate("MainWindow", "Bitcoin"))
        self.r2.setText(_translate("MainWindow", "Doge Coin"))
        self.r3.setText(_translate("MainWindow", "Etherum"))
        self.r4.setText(_translate("MainWindow", "Cardano"))
        self.r5.setText(_translate("MainWindow", "Tether"))
        self.r6.setText(_translate("MainWindow", "Bitcoin"))
        self.r7.setText(_translate("MainWindow", "Doge Coin"))
        self.r8.setText(_translate("MainWindow", "Etherum"))
        self.r9.setText(_translate("MainWindow", "Cardano"))
        self.r10.setText(_translate("MainWindow", "Tether"))
        self.label_19.setText(_translate("MainWindow", "Live Price of crypto currencies"))
        self.pushButton_3.setText(_translate("MainWindow", "Invest"))
        self.profit_in.setText(_translate("MainWindow", "-> Profit For the above Investment:"))
        self.profit.setText(_translate("MainWindow", "-> Profit:"))
if _name_ == "_main_":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    sys.exit(app.exec_())
```

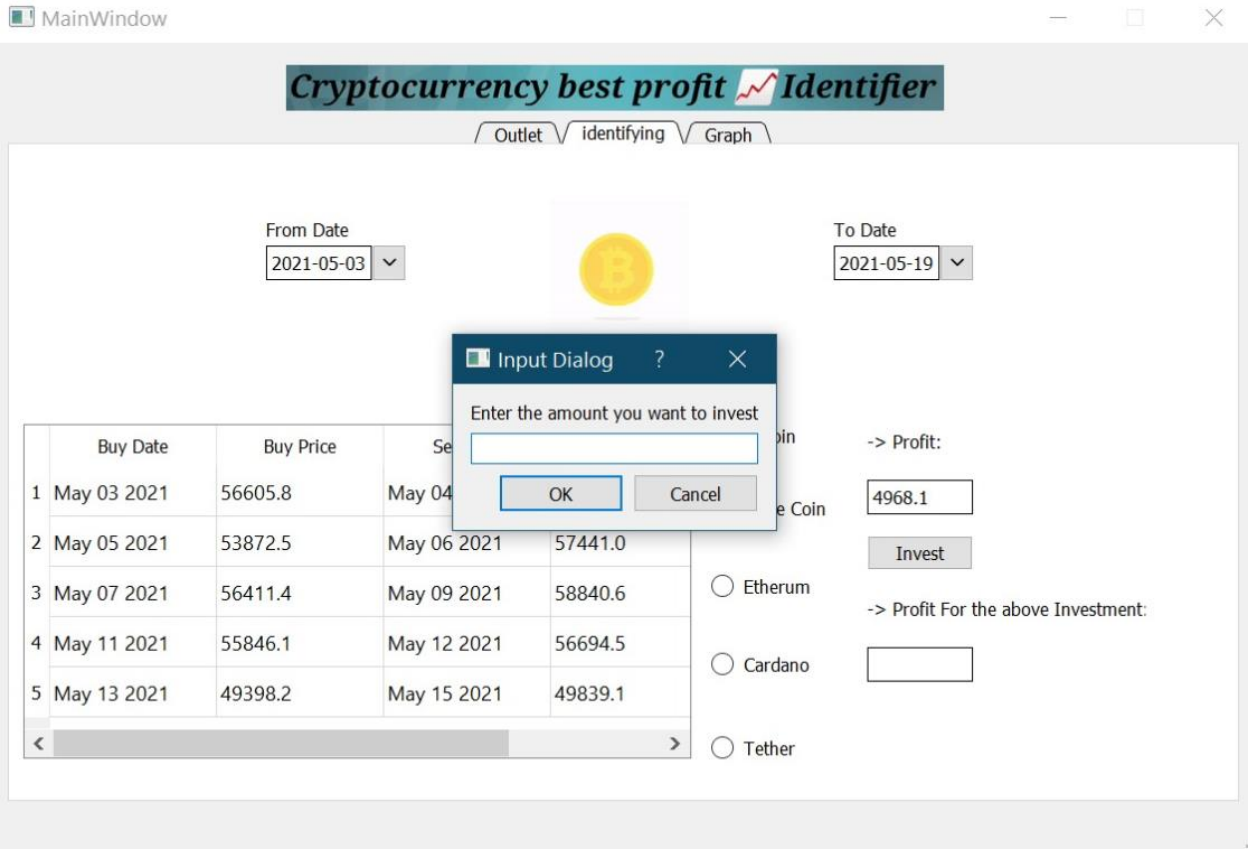
RESULTS:



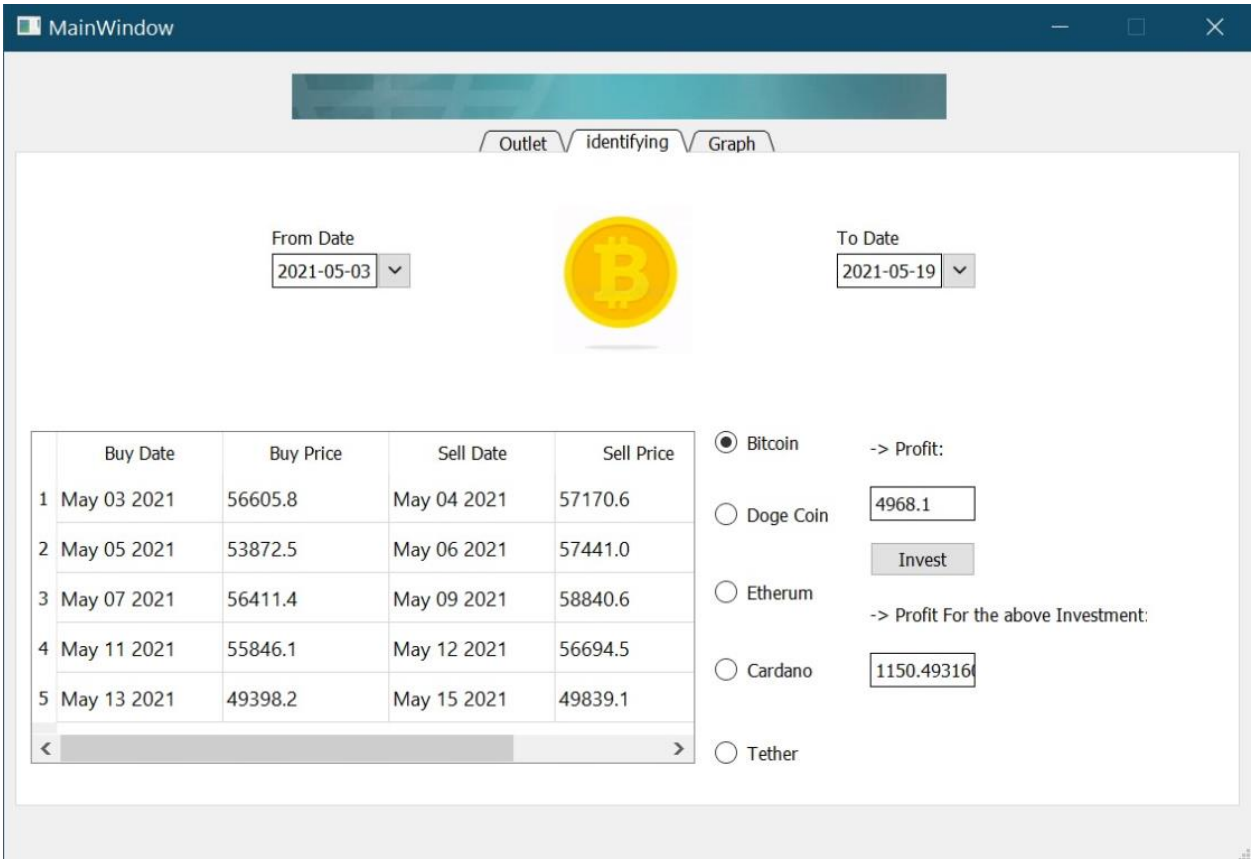
This image describes the basic interface of our application which shows live prices and the data of the selected Cryptocurrency



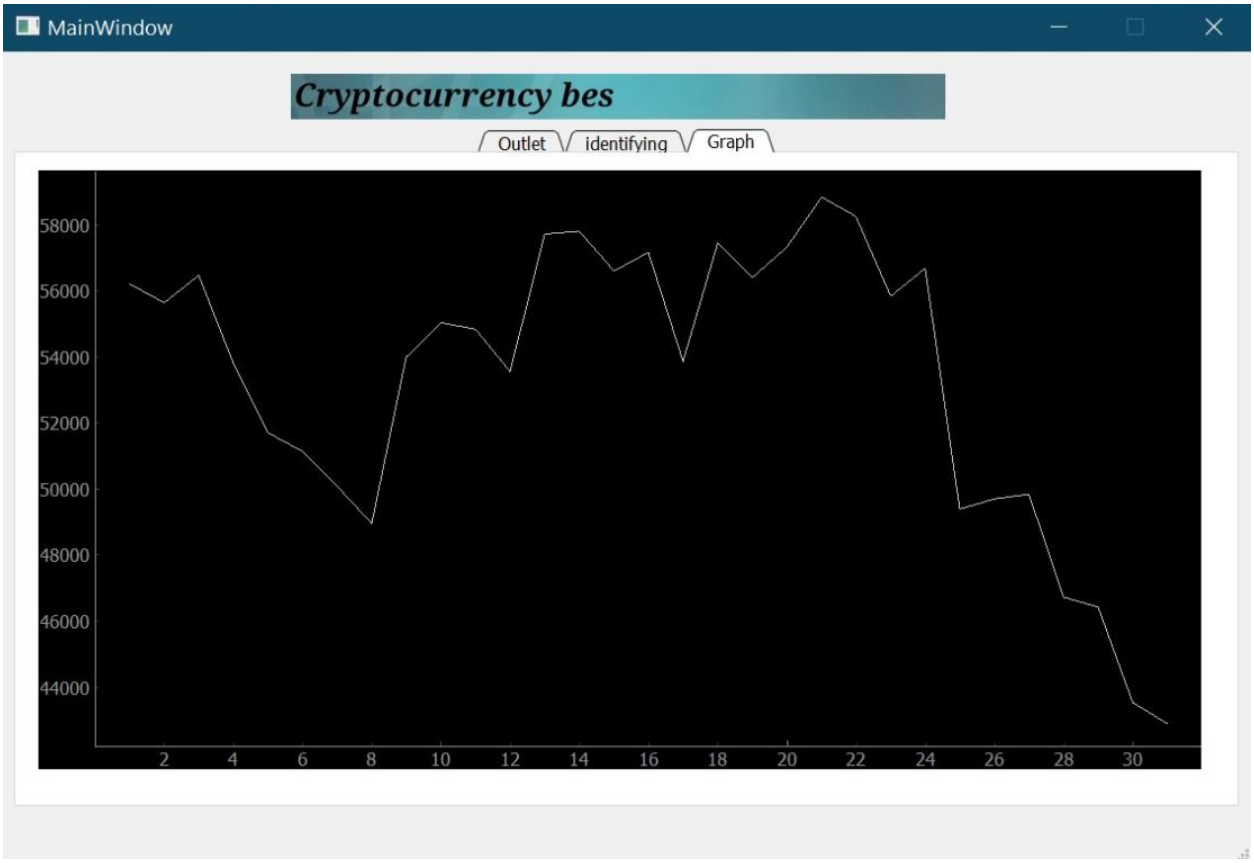
This image shows an warning example when both dates are given same where it shows that we have entered same dates please change it



This image shows the maximum profit obtained in the selected cryptocurrency and also shows how we can find the profit obtained by giving the value invested on it



This image shows all the profits by the cryptocurrency and the profit for the invested amount by you



This Image shows the graph of the cryptocurrencies day by day from last month

ADVANTAGES:

- By working with many buy and sell operations due to its recurring property. Multiple buy and sell operations take place, and the best profits are returned.
- Divide and Conquer algorithm's solutions are always optimal.
- With cryptocurrency:
    - Elimination of Banking Fees
    - Low Transaction Fees for International Payments
    - Being accessible from anywhere and from any device that have an internet connection

DISADVANTAGES:

- The worst-case time complexity of the program is more Θ(n^2*log(n)).
- Space complexity of the function is more Θ(log(n)). This is computed using the maximum memory utilization by the program. Each function call represents the function solving a subproblem.
- The program requires multiple passes over slices of the entire list.
- The minimum profit is zero. Losses are not accounted for and displayed by the program.
- With Cryptocurrency:
    - Scalability.
    - Cybersecurity issues.
    - Price volatility and lack of inherent value.
    - Regulations.

CONCLUSION:

The emergence of cryptocurrency has sparked a debate for the future. Cryptocurrency management is very important for the current financial scenario. Cryptocurrency is the best way to carry money as it prevents theft and can be used anywhere and anytime. Value of cryptocurrency keeps on changing and that is why sometimes it becomes difficult to manage the finances, but cryptocurrency is safe and much more secure than the traditional way of keeping money. Cryptocurrency can be made more efficient with methods of managing the changing values of cryptocurrencies. Therefore, By studying and investing in the market more carefully, we can invest the money for cryptocurrency in better ways.