

**Data Structure Course Projects Report**

Class：19G212

Student ID：20211002548

Name：Shuang Liang

Supervisor：Guilin Li

Date：2022.07.07

[1. Discrete Event Simulation: Customs Checkpoint Simulation System 4](#_Toc32113)

[1.1 Course project topics and requirements 4](#_Toc15475)

[【Problem Description】 4](#_Toc24696)

[【Basic Requirements】 4](#_Toc26364)

[【Extended Requirements】 4](#_Toc31400)

[1.2 Requirement Analysis 5](#_Toc3373)

[1.2.1 Input 5](#_Toc25455)

[1.2.2 Output 5](#_Toc5098)

[1.2.3 Function 5](#_Toc4771)

[1.3 Design 5](#_Toc10282)

[1.3.1 Design Idea 5](#_Toc11253)

[1.3.1.1 Data Structure Dsesign 5](#_Toc10017)

[1.3.1.2 Algorithm Design 6](#_Toc26344)

[1.3.2 Design representation 7](#_Toc1251)

[1.3.3 Detailed Design 7](#_Toc32624)

[1.3.3.1 Interface Design 7](#_Toc3325)

[1.3.3.2 Core Algorithm Design 8](#_Toc15533)

[1.4 Debugging Analysis 8](#_Toc29897)

[1.4.1 Problems encountered and Solutions 9](#_Toc18013)

[1.4.1.1 Problems encountered 9](#_Toc13719)

[1.4.1.2 Solutions 9](#_Toc1784)

[1.4.2 Time and Spatial Complexity Analysis 9](#_Toc5337)

[1.4.3 Improved Algorithm 9](#_Toc1941)

[1.5 User 's Manual 10](#_Toc13046)

[1.5.1 Enter Start and End Dates 10](#_Toc23558)

[1.5.2 Enter other parameters 10](#_Toc10447)

[1.6 Test Data and Test Results 11](#_Toc1923)

[1.6.1 Test Case 11](#_Toc23018)

[1.6.2 Output Results 11](#_Toc9314)

[1.7 Source Program List 13](#_Toc6247)

[1.7.1 Time controller Calendar.py 13](#_Toc14526)

[1.7.2 Circular Queue Queue.py 18](#_Toc15990)

[1.7.3 GUI and Main T1\_main.py 19](#_Toc9542)

[2. Calculate the truth value of propositional calculus formulas 22](#_Toc17502)

[2.1 Calculate the truth value of propositional calculus formulas 22](#_Toc31280)

[【Problem Description】 22](#_Toc13918)

[【Basic Requirements】 22](#_Toc25620)

[【Extension Requirements】 22](#_Toc673)

[2.2 Requirement Analysis 23](#_Toc1184)

[2.2.1 Input 23](#_Toc10139)

[2.3 Design 23](#_Toc15857)

[2.3.1 Design Idea 23](#_Toc20110)

[2.3.1.1 Data Structure Design 23](#_Toc28042)

[2.3.1.2 Algorithm Design 24](#_Toc31502)

[2.3.2 Design Representation 24](#_Toc15433)

[2.3.3 Detailed Design 24](#_Toc32692)

[2.3.3.1 Test the Legitimacy of Propositional Formula 24](#_Toc4813)

[2.3.3.2 Infix Expression Converted to Postfix Expression 25](#_Toc10598)

[2.3.3.3 Convert Postfix Expression to Expression Tree 25](#_Toc32287)

[2.3.3.4 Calculate the Expression Tree 26](#_Toc24026)

[2.3.3.5 Find all non repeating propositional arguments in propositional formulas 27](#_Toc28324)

[2.3.3.6 Enumerate all truth values of propositional arguments 27](#_Toc3907)

[2.3.3.7 Visualization of the Binary Tree 27](#_Toc19361)

[2.3.3.8 Expansion requirements - calculate the value of arithmetic expression 28](#_Toc13891)

[2.4 Debugging analysis 29](#_Toc26583)

[2.4.1 Problems encountered and Solutions 29](#_Toc19611)

[2.4.2 Time and Space Complexity 29](#_Toc2048)

[2.5 User 's Manual 29](#_Toc22065)

[2.5.1 Input 29](#_Toc29235)

[2.5.2 Output 29](#_Toc13150)

[2.6 Test Data and Test Results 29](#_Toc30078)

[Test Case 1 29](#_Toc3752)

[Test Case 2 30](#_Toc4919)

[Test Case 3 30](#_Toc2389)

[2.7 Source Program List 31](#_Toc32713)

[2.7.1 Implementation of Simple Stack Stack.py 31](#_Toc9588)

[2.7.2 Implementation of Binary Tree BTree.py 31](#_Toc7952)

[2.7.3 Binary Tree Visualization BtreeVisualization.py 32](#_Toc8528)

[2.7.4 main T2\_main.py 33](#_Toc135)

[2.7.5 Expansion requirements main T2\_ExtendedRequirements.py 35](#_Toc22382)

# 

# Discrete Event Simulation: Customs Checkpoint Simulation System

## Course project topics and requirements

### 【Problem Description】

Consider a customs checkpoint responsible for checking transit vehicles and develop a concrete simulation system. For this system, the following basic considerations are assumed:

(1) The duty of the customs is to check the passing vehicles, here only one direction of traffic inspection is simulated.

(2) Assuming that vehicles arrive at a certain rate, there is a certain randomness, and a vehicle arrives every a to b minutes.

(3) The customs has k inspection channels, and it takes c to d minutes to inspect a vehicle.

(4) Arriving vehicles wait in line on a dedicated line. Once an inspection channel is free, the first vehicle in the queue will enter the channel for inspection. If a vehicle arrives with an empty lane and there is no waiting vehicle, it immediately enters the lane and starts checking.

(5) The desired data include the average waiting time of vehicles and the average time passing through checkpoints.

### 【Basic Requirements】

The system needs to simulate the inspection process at a customs checkpoint and output a series of events, as well as the average queuing time and average transit time for vehicles.

### 【Extended Requirements】

Please modify the customs checkpoint simulation system to use a management strategy of one waiting queue per inspection channel. Do some simulations of this new strategy and compare the simulation results with the strategy of sharing the waiting queue.

## Requirement Analysis

### 1.2.1 Input

Users inputs the precise year, month, day, hour, minute and second from the drop-down date selector of the graphical interactive interface, simulates the start time and end time, and inputs the values of a, b, c, d and k.

### 1.2.2 Output

The simulated events at the current time and the corresponding time are output in chronological order, and the average waiting time of the vehicle and the average time of passing the checkpoint are output at the end.

### 1.2.3 Function

Simulate customs inspection, simulate the arrival, inspection and departure of inspection vehicles according to the input time and the values of parameters a, b, c, d and k, and calculate the average waiting time of vehicles and the average time of passing through the inspection station.

## 1.3 Design

### 1.3.1 Design Idea

#### 1.3.1.1 Data Structure Dsesign

##### 1.3.1.1.1 Storage Structure Design

Python built in library Datetime.datetime,Used to store and calculate accurate time.

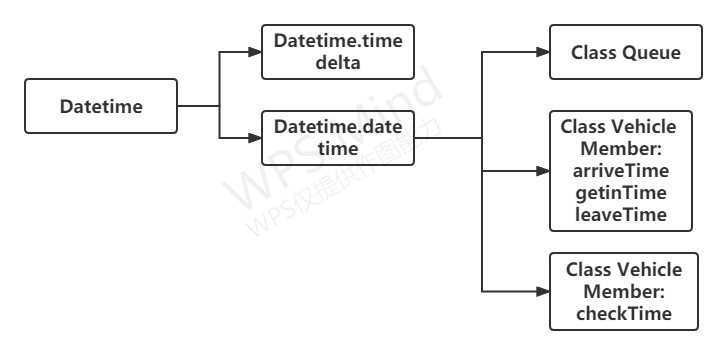
Python built in library Datetime.timedelta,Used to store and calculate precise time intervals.

Custom class Queue,Realize circular queue, which is used to simulate the queue waiting for customs inspection.

Custom class Vehicle，Simulate vehicles waiting for inspection，data members are：number(Vehicle number,int type),arriveTime,checkTime（Time interval required for inspection）,getinTime（Time to start inspection）,leaveTime（Departure time）.

|  |  |  |
| --- | --- | --- |
| Class Vehicle | | |
| Data member | Member type | Significance |
| number | int | Vehicle number |
| arriveTime | Datetime.datetime | arrival time |
| checkTime | Datetime.timedelta | Time interval required for inspection |
| getinTime | Datetime.datetime | Time to start inspection |
| leaveTime | Datetime.datetime | Departure time |

##### 1.3.1.1.2 Logical Structure Design



#### 1.3.1.2 Algorithm Design

(1) Preprocess the random number of each vehicle Arrivetime (datetime.timedelta type), vehicle Checktime (datetime.timedelta type) and its number, and store the vehicle in the list.

(2) List of vehicles [i] Arrivetime (datetime.timedelta type) sums the prefixes to get the specific arrival time of each vehicle vehicles[i] Arrivetime (datetime.datetime type).

(3) Starting from the start time, traverse every integer minute until the end time.

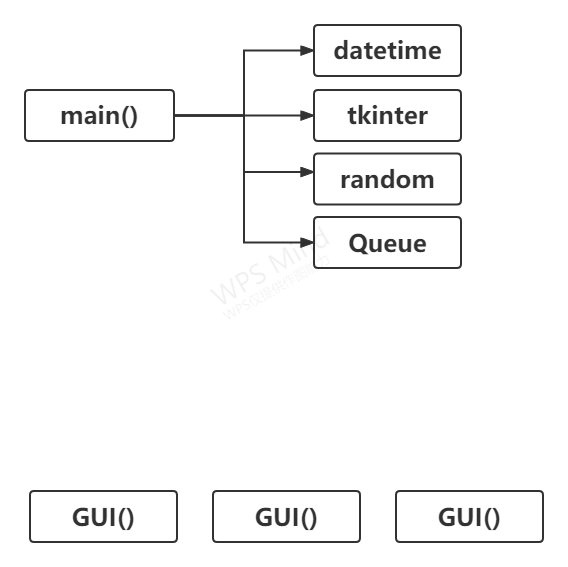
(4) While traversing the time, judge vehicles[i] Whether arrivetime is the current time. If so, vehicles[i] Join the waiting queue.

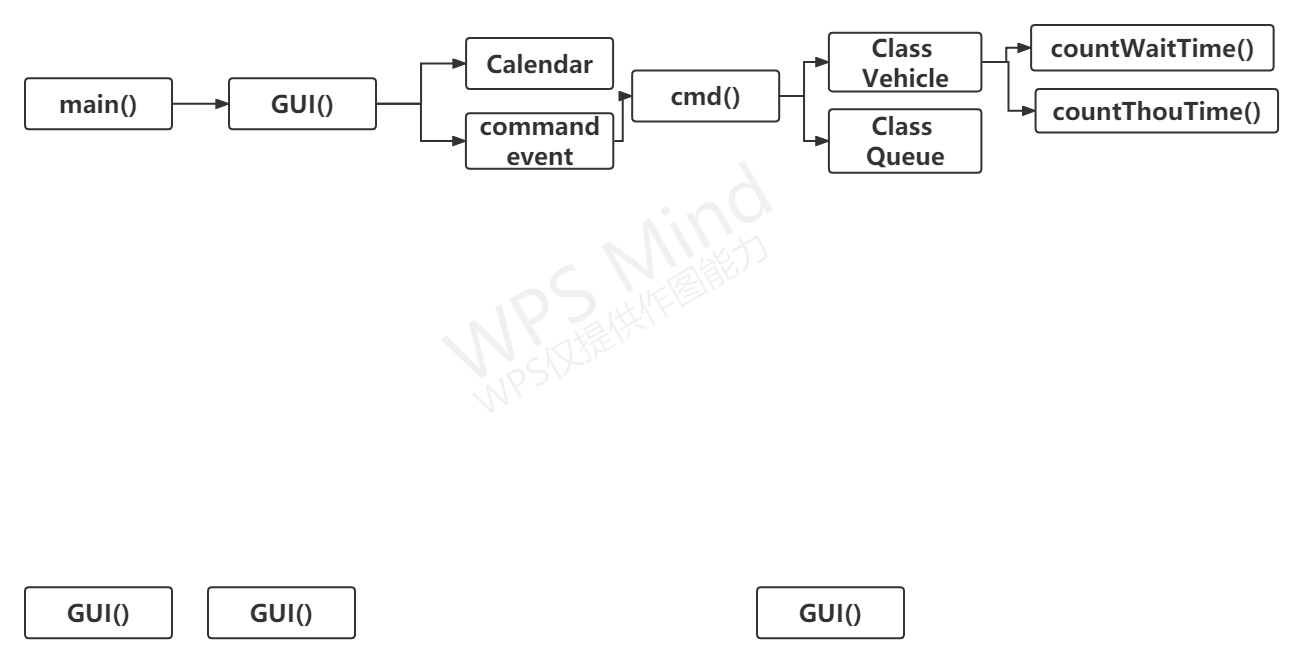
Judge whether the inspection queue is full. If it is not, it will wait for the header element of the queue to leave the queue, and record the start of inspection Time (vehicle.getintime).

Judge whether the inspection of the head element of the inspection queue has been completed. If it is completed, it will be out of the queue and the departure time (vehicle.leavetime) will be recorded.

（5）Count and output the average waiting time and average passing time of each vehicle

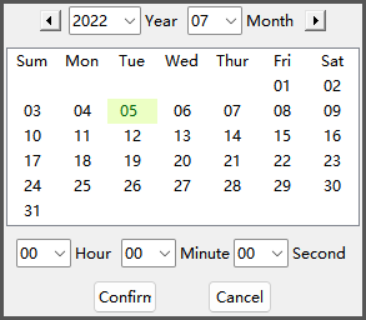
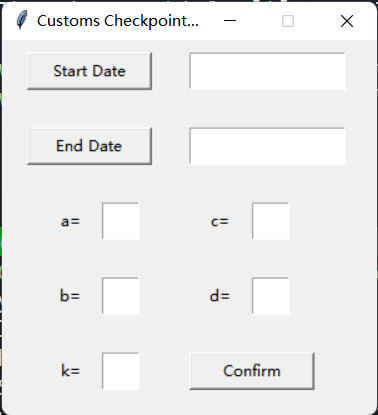
### 1.3.2 Design representation





### 1.3.3 Detailed Design

#### 1.3.3.1 Interface Design



#### 1.3.3.2 Core Algorithm Design

#End at end time

while (currentTime <= endTime):

#Judge whether there are vehicles arriving at this time

if (vehicles[i].arriveTime<=currentTime) then

#Vehicles enter the waiting queue

waitQueue.enqueue(vehicles[i])

#Record arrival time

Record currentTime as arriveTime

#Print arrival events

Print arrive event

#Judge whether the check queue is full

if (not checkQueue.full()) and (not waitQueue.empty())

#Vehicles leave the waiting queue

waitQueue.dequeue()

#The vehicle enters the inspection queue

checkQueue.enqueue()

#Record the time when the inspection started

Record current time as getinTime

#Judge whether the check queue is empty

if (not checkQueue.empty()) then

#Judge whether there are vehicles leaving at this time

If checkQueue.peek().getinTime and #Judge whether the check queue is empty

checkQueue.peek().checkTime>=currentTime then

#Vehicle leaves the inspection queue

checkQueue.dequeue()

#Record the departure time

Record currentTime as leaveTime

#Print vehicle departure event

Print leave event

#Simulate the next minute

currentTime+=1 min

## 1.4 Debugging Analysis

### 1.4.1 Problems encountered and Solutions

#### 1.4.1.1 Problems encountered

(1) The date is entered interactively in the graphical interface.

(2) The graphical interface is difficult to transfer data to the core algorithm.

(3) Convert the obtained time string into a datetime object.

#### 1.4.1.2 Solutions

(1) Customize the drop-down calendar space calendar class.

(2) The core algorithm is encapsulated into a function and triggered by button events.

(3) Use format controllers or regular expressions.

### 1.4.2 Time and Spatial Complexity Analysis

Time complexity：O(N)

Spatial complexity：O(N)

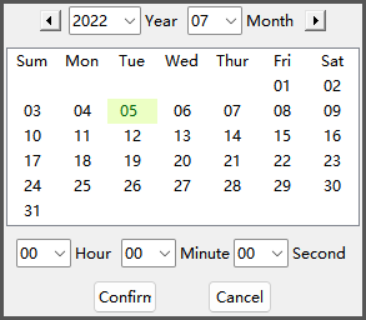
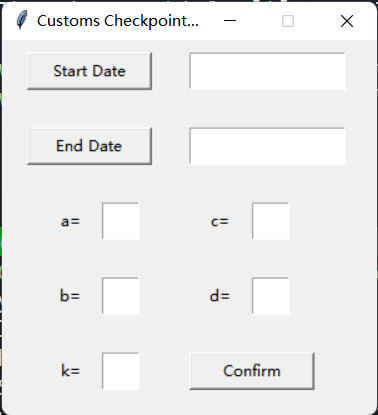
### 1.4.3 Improved Algorithm

Customize the customs inspection channel class, use the blocking queue, and so on.

## 1.5 User 's Manual

### 1.5.1 Enter Start and End Dates

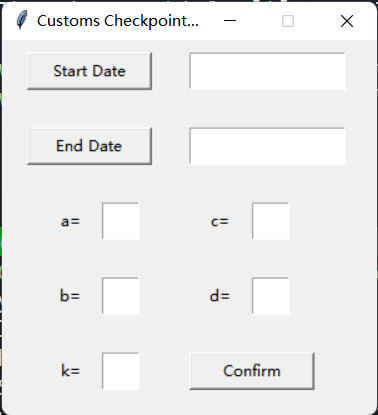
**2.Choose**



**1.Click**

**3.Confirm**

### 1.5.2 Enter other parameters



**2.Start simulation**

**1.Input**

## 1.6 Test Data and Test Results

### 1.6.1 Test Case

|  |  |  |  |
| --- | --- | --- | --- |
| Parameter | Value | Parameter | Value |
| startTime | 2022-07-05 00:00:00 | c | 10 |
| endTime | 2022-07-05 06:00:00 | d | 3 |
| a | 5 | k | 5 |
| b | 3 |  |  |

### 1.6.2 Output Results

2022-07-05 00:05:00

Event: Vehicle 1 arrive at the custom

2022-07-05 00:05:00

Event: Vehicle 0 leave the custom

2022-07-05 00:15:00

Event: Vehicle 2 arrive at the custom

2022-07-05 00:15:00

Event: Vehicle 1 leave the custom

2022-07-05 00:21:00

Event: Vehicle 3 arrive at the custom

2022-07-05 00:21:00

Event: Vehicle 2 leave the custom

2022-07-05 00:30:00

Event: Vehicle 4 arrive at the custom

2022-07-05 00:30:00

Event: Vehicle 3 leave the custom

2022-07-05 00:39:00

Event: Vehicle 5 arrive at the custom

2022-07-05 00:39:00

Event: Vehicle 4 leave the custom

2022-07-05 00:45:00

Event: Vehicle 6 arrive at the custom

2022-07-05 00:45:00

Event: Vehicle 5 leave the custom

2022-07-05 00:51:00

Event: Vehicle 7 arrive at the custom

2022-07-05 00:51:00

Event: Vehicle 6 leave the custom

2022-07-05 00:59:00

Event: Vehicle 8 arrive at the custom

2022-07-05 00:59:00

Event: Vehicle 7 leave the custom

2022-07-05 01:06:00

Event: Vehicle 9 arrive at the custom

2022-07-05 01:06:00

Event: Vehicle 8 leave the custom

2022-07-05 01:16:00

Event: Vehicle 10 arrive at the custom

2022-07-05 01:16:00

Event: Vehicle 9 leave the custom

2022-07-05 01:23:00

Event: Vehicle 11 arrive at the custom

2022-07-05 01:23:00

Event: Vehicle 10 leave the custom

2022-07-05 01:31:00

Event: Vehicle 12 arrive at the custom

2022-07-05 01:31:00

Event: Vehicle 11 leave the custom

2022-07-05 01:41:00

Event: Vehicle 13 arrive at the custom

2022-07-05 01:41:00

Event: Vehicle 12 leave the custom

2022-07-05 01:50:00

Event: Vehicle 14 arrive at the custom

2022-07-05 01:50:00

Event: Vehicle 13 leave the custom

2022-07-05 01:56:00

Event: Vehicle 15 arrive at the custom

2022-07-05 01:56:00

Event: Vehicle 14 leave the custom

2022-07-05 02:01:00

Event: Vehicle 16 arrive at the custom

2022-07-05 02:01:00

Event: Vehicle 15 leave the custom

2022-07-05 02:10:00

Event: Vehicle 17 arrive at the custom

2022-07-05 02:10:00

Event: Vehicle 16 leave the custom

2022-07-05 02:18:00

Event: Vehicle 18 arrive at the custom

2022-07-05 02:18:00

Event: Vehicle 17 leave the custom

2022-07-05 02:26:00

Event: Vehicle 19 arrive at the custom

2022-07-05 02:26:00

Event: Vehicle 18 leave the custom

2022-07-05 02:31:00

Event: Vehicle 20 arrive at the custom

2022-07-05 02:31:00

Event: Vehicle 19 leave the custom

2022-07-05 02:37:00

Event: Vehicle 21 arrive at the custom

2022-07-05 02:37:00

Event: Vehicle 20 leave the custom

2022-07-05 02:42:00

Event: Vehicle 22 arrive at the custom

2022-07-05 02:42:00

Event: Vehicle 21 leave the custom

2022-07-05 02:50:00

Event: Vehicle 23 arrive at the custom

2022-07-05 02:50:00

Event: Vehicle 22 leave the custom

2022-07-05 02:56:00

Event: Vehicle 24 arrive at the custom

2022-07-05 02:56:00

Event: Vehicle 23 leave the custom

2022-07-05 03:06:00

Event: Vehicle 25 arrive at the custom

2022-07-05 03:06:00

Event: Vehicle 24 leave the custom

2022-07-05 03:15:00

Event: Vehicle 26 arrive at the custom

2022-07-05 03:15:00

Event: Vehicle 25 leave the custom

2022-07-05 03:23:00

Event: Vehicle 27 arrive at the custom

2022-07-05 03:23:00

Event: Vehicle 26 leave the custom

2022-07-05 03:28:00

Event: Vehicle 28 arrive at the custom

2022-07-05 03:28:00

Event: Vehicle 27 leave the custom

2022-07-05 03:34:00

Event: Vehicle 29 arrive at the custom

2022-07-05 03:34:00

Event: Vehicle 28 leave the custom

2022-07-05 03:43:00

Event: Vehicle 30 arrive at the custom

2022-07-05 03:43:00

Event: Vehicle 29 leave the custom

2022-07-05 03:50:00

Event: Vehicle 31 arrive at the custom

2022-07-05 03:50:00

Event: Vehicle 30 leave the custom

2022-07-05 03:56:00

Event: Vehicle 32 arrive at the custom

2022-07-05 03:56:00

Event: Vehicle 31 leave the custom

2022-07-05 04:04:00

Event: Vehicle 33 arrive at the custom

2022-07-05 04:04:00

Event: Vehicle 32 leave the custom

2022-07-05 04:10:00

Event: Vehicle 34 arrive at the custom

2022-07-05 04:10:00

Event: Vehicle 33 leave the custom

2022-07-05 04:20:00

Event: Vehicle 35 arrive at the custom

2022-07-05 04:20:00

Event: Vehicle 34 leave the custom

2022-07-05 04:29:00

Event: Vehicle 36 arrive at the custom

2022-07-05 04:29:00

Event: Vehicle 35 leave the custom

2022-07-05 04:37:00

Event: Vehicle 37 arrive at the custom

2022-07-05 04:37:00

Event: Vehicle 36 leave the custom

2022-07-05 04:44:00

Event: Vehicle 38 arrive at the custom

2022-07-05 04:44:00

Event: Vehicle 37 leave the custom

2022-07-05 04:54:00

Event: Vehicle 39 arrive at the custom

2022-07-05 04:54:00

Event: Vehicle 38 leave the custom

2022-07-05 05:01:00

Event: Vehicle 40 arrive at the custom

2022-07-05 05:01:00

Event: Vehicle 39 leave the custom

2022-07-05 05:09:00

Event: Vehicle 41 arrive at the custom

2022-07-05 05:09:00

Event: Vehicle 40 leave the custom

2022-07-05 05:19:00

Event: Vehicle 42 arrive at the custom

2022-07-05 05:19:00

Event: Vehicle 41 leave the custom

2022-07-05 05:29:00

Event: Vehicle 43 arrive at the custom

2022-07-05 05:29:00

Event: Vehicle 42 leave the custom

2022-07-05 05:36:00

Event: Vehicle 44 arrive at the custom

2022-07-05 05:36:00

Event: Vehicle 43 leave the custom

2022-07-05 05:43:00

Event: Vehicle 45 arrive at the custom

2022-07-05 05:43:00

Event: Vehicle 44 leave the custom

2022-07-05 05:52:00

Event: Vehicle 46 arrive at the custom

2022-07-05 05:52:00

Event: Vehicle 45 leave the custom

2022-07-05 06:00:00

Event: Vehicle 47 arrive at the custom

2022-07-05 06:00:00

Event: Vehicle 46 leave the custom

Average Wait Time: 0:10:00

Average Through Time: 0:07:00

## 1.7 Source Program List

### 1.7.1 Time controller Calendar.py

import datetime

import calendar

import tkinter as tk

from tkinter import StringVar, ttk

import tkinter.font as tkFont

datetime = calendar.datetime.datetime

timedelta = calendar.datetime.timedelta

class Calendar:

def \_\_init\_\_(self, point = None):

# self.root = root

self.master = tk.Toplevel()

self.master.withdraw()

self.master.attributes('-topmost' ,True)

fwday = calendar.SUNDAY

year = datetime.now().year

month = datetime.now().month

locale = None

sel\_bg = '#ecffc4'

sel\_fg = '#05640e'

self.\_date = datetime(year, month, 1) #The first day of each month

self.\_selection = None #Set as unselected date

self.G\_Frame = ttk.Frame(self.master)

self.\_cal = self.\_\_get\_calendar(locale, fwday)

self.\_\_setup\_styles()

#Create a custom style self.\_\_place\_widgets()

# pack/grid Widget

self.\_\_config\_calendar()

# Adjust calendar columns and installation Tags

# Configure the canvas and the correct binding to select the date.

self.\_\_setup\_selection(sel\_bg, sel\_fg)

#Storage item ID for later insertion. self.\_items = [self.\_calendar.insert('', 'end', values='') for \_ in range(6)]

#Insert date in the current empty calendar

self.\_update()

self.G\_Frame.pack(expand = 1, fill = 'both')

self.master.overrideredirect(1)

self.master.update\_idletasks()

width, height = self.master.winfo\_reqwidth(), self.master.winfo\_reqheight()

self.height=height

if point:

x, y = point[0], point[1]

else:

x, y = (self.master.winfo\_screenwidth() - width)/2, (self.master.winfo\_screenheight() - height)/2

self.master.geometry('%dx%d+%d+%d' % (width, height, x, y)) #Window position centered

self.master.after(300, self.\_main\_judge)

self.master.deiconify()

self.master.focus\_set()

# self.master.mainloop()

self.master.wait\_window()

#Wait should be used here\_ Window suspends the window. If you use mainloop, it may cause many errors in the main program

def \_\_get\_calendar(self, locale, fwday):

if locale is None:

return calendar.TextCalendar(fwday)

else:

return calendar.LocaleTextCalendar(fwday, locale)

def \_\_setitem\_\_(self, item, value):

if item in ('year', 'month'):

raise AttributeError("attribute '%s' is not writeable" % item)

elif item == 'selectbackground':

self.\_canvas['background'] = value

elif item == 'selectforeground':

self.\_canvas.itemconfigure(self.\_canvas.text, item=value)

else:

self.G\_Frame.\_\_setitem\_\_(self, item, value)

def \_\_getitem\_\_(self, item):

if item in ('year', 'month'):

return getattr(self.\_date, item)

elif item == 'selectbackground':

return self.\_canvas['background']

elif item == 'selectforeground':

return self.\_canvas.itemcget(self.\_canvas.text, 'fill')

else:

r = ttk.tclobjs\_to\_py({item: ttk.Frame.\_\_getitem\_\_(self, item)})

return r[item]

def \_\_setup\_styles(self):

#Custom TTK style

style = ttk.Style(self.master)

arrow\_layout = lambda dir: (

[('Button.focus', {'children': [('Button.%sarrow' % dir, None)]})]

)

style.layout('L.TButton', arrow\_layout('left'))

style.layout('R.TButton', arrow\_layout('right'))

def \_\_place\_widgets(self):

#Header frame and its widgets

Input\_judgment\_num = self.master.register(self.Input\_judgment)

#You need to wrap the function. It's necessary hframe = ttk.Frame(self.G\_Frame)

gframe = ttk.Frame(self.G\_Frame)

bframe = ttk.Frame(self.G\_Frame)

xframe = ttk.Frame(self.G\_Frame)

hframe.pack(in\_=self.G\_Frame, side='top', pady=5, anchor='center')

gframe.pack(in\_=self.G\_Frame, fill=tk.X, pady=5)

bframe.pack(in\_=self.G\_Frame, pady=5)

xframe.pack(in\_=self.G\_Frame, side='bottom', pady=5)

lbtn = ttk.Button(hframe, style='L.TButton', command=self.\_prev\_month)

lbtn.grid(in\_=hframe, column=0, row=0, padx=5)

rbtn = ttk.Button(hframe, style='R.TButton', command=self.\_next\_month)

rbtn.grid(in\_=hframe, column=5, row=0, padx=5)

#year drop-down box

self.CB\_year = ttk.Combobox(hframe, width = 5, values = [str(year) for year in range(datetime.now().year, datetime.now().year-11,-1)], validate = 'key', validatecommand = (Input\_judgment\_num, '%P'))

self.CB\_year.current(0)

self.CB\_year.grid(in\_=hframe, column=1, row=0)

self.CB\_year.bind('<KeyPress>', lambda event:self.\_update(event, True))

self.CB\_year.bind("<<ComboboxSelected>>", self.\_update)

tk.Label(hframe, text = 'Year', justify = 'left').grid(in\_=hframe, column=2, row=0, padx=(0,5))

#month drop-down box

self.CB\_month = ttk.Combobox(hframe, width = 3, values = ['%02d' % month for month in range(1,13)], state = 'readonly')

self.CB\_month.current(datetime.now().month - 1)

self.CB\_month.grid(in\_=hframe, column=3, row=0)

self.CB\_month.bind("<<ComboboxSelected>>", self.\_update)

tk.Label(hframe, text = 'Month', justify = 'left').grid(in\_=hframe, column=4, row=0)

# Calendar part

self.\_calendar = ttk.Treeview(gframe, show='', selectmode='none', height=7)

self.\_calendar.pack(expand=1, fill='both', side='bottom', padx=5)

#Time drop-down box

self.CB\_hour = ttk.Combobox(bframe, width = 3, values = ['%02d' % hour for hour in range(0,24)], validate = 'key', validatecommand = (Input\_judgment\_num, '%P'))

self.CB\_hour.current(0)

self.CB\_hour.grid(in\_=bframe, column=0, row=0)

self.CB\_hour.bind('<KeyPress>', lambda event:self.\_update(event, True))

self.CB\_hour.bind("<<ComboboxSelected>>", self.\_update)

tk.Label(bframe, text = 'Hour').grid( in\_=bframe,column=1, row=0, padx=(0,5))

#minute drop-down box

self.CB\_mins = ttk.Combobox(bframe, width = 3, values = ['%02d' % mins for mins in range(0,60)], validate = 'key', validatecommand = (Input\_judgment\_num, '%P'))

self.CB\_mins.current(0)

self.CB\_mins.grid(in\_=bframe, column=2, row=0)

self.CB\_mins.bind('<KeyPress>', lambda event:self.\_update(event, True))

self.CB\_mins.bind("<<ComboboxSelected>>", self.\_update)

tk.Label(bframe, text = 'Minute').grid(column=3, row=0)

#second drop-down box

self.CB\_seconds = ttk.Combobox(bframe, width = 3, values = ['%02d' % secds for secds in range(0,60)], validate = 'key', validatecommand = (Input\_judgment\_num, '%P'))

self.CB\_seconds.current(0)

self.CB\_seconds.grid(in\_=bframe, column=4, row=0)

self.CB\_seconds.bind('<KeyPress>', lambda event:self.\_update(event, True))

self.CB\_seconds.bind("<<ComboboxSelected>>", self.\_update)

tk.Label(bframe, text = 'Second',justify = 'left').grid(column=5, row=0)

ttk.Button(xframe, text = "Confirm", width = 6, command = lambda: self.\_exit(True)).grid(row = 1, column = 0, sticky = 'ns', padx = 20)

ttk.Button(xframe, text = "Cancel", width = 6, command = self.\_exit).grid(row = 1, column = 4, sticky = 'ne', padx = 20)

tk.Frame(self.G\_Frame, bg = '#565656').place(x = 0, y = 0, relx = 0, rely = 0, relwidth = 1, relheigh = 2/200)

tk.Frame(self.G\_Frame, bg = '#565656').place(x = 0, y = 0, relx = 0, rely = 198/200, relwidth = 1, relheigh = 2/200)

tk.Frame(self.G\_Frame, bg = '#565656').place(x = 0, y = 0, relx = 0, rely = 0, relwidth = 2/200, relheigh = 1)

tk.Frame(self.G\_Frame, bg = '#565656').place(x = 0, y = 0, relx = 198/200, rely = 0, relwidth = 2/200, relheigh = 1)

def \_\_config\_calendar(self):

# cols = self.\_cal.formatweekheader(3).split()

cols = ['Sum','Mon','Tue','Wed','Thur','Fri','Sat']

self.\_calendar['columns'] = cols

self.\_calendar.tag\_configure('header', background='grey90')

self.\_calendar.insert('', 'end', values=cols, tag='header')

#Adjust its column width

font = tkFont.Font()

maxwidth = max(font.measure(col) for col in cols)

for col in cols:

self.\_calendar.column(col, width=maxwidth, minwidth=maxwidth,

anchor='center')

def \_\_setup\_selection(self, sel\_bg, sel\_fg):

def \_\_canvas\_forget(evt):

canvas.place\_forget()

self.\_selection = None

self.\_font = tkFont.Font()

self.\_canvas = canvas = tk.Canvas(self.\_calendar, background=sel\_bg, borderwidth=0, highlightthickness=0)

canvas.text = canvas.create\_text(0, 0, fill=sel\_fg, anchor='w')

canvas.bind('<Button-1>', \_\_canvas\_forget)

self.\_calendar.bind('<Configure>', \_\_canvas\_forget)

self.\_calendar.bind('<Button-1>', self.\_pressed)

def \_build\_calendar(self):

year, month = self.\_date.year, self.\_date.month

header = self.\_cal.formatmonthname(year, month, 0)

#Update the date displayed in the calendar

cal = self.\_cal.monthdayscalendar(year, month)

for indx, item in enumerate(self.\_items):

week = cal[indx] if indx < len(cal) else []

fmt\_week = [('%02d' % day) if day else '' for day in week]

self.\_calendar.item(item, values=fmt\_week)

def \_show\_select(self, text, bbox):

x, y, width, height = bbox

textw = self.\_font.measure(text)

canvas = self.\_canvas

canvas.configure(width = width, height = height)

canvas.coords(canvas.text, (width - textw)/2, height / 2 - 1)

canvas.itemconfigure(canvas.text, text=text)

canvas.place(in\_=self.\_calendar, x=x, y=y)

def \_pressed(self, evt = None, item = None, column = None, widget = None):

"" "click somewhere on the calendar." ""

if not item:

x, y, widget = evt.x, evt.y, evt.widget

item = widget.identify\_row(y)

column = widget.identify\_column(x)

if not column or not item in self.\_items:

#Click in the weekday row or just outside the column.

return

item\_values = widget.item(item)['values']

if not len(item\_values):

#The line of this month is empty.

return

text = item\_values[int(column[1]) - 1]

if not text:

return

bbox = widget.bbox(item, column)

if not bbox: #Calendar is not visible yet

self.master.after(20, lambda : self.\_pressed(item = item, column = column, widget = widget))

return

text = '%02d' % text #day

self.\_selection = (text, item, column)

self.\_show\_select(text, bbox)

def \_prev\_month(self):

"" "update the calendar to show the previous month." ""

self.\_canvas.place\_forget()

self.\_selection = None

self.\_date = self.\_date - timedelta(days=1)

self.\_date = datetime(self.\_date.year, self.\_date.month, 1)

self.CB\_year.set(self.\_date.year)

self.CB\_month.set(self.\_date.month)

self.\_update()

def \_next\_month(self):

"" "update the calendar to show the next month." ""

self.\_canvas.place\_forget()

self.\_selection = None

year, month = self.\_date.year, self.\_date.month

self.\_date = self.\_date + timedelta(

days=calendar.monthrange(year, month)[1] + 1)

self.\_date = datetime(self.\_date.year, self.\_date.month, 1)

self.CB\_year.set(self.\_date.year)

self.CB\_month.set(self.\_date.month)

self.\_update()

def \_update(self, event = None, key = None):

"" "refresh interface" ""

if key and event.keysym != 'Return': return

year = int(self.CB\_year.get())

month = int(self.CB\_month.get())

hour = int(self.CB\_hour.get())

mins = int(self.CB\_mins.get())

seconds = int(self.CB\_seconds.get())

if year == 0 or year > 9999: return

self.\_canvas.place\_forget()

self.\_date = datetime(year, month, 1,hour,mins,seconds)

self.\_build\_calendar()#Rebuild calendar

if year == datetime.now().year and month == datetime.now().month:

day = datetime.now().day

for \_item, day\_list in enumerate(self.\_cal.monthdayscalendar(year, month)):

if day in day\_list:

item = 'I00' + str(\_item + 2)

column = '#' + str(day\_list.index(day)+1)

self.master.after(100, lambda :self.\_pressed(item = item, column = column, widget = self.\_calendar))

def \_exit(self, confirm = False):

if not confirm: self.\_selection = None

self.master.destroy()

def \_main\_judge(self):

"" "determine whether the window is at the top level" ""

try:

if self.master.focus\_displayof() == None or 'toplevel' not in str(self.master.focus\_displayof()): self.\_exit()

else: self.master.after(10, self.\_main\_judge)

except:

self.master.after(10, self.\_main\_judge)

def selection(self):

"" "returns the date time that represents the currently selected date." ""

if not self.\_selection:

return None

year, month = self.\_date.year, self.\_date.month

hour=self.\_date.hour

mins = self.\_date.minute

seconds = self.\_date.second

return str(datetime(year, month, int(self.\_selection[0]),hour,mins,seconds))

def Input\_judgment(self, content):

"" "input judgment" ""

if content.isdigit() or content == "":

return True

else:

return False

### 1.7.2 Circular Queue Queue.py

class Queue:

def \_\_init\_\_(self, limit=10):

self.data = [None] \* limit

self.head = -1

self.tail = -1

def full(self):

if (self.head+1)%len(self.data)==self.tail:

return True

else:

return False

def enqueue(self, val): # O(1)

if self.empty():

self.head=0

self.tail=0

self.data[0]=val

else:

if (self.head+1)%len(self.data)==self.tail:

raise RuntimeError()

else:

self.head= (self.head + 1) % len(self.data)

self.data[self.head] = val

def peek(self):

return self.data[self.head]

def dequeue(self): # O(1)

if self.empty():

raise RuntimeError()

ret = self.data[self.tail]

self.data[self.tail] = None

self.tail = (self.tail + 1) % len(self.data)

if (self.head+1)%len(self.data)==self.tail:

self.tail=self.head=-1

return ret

def resize(self, newsize):

assert(len(self.data) < newsize)

newq=Queue(newsize)

for i in self:

newq.enqueue(i)

self.data=newq.data

self.head=newq.head

self.tail=newq.tail

def empty(self):

if self.head==self.tail==-1:

return True

return False

def \_\_bool\_\_(self):

return not self.empty()

def \_\_str\_\_(self):

if not(self):

return ''

return ', '.join(str(x) for x in self)

def \_\_repr\_\_(self):

return str(self)

def \_\_iter\_\_(self):

head=(self.head+len(self.data))%len(self.data)

tail=(self.tail+len(self.data))%len(self.data)

i=tail

while (i!=head):

yield self.data[i]

i=(i+1)%len(self.data)

else:

yield (self.data)

### 1.7.3 GUI and Main T1\_main.py

import tkinter as tk

from tkinter import StringVar, ttk

from Calendar import Calendar

from Queue import Queue

from datetime import datetime as dt

from datetime import timedelta as td

import random as rd

class Vehicle():

def \_\_init\_\_(self, number,arriveTime, checkTime):

self.number = number

self.arriveTime = arriveTime

self.checkTime = checkTime

self.getinTime = None

self.leaveTime = None

def countWaitTime(self):

return self.leaveTime-self.arriveTime-self.checkTime

def countThouTime(self):

return self.leaveTime-self.arriveTime

def GUI():

def getdate(type):

for date in [Calendar().selection()]:

if date:

if(type == 'start'): #If it is a start button, assign it to the start date start\_date.set(date)

elif(type == 'end'):

end\_date.set(date)

def cmd():

a=int(entry3.get())

b=int(entry4.get())

c=int(entry5.get())

d=int(entry6.get())

k=int(entry7.get())

start=dt.strptime(start\_date.get(), "%Y-%m-%d %H:%M:%S")

end=dt.strptime(end\_date.get(), "%Y-%m-%d %H:%M:%S")

currentTime=start

vehicles = [Vehicle(0,start+td(minutes=rd.randint(a, b)),

checkTime=td(minutes=rd.randint(c, d)))]

n = int((end-start).total\_seconds()//60//a)

for i in range(1, n):

vehicles.append(Vehicle(i,vehicles[i-1].arriveTime+td(minutes=rd.randint(a, b)), td(minutes=rd.randint(c, d))))

waitQueue = Queue(limit=n)

checkQueue = Queue(limit=k)

i=0

l=0

while (currentTime <= end):

if vehicles[i].arriveTime<=currentTime:

waitQueue.enqueue(vehicles[i])

vehicles[i].arriveTime=currentTime

i+=1

print(currentTime)

print("Event:","Vehicle",vehicles[i].number,"arrive at the custom")

if (not checkQueue.full()) and (not waitQueue.empty()):

tmp=waitQueue.dequeue()

checkQueue.enqueue(tmp)

tmp.getinTime = currentTime

if not checkQueue.empty():

if checkQueue.peek().getinTime + checkQueue.peek().checkTime>=currentTime:

tmp=checkQueue.dequeue()

vehicles[i].leaveTime=currentTime

l+=1

print(currentTime)

print("Event:","Vehicle",tmp.number,"leave the custom")

currentTime+=td(minutes=1)

avgeWaitTime=td()

avgeThouTime=td()

for j in range(l):

avgeWaitTime+=vehicles[i].countWaitTime()

avgeThouTime+=vehicles[i].countThouTime()

print("Average Wait Time:",-avgeWaitTime/l)

print("Average Through Time:",-avgeThouTime/l)

root = tk.Tk()

root.title('Customs Checkpoint Simulation System')

root.geometry('300x300')

root.resizable(False,False)

start\_date=tk.StringVar()

end\_date=tk.StringVar()

button1=tk.Button(root,

width=15,

text='Start Date',

command=lambda: getdate('start'))

button1.place(x=20,

y=10,

width=100,

height=30)

entry1=tk.Entry(root, textvariable=start\_date)

entry1.place(x=150,

y=10,

width=125,

height=30)

button2=tk.Button(root, width=15, text='End Date', command=lambda: getdate(

'end'))

button2.place(x=20,

y=70,

width=100,

height=30)

entry2=tk.Entry(root,textvariable=end\_date)

entry2.place(x=150,

y=70,

width=125,

height=30)

button3=tk.Button(root,text="Confirm",command = cmd)

entry3=tk.Entry(root,width=5)

entry4=tk.Entry(root,width=5)

entry5=tk.Entry(root,width=5)

entry6=tk.Entry(root,width=5)

entry7=tk.Entry(root,width=5)

label1=tk.Label(root,text='a=')

label2=tk.Label(root,text='b=')

label3=tk.Label(root,text='c=')

label4=tk.Label(root,text='d=')

label5=tk.Label(root,text='k=')

entry3.place(x=80,

y=130,

width=30,

height=30)

label1.place(x=40,

y=130,

width=30,

height=30)

entry4.place(x=80,

y=190,

width=30,

height=30)

label2.place(x=40,

y=190,

width=30,

height=30)

entry5.place(x=200,

y=130,

width=30,

height=30)

label3.place(x=160,

y=130,

width=30,

height=30)

entry6.place(x=200,

y=190,

width=30,

height=30)

label4.place(x=160,

y=190,

width=30,

height=30)

entry7.place(x=80,

y=250,

width=30,

height=30)

label5.place(x=40,

y=250,

width=30,

height=30)

button3.place(x=150,

y=250,

width=100,

height=30)

root.mainloop()

GUI()

# 2. Calculate the truth value of propositional calculus formulas

## 2.1 Calculate the truth value of propositional calculus formulas

【Problem Description】

The propositional calculus formula refers to a formula composed of logical variables (its value is TRUE or FALSE) and logical operators ∧ (AND), ∨ (OR) and (NOT) according to certain rules (operations such as implication can be used ∧ , ∨ and to represent). The sequence of formula operations is ! , ∧, ∨, and parentheses () can change the priority. Given a propositional calculus formula and the value of each variable, it is required to design a program to calculate the truth value of the formula.

### 【Basic Requirements】

(1) Use a binary tree to calculate the truth value of the formula.

Firstly, use the stack to change the infix form of the formula into the suffix form. Secondly, according to the suffix form, construct the corresponding binary tree from the leaf node. Finally, traverse the binary tree in post-order, and find the value of each subtree. That is, each time a node is reached, the value of its subtree has been calculated. When the root node is reached, the truth value of the formula is obtained.

(2) Design a variety of propositional calculus formulas in different forms, and check the validity of each propositional calculus formula.

(3) The identifier of the logical argument is not limited to a single letter, but can be an alphanumeric string of any length. Logical arguments can appear multiple times in a formula.

(4) Print the construction process of the binary tree, print suffix form of the formula and the post-order traversal sequence of the binary tree.

(5) Enter the value of each variable, calculate and display the truth value of the formula, print the evaluation process of the binary tree.

(6) Display the truth table of the formula.

### 【Extension Requirements】

Please replace logical operators with arithmetic operators and use binary tree to calculate the arithmetic expression.

## 2.2 Requirement Analysis

### 2.2.1 Input

Enter the propositional calculus formula.

**2.2.2 Function**

(1) Test the legitimacy of propositional calculus formula.

(2) Convert propositional calculus formula into suffix expression.

(3) Convert the suffix expression into an expression tree and print its construction process.

(4) Using the expression tree to calculate the truth value of the propositional calculus formula, and print its truth table.

**2.2.3 Output**

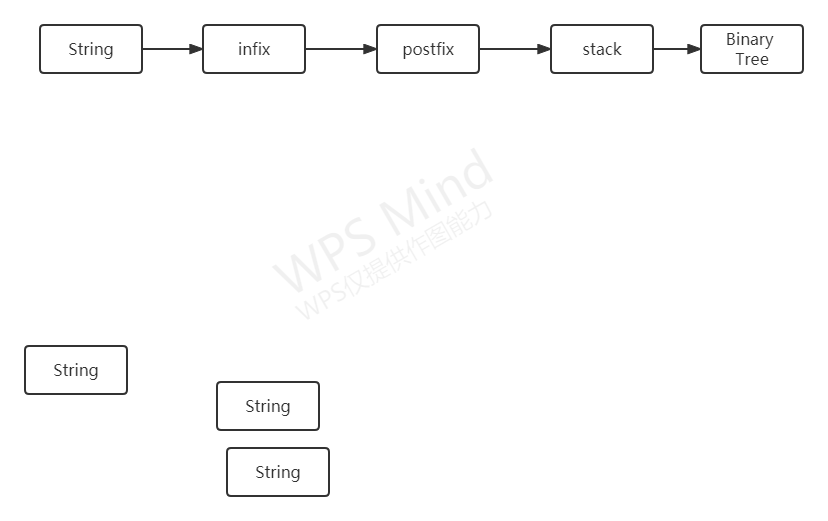
Output the suffix expression of propositional calculus formula, the construction process of expression tree, and the truth table of propositional calculus formula.

## 2.3 Design

### 2.3.1 Design Idea

#### 2.3.1.1 Data Structure Design

##### 2.3.1.1.1 Logical Structure Design



##### 2.3.1.1.2 Storage Structure Design

1. Use the dictionary to store the priority of operators, which is convenient for looking up the dictionary when using.

2. Use a binary tree to store the expression tree.

3. Use the stack to build suffix expressions and expression trees.

#### 2.3.1.2 Algorithm Design

(1) Stack is used to check the validity of propositional formula.

(2) Linear search is used to distinguish propositional arguments from operators.

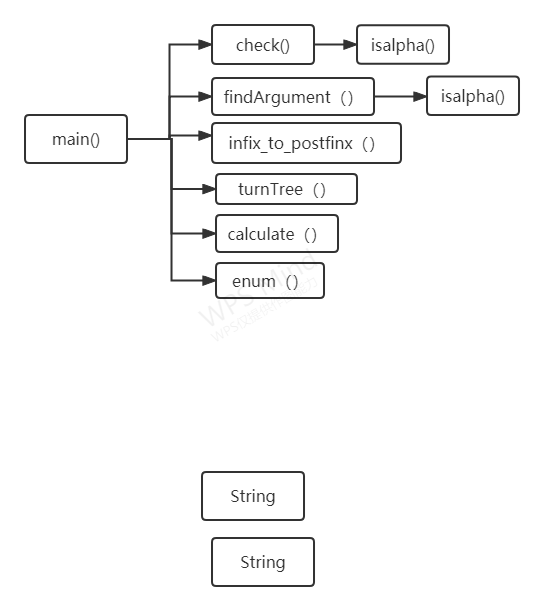
(3) Use stack to convert infix expression into suffix expression.

(4) Use stack to transform suffix expression into expression tree.

(5) Using binary tree to recursively calculate the truth value of a proposition.

All interpretations of propositional formulas are generated by binary transformation.

### 2.3.2 Design Representation



### 2.3.3 Detailed Design

#### 2.3.3.1 Test the Legitimacy of Propositional Formula

#Distinguishing propositional arguments from truth values

def isalpha(word):

If ('!','∧','∨','(',')',' ') is in word

return False

else

return True

#Check legitimacy

def check(expr):

s = Stack()

#Check the legitimacy of the beginning and end of the propositional formula

if expr begins with ['∧','∨'] or ends with ['∧','∨','!']

return False

else:

#Illegal enumeration operator

if expr[i] in ['∧','∨'] and expr[i+1] in ['∧','∨',')']

return False

elif expr[i] in ['!','('] and expr[i+1] in ['∧','∨',')']:

return False

elif expr[i]=='(':expr[i+1] in ['∧','∨']:

return False

elif expr[i]==')' and expr[i+1]=='!':

return False

#Check the validity of brackets

if expr[i] == '(':

s.push(expr[i])

elif expr[i] == ')':

if s.empty():

return False

elif s.pop() != '(':

return False

return s.empty()

#### 2.3.3.2 Infix Expression Converted to Postfix Expression

(1) Create a stack and a linear table.

(2) Traversing infix, the array elements are propositional arguments and directly enter the linear table; The array elements are operators and (), advanced stack.

a. When entering the stack, the priority of the top element is obtained first. If the priority of this element is greater than or equal to the top element, the top element pops up and is stored in the linear table.

b. If the element is "(", it is directly put on the stack.

c. If the stack is empty or the element at the top of the stack is "(", it will be directly put on the stack.

d. If the element is ")" the element in the stack pops up and is stored in the linear table until the element at the top of the stack is "(".

(4) Pop up the remaining elements in the stack in turn and put them into the linear table.

def infix\_to\_postfix(expr):

ops = Stack()

postfix = []

toks = expr.split()

#Handling parentheses

for c in toks:

if c == '(':

ops.push(c)。

elif c == ')':

while ops.peek() != '(':

postfix.append(ops.pop())

ops.pop()

#Dealing with propositional arguments

elif isalpha(c):

postfix.append(c)

else:

#Push by operator priority

while bool(ops) and prec[ops.peek()] >= prec[c]:

postfix.append(ops.pop())

ops.push(c)

#Pop the remaining operators out of the stack

while bool(ops):

postfix.append(ops.pop())

return postfix

#### 2.3.3.3 Convert Postfix Expression to Expression Tree

(1) Create a stack and prepare each symbol of the suffix expression to be pushed into the stack.

(2) If the symbol is a number, the number represents a single node tree and is pushed onto the stack.

(3) If the symbol is a binary operator, pop out two elements from the stack at a time, the first is the right subtree, the second is the left subtree, and then push the newly generated tree onto the stack.

(4) If the symbol is a unary operator, pop out one element as a right subtree from the stack at a time, and then push the newly generated tree onto the stack.

(5) Finally, there is only one tree node element left in the stack. Pop it out and it is the root node of the expression tree.

def turnTree(postfix):

s=Stack()

for i in postfix:

if isalpha(i):

s.push(BTree.Node(i))

elif i=='!':

a=s.pop()

tmp=BTree.Node(i,left=None,right=a)

BTree.pprint(tmp)

s.push(tmp)

elif i in ['∧','∨']:

a=s.pop()

b=s.pop()

tmp=BTree.Node(i,left=b,right=a)

BTree.pprint(tmp)

s.push(tmp)

t=BTree()

t.root=s.pop()

return t

#### 2.3.3.4 Calculate the Expression Tree

(1) Traverse the expression tree from the root node.

(2) If the node value is an operator, recursively call the calculate function to calculate the value of this node as the root node.

(3) If the node value is a propositional argument, return its true value.

def calculate(node):

if node:

if node.val not in ('∧','∨','!'):

return values[node.val]

elif node.val =='!':

return not calculate(node.left)

elif node.val == '∧':

return calculate(node.left) and calculate(node.right)

elif node.val == '∨':

return calculate(node.left) or calculate(node.right)

#### 2.3.3.5 Find all non repeating propositional arguments in propositional formulas

（1）Traverse the propositional formula. If it is not an operator, it will be added to the list of propositional arguments.

def findArgument(expr):

arguments=[]

temp=expr.split()

for i in temp:

if i not in ('!','∧','∨','(',')',' '):

arguments.append(i)

return arguments

#### 2.3.3.6 Enumerate all truth values of propositional arguments

(1) Traversal interval [0,2^n], n is the number of proposition arguments.

(2) Convert I into binary, and then convert it into a string, with a supplementary leading '0' of less than n bits.

(3) Convert each STR into decimal system, corresponding to the truth value of the nth proposition argument, and add it to the truth dictionary.

(4) Each time the loop is executed, the truth dictionary is added to the list of truth dictionaries.

def enum(arguements):

lst=[]

for i in range(2\*\*len(arguments)):

dic={}

s=str(bin(i))[2::]

if len(s)<len(arguments):

s='0'\*(len(arguments)-len(s))+s

s=[int (k) for k in s]

for j in range(len(arguments)):

dic[arguments[j]]=s[j]

lst.append(dic)

return lst

#### 2.3.3.7 Visualization of the Binary Tree

import networkx as nx

import matplotlib.pyplot as plt

#Recursively traverse the binary tree and calculate the seat of each node

def create\_graph(G, node, pos={}, x=0, y=0, layer=1):

pos[node.val] = (x, y)

if node.left:

G.add\_edge(node.val, node.left.val)

l\_x, l\_y = x - 1 / 2 \*\* layer, y - 1

l\_layer = layer + 1

create\_graph(G, node.left, x=l\_x, y=l\_y, pos=pos, layer=l\_layer)

if node.right:

G.add\_edge(node.val, node.right.val)

r\_x, r\_y = x + 1 / 2 \*\* layer, y - 1

r\_layer = layer + 1

create\_graph(G, node.right, x=r\_x, y=r\_y, pos=pos, layer=r\_layer)

return (G, pos)

def draw(node):

#Draw a graph with a node as the root

graph = nx.DiGraph()

graph, pos = create\_graph(graph, node)

fig, ax = plt.subplots(figsize=(8, 10))

nx.draw\_networkx(graph, pos, ax=ax, node\_size=1000)

plt.show()

#### 2.3.3.8 Expansion requirements - calculate the value of arithmetic expression

from Stack import Stack

import BTreeVisualization as BTV

from BTree import BTree

prec = {'\*': 2,'/': 2, '+': 1,'-':1,'(': 0, ')': 0}

def infix\_to\_postfix(expr):

ops = Stack()

postfix = []

toks = expr.split()

for c in toks:

if c == '(':

ops.push(c)

elif c == ')':

while ops.peek() != '(':

postfix.append(ops.pop())

ops.pop()

elif c.isdigit():

postfix.append(c)

else:

while bool(ops) and prec[ops.peek()] >= prec[c]:

postfix.append(ops.pop())

ops.push(c)

while bool(ops):

postfix.append(ops.pop())

return postfix

def turnTree(postfix):

s=Stack()

for i in postfix:

if i.isdigit():

s.push(BTree.Node(i))

elif i in ['+','-','/','\*']:

a=s.pop()

b=s.pop()

tmp=BTree.Node(i,left=b,right=a)

BTree.pprint(tmp)

s.push(tmp)

t=BTree()

t.root=s.pop()

return t

def calculate(node):

if node:

if node.val not in ('+','-','\*','/'):

return int(node.val)

elif node.val =='+':

return calculate(node.left) + calculate(node.right)

elif node.val == '-':

return calculate(node.left) - calculate(node.right)

elif node.val == '\*':

return calculate(node.left) \* calculate(node.right)

elif node.val == '/':

return calculate(node.left) / calculate(node.right)

a=input()

a=infix\_to\_postfix(a)

print(a)

a=turnTree(a)

print(calculate(a.root))

## 2.4 Debugging analysis

### 2.4.1 Problems encountered and Solutions

Problem encountered: when constructing the expression tree, the sequence of arguments of binary operator proposition is inverted.

Solution: when constructing the expression tree with the help of stack, take the proposition argument of pop first as the right node, and the proposition argument of pop later as the left node。

### 2.4.2 Time and Space Complexity

Time complexity: O (n)

Space complexity: O (n)

## 2.5 User 's Manual

### 2.5.1 Input

Enter a propositional formula, and the arguments and operators of each proposition are separated by spaces. If the formula is illegal, re-enter it according to the prompt.

### 2.5.2 Output

Output the postfix expression (post order traversal of binary tree), the construction process of expression tree, expression tree, and the truth table of expression tree in turn.

## 2.6 Test Data and Test Results

### Test Case 1

|  |  |
| --- | --- |
| Test Input | ∧ ( ! A ∧ B ) ∨ C |
| Test Purpose | Function of checking the validity of formulas |
| Correct Output | Retry! |
| Actual Output | Retry! |
| Cause of error |  |
| current state | Passed |

### Test Case 2

|  |  |
| --- | --- |
| Test Input | ( ! A ∧ B ) ∨ C |
| Test Purpose | Check the function of infix expression to suffix expression, and check the function of establishing expression tree and visualization |
| Correct Output | A ! B ∧ C ∨ |
| Actual Output | A ! B ∧ C ∨Figure 2022-07-07 095000Figure 2022-07-07 095004Figure 2022-07-07 095007 |
| Cause of error |  |
| current state | Passed |

### Test Case 3

|  |  |
| --- | --- |
| Test Input | ( ! A ∧ B ) ∨ A ∧ C |
| Test Purpose | The function of checking and judging the number of arguments of propositions and printing the truth table. |
| Correct Output | {'A': 0, 'B': 0, 'C': 0} 0 {'A': 0, 'B': 0, 'C': 1} 0  {'A': 1, 'B': 0, 'C': 0} 0 {'A': 1, 'B': 0, 'C': 1} 1  {'A': 0, 'B': 1, 'C': 0} 0 {'A': 0, 'B': 1, 'C': 1} 1  {'A': 1, 'B': 1, 'C': 0} 0 {'A': 1, 'B': 1, 'C': 1} 1  {'A': 0, 'B': 0, 'C': 0} 0 {'A': 0, 'B': 0, 'C': 1} 0  {'A': 1, 'B': 0, 'C': 0} 0 {'A': 1, 'B': 0, 'C': 1} 1  {'A': 0, 'B': 1, 'C': 0} 0 {'A': 0, 'B': 1, 'C': 1} 1  {'A': 1, 'B': 1, 'C': 0} 0 {'A': 1, 'B': 1, 'C': 1} 1 |
| Actual Output | {'A': 0, 'B': 0, 'C': 0} 0 {'A': 0, 'B': 0, 'C': 1} 0  {'A': 1, 'B': 0, 'C': 0} 0 {'A': 1, 'B': 0, 'C': 1} 1  {'A': 0, 'B': 1, 'C': 0} 0 {'A': 0, 'B': 1, 'C': 1} 1  {'A': 1, 'B': 1, 'C': 0} 0 {'A': 1, 'B': 1, 'C': 1} 1  {'A': 0, 'B': 0, 'C': 0} 0 {'A': 0, 'B': 0, 'C': 1} 0  {'A': 1, 'B': 0, 'C': 0} 0 {'A': 1, 'B': 0, 'C': 1} 1  {'A': 0, 'B': 1, 'C': 0} 0 {'A': 0, 'B': 1, 'C': 1} 1  {'A': 1, 'B': 1, 'C': 0} 0 {'A': 1, 'B': 1, 'C': 1} 1 |
| Cause of error |  |
| current state | Passed |

## 2.7 Source Program List

### 2.7.1 Implementation of Simple Stack Stack.py

class Stack:

def \_\_init\_\_(self):

self.data = []

def push(self, val):

self.data.append(val)

def pop(self):

assert not self.empty()

ret = self.data[-1]

del self.data[-1]

return ret

def peek(self):

assert not self.empty()

return self.data[-1]

def empty(self):

return len(self.data) == 0

def \_\_bool\_\_(self):

return not self.empty()

### 2.7.2 Implementation of Binary Tree BTree.py

import networkx as nx

import matplotlib.pyplot as plt

class BTree:

class Node:

def \_\_init\_\_(self, val, left=None, right=None):

self.val = val

self.left = left

self.right = right

def \_\_iter\_\_(self):

def iter\_rec(n):

if n:

yield from iter\_rec(n.left)

yield from iter\_rec(n.right)

yield n

return iter\_rec(self.root)

def \_\_init\_\_(self):

self.root = None

def height(node):

def height\_rec(t):

if not t:

return 0

else:

return 1 + max(height\_rec(t.left), height\_rec(t.right))

return height\_rec(node)

def pprint(node,width=128):

height = BTree.height(node)

nodes = [(node, 0)]

prev\_level = 0

repr\_str = ''

while nodes:

n,level = nodes.pop(0)

if prev\_level != level:

prev\_level = level

repr\_str += '\n'

if not n:

if level < height-1:

nodes.extend([(None, level+1), (None, level+1)])

repr\_str += '{val:^{width}}'.format(val='-', width=width//2\*\*level)

elif n:

if n.left or level < height-1:

nodes.append((n.left, level+1))

if n.right or level < height-1:

nodes.append((n.right, level+1))

repr\_str += '{val:^{width}}'.format(val=n.val, width=width//2\*\*level)

print(repr\_str)

print('-'\*128)

### 2.7.3 Binary Tree Visualization BtreeVisualization.py

import networkx as nx

import matplotlib.pyplot as plt

def create\_graph(G, node, pos={}, x=0, y=0, layer=1):

pos[node.val] = (x, y)

if node.left:

G.add\_edge(node.val, node.left.val)

l\_x, l\_y = x - 1 / 2 \*\* layer, y - 1

l\_layer = layer + 1

create\_graph(G, node.left, x=l\_x, y=l\_y, pos=pos, layer=l\_layer)

if node.right:

G.add\_edge(node.val, node.right.val)

r\_x, r\_y = x + 1 / 2 \*\* layer, y - 1

r\_layer = layer + 1

create\_graph(G, node.right, x=r\_x, y=r\_y, pos=pos, layer=r\_layer)

return (G, pos)

def draw(node):

graph = nx.DiGraph()

graph, pos = create\_graph(graph, node)

fig, ax = plt.subplots(figsize=(8, 10))

nx.draw\_networkx(graph, pos, ax=ax, node\_size=1000)

plt.show()

### 2.7.4 main T2\_main.py

from Stack import Stack

import BTreeVisualization as BTV

from BTree import BTree

def isalpha(word):

for i in ('!','∧','∨','(',')',' '):

if i in word:

return False

return True

def findArgument(expr):

arguments=[]

temp=expr.split()

for i in temp:

if i not in ('!','∧','∨','(',')',' '):

arguments.append(i)

return arguments

def check(expr):

s = Stack()

expr=expr

if expr[0] in ['∧','∨']:

return False

elif (expr[-1] in ['∧','∨','!']):

return False

else:

for i in range(len(expr)):

if i<len(expr)-1:

if expr[i] in ['∧','∨']:

if expr[i+1] in ['∧','∨',')']:

return False

elif expr[i]=='!':

if expr[i+1] in ['∧','∨',')']:

return False

elif expr[i]=='(':

if expr[i+1] in ['∧','∨']:

return False

elif expr[i]==')':

if expr[i+1]=='!':

return False

if expr[i] == '(':

s.push(expr[i])

elif expr[i] == ')':

if s.empty():

return False

elif s.pop() != '(':

return False

return s.empty()

prec = {'!': 2,'∧': 1, '∨': 1,'(': 0, ')': 0}

def infix\_to\_postfix(expr):

ops = Stack()

postfix = []

toks = expr.split()

for c in toks:

if c == '(':

ops.push(c)

elif c == ')':

while ops.peek() != '(':

postfix.append(ops.pop())

ops.pop()

elif isalpha(c):

postfix.append(c)

else:

while bool(ops) and prec[ops.peek()] >= prec[c]:

postfix.append(ops.pop())

ops.push(c)

while bool(ops):

postfix.append(ops.pop())

return postfix

def turnTree(postfix):

s=Stack()

for i in postfix:

if isalpha(i):

s.push(BTree.Node(i))

elif i=='!':

a=s.pop()

tmp=BTree.Node(i,left=None,right=a)

BTree.pprint(tmp)

s.push(tmp)

elif i in ['∧','∨']:

a=s.pop()

b=s.pop()

tmp=BTree.Node(i,left=b,right=a)

BTree.pprint(tmp)

s.push(tmp)

t=BTree()

t.root=s.pop()

return t

def calculate(node):

if node:

if node.val not in ('∧','∨','!'):

return values[node.val]

elif node.val =='!':

return not calculate(node.left)

elif node.val == '∧':

return calculate(node.left) and calculate(node.right)

elif node.val == '∨':

return calculate(node.left) or calculate(node.right)

def enum(arguements):

lst=[]

for i in range(2\*\*len(arguments)):

dic={}

s=str(bin(i))[2::]

if len(s)<len(arguments):

s='0'\*(len(arguments)-len(s))+s

s=[int (k) for k in s]

for j in range(len(arguments)):

dic[arguments[j]]=s[j]

lst.append(dic)

return lst

print("请输入命题公式 示例:( ! A ∧ B ) ∨ C")

expr=input()

values=None

legal=check(expr)

while not legal:

print("Retry!")

expr=input()

legal=check(expr)

else:

arguments=findArgument(expr)

expr1=infix\_to\_postfix(expr)

t=turnTree(expr1)

explain=enum(arguments)

BTree.pprint(t.root)

for i in range(2\*\*len(arguments)):

values=explain[i]

print(explain[i],calculate(t.root))

### 2.7.5 Expansion requirements main T2\_ExtendedRequirements.py

from Stack import Stack

import BTreeVisualization as BTV

from BTree import BTree

prec = {'\*': 2,'/': 2, '+': 1,'-':1,'(': 0, ')': 0}

def infix\_to\_postfix(expr):

ops = Stack()

postfix = []

toks = expr.split()

for c in toks:

if c == '(':

ops.push(c)

elif c == ')':

while ops.peek() != '(':

postfix.append(ops.pop())

ops.pop()

elif c.isdigit():

postfix.append(c)

else:

while bool(ops) and prec[ops.peek()] >= prec[c]:

postfix.append(ops.pop())

ops.push(c)

while bool(ops):

postfix.append(ops.pop())

return postfix

def turnTree(postfix):

s=Stack()

for i in postfix:

if i.isdigit():

s.push(BTree.Node(i))

elif i in ['+','-','/','\*']:

a=s.pop()

b=s.pop()

tmp=BTree.Node(i,left=b,right=a)

BTree.pprint(tmp)

s.push(tmp)

t=BTree()

t.root=s.pop()

return t

def calculate(node):

if node:

if node.val not in ('+','-','\*','/'):

return int(node.val)

elif node.val =='+':

return calculate(node.left) + calculate(node.right)

elif node.val == '-':

return calculate(node.left) - calculate(node.right)

elif node.val == '\*':

return calculate(node.left) \* calculate(node.right)

elif node.val == '/':

return calculate(node.left) / calculate(node.right)

a=input()

a=infix\_to\_postfix(a)

print(a)

a=turnTree(a)

print(calculate(a.root))