IMPLEMENTATION OF GRAPH ALGORITHMS

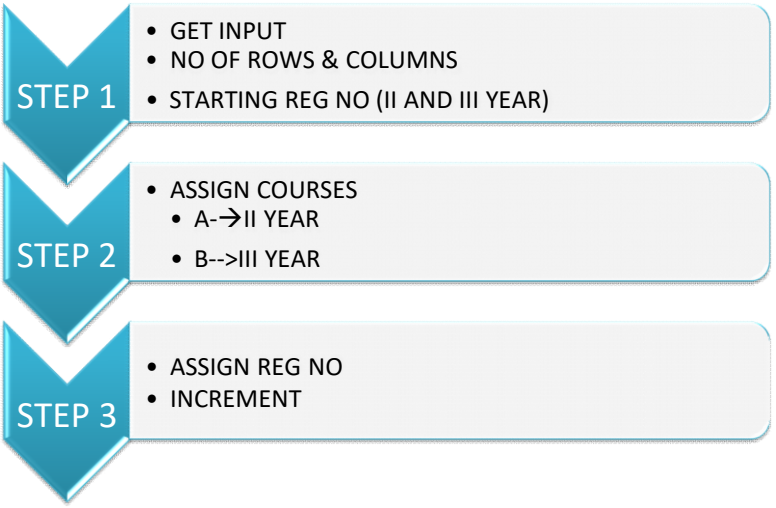
OBJECTIVE:

To build a program to find the Zig-Zag seating arrangement ofII year and Ill year students in Hall Number A10 using suitable data structures and algorithms.

CONSTRAINTS:

- Any number of rows and columns as per user's choice.
- One student can occupy a desk.
- Maximum distance should be maintained between the students ofsame year.
- The exam is on
  - Course A for all II-year students
  - Course B for all Ill year students during same session.

PROCESS FLOW:



JUSTIFICATION FOR ADT AND ALGORITHM IMPLEMENTED:

- Here we have used ARRAYS.
- Arrays store elements in contiguous memory locations, resulting ineasily calculable addresses for the elements stored and this allowsfaster access to an element at a specific index other than Map, Queue, Set, Stack and Table .
- In case of ADT , it is difficult to storing and finding the data's.

This difference in the data storage scheme decides which data structurewould be more suitable for a given situation

ALGORITHM:

- Import the PrettyTablefrom prettytable.
- As per the user choices enter the number of rows*('row')* andcolumns*('col')* in A10 halls.
- Enter the register number of second *('reg_no_sec')*and third years*('reg_no_third')*.
- Split the register number into two as per case before "D" and after"D" present in that. After that store it into list.
- Create the new stack *'a'* and then initialise values to zero
- Create a *'courses*' list and point the elements using temporary *index=0*.
- For separate the number of rows and columns in prettytable addingthe extra columns to addressing the rows and also addressing the columns using temporary stack.
- For first column one second year student(courses) and next row onethird year student(courses) and so on.
- If the number of rows and columns is even
    - check the previous row respective column's courses are same.
    - Change the courses alternatively for zig zag arrangements.
- If the '***index'*** is greater than one and assign to zero for zig zagpattern.
- After arrangements for courses,regnos are appended accordingly.
- As per user choices it starts with given register number for respective years
- Append the reg no repectively ,then increment the register number.
- The register number is already split it into two elements and store into list. The first element doesn't any changes is command for all students and the second element is increment for every changes. Repeat the previous two steps for both years.
- As per our college register number, after the "D" it has zerosapended in the fron according to the register number.
- When register number is incremented ,zeros doesn't add in that. Soadd the zeros as per college register number format using if condition.

Print the zig zag hall arrangement in A10 with ***prettytable***.

COMPLETE SOURCE CODE:

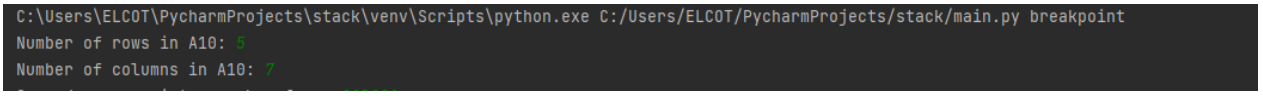
```
from prettytable import PrettyTable
row = int(input("Number of rows in A10: "))// get no of rows
col = int(input("Number of columns in A10: "))// get no of columns
reg_no_sec = input("Second year register number from: ").split("D")//starting reg no (II year)
reg_no_third = input("Third year register number from: ").split("D") //starting reg no (III year)
if_name_=="_main_":
   a=[];q=[] // initialise array a
   for i in range(0,row):
      a.append([0 for j in range(0,col+1)]) // initialise  array elements to  0
   courses=["A","B"] // array for courses
   index=0// index for accessing courses array
   for j in range(0,row,1):
      for i in range(0,col+1,1):
         if i==0:// first column – row index
            a[j][i]=str(j+1)
         if i!=0:// other columns
            a[j][i]=courses[index] // assign course
            index+=1// increment index
         if j>0 and a[j][i]==a[j-1][i]:// check this element's course with previous
                                          row's respective element's course if same
               if a[j][i]=="A":        // if A assign B and vice versa
                  a[j][i]="B"
               else:
                  a[j][i]="A"
if index>1://setting index
                  index=0
temp=["ROW/COLUMNS"]
for i in range(0,col+1):
   if col!=len(temp)-1: temp.append(str(i+1))
   for j in range(0,row):
      if a[j][i]=="B":// if course B the element will be as "B-19DXXX" XXX –reg no
         a[j][i]=0
         t = "B-" + reg_no_sec[0] + "D" + reg_no_sec[1]
         reg_no_sec[1] = int(reg_no_sec[1]) + 1
         if int(reg_no_sec[1]) < 10:
            reg_no_sec[1] = "00" + str(reg_no_sec[1])
         elif int(reg_no_sec[1]) > 9 and int(reg_no_sec[1]) < 100:
            reg_no_sec[1] = "0" + str(reg_no_sec[1])
         else:  // if course A  the element will be as "A-20DXXX" XXX –reg no
                  reg_no_sec[1] = str(reg_no_sec[1])a[j][i] = t
               if a[j][i]== "A":
                  a[j][i]=0
                  t = "B-" + reg_no_third[0] + "D" + reg_no_third[1]
                  reg_no_third[1] = int(reg_no_third[1]) + 1
               if int(reg_no_third[1]) < 10:
                  reg_no_third[1] = "00" + str(reg_no_third[1])
```

```
                elif int(reg_no_third[1]) > 9 and int(reg_no_third[1]) < 100:reg_no_third[1] =
                    "0" + str(reg_no_third[1])
                else:
                    reg_no_third[1] = str(reg_no_third[1])
                    a[j][i]=t
print("\n-----Courses A for all second years----------") // print the table
print("-----Courses B for all Third years ---------- ")
print("\nEntrance from here--------->")
table=PrettyTable(temp)for i
in range(0,row,1):
   table.add_row(a[i])
print(table)
```

USE CASE:

EXAMPLE 1:

1. Enter the number of rows and columns:



2. Enter the starting reg no's of both years:

3. The respective courses assigned for the batches will be displayed:

4. Next the Zig-Zag arrangements of the hall will be displayed

First courses are assigned

A	B	A	B	A	B	A
B	A	B	A	B	A	B
A	B	A	B	A	B	A
B	A	B	A	B	A	B
A	B	A	B	A	B	A

Then reg no's are assigned:

A-20D001	B-19D003	A-20D006	B-19D008	A-20D011	B-19D013	A-20D016
B-19D001	A-20D004	B-19D006	A-20D009	B-19D011	A-20D014	B-19D016
A-20D002	B-19D004	A-20D007	B-19D009	A-20D012	B-19D014	A-20D017
B-19D002	A-20D005	B-19D007	A-20D010	B-19D012	A-20D015	B-19D017
A-20D003	B-19D005	A-20D008	B-19D010	A-20D013	B-19D015	A-20D018

At last, the hall arrangement is printed:

```
C:\Users\ELCOT\PycharmProjects\stack\venv\Scripts\python.exe C:/Users/ELCOT/PycharmProjects/stack/main.py breakpoint
Number of rows in A10: 5
Number of columns in A10: 7
Second year register number from: 20D001
Third year register number from: 19D001

-----Courses A for all second years-----
-----Courses B for all Third years-----

Entrance from here--->
+-------------+----------+----------+----------+----------+----------+----------+----------+
| ROW/COLUMNS |    1     |    2     |    3     |    4     |    5     |    6     |    7     |
+-------------+----------+----------+----------+----------+----------+----------+----------+
|      1      | A-20D001 | B-19D003 | A-20D006 | B-19D008 | A-20D011 | B-19D013 | A-20D016 |
|      2      | B-19D001 | A-20D004 | B-19D006 | A-20D009 | B-19D011 | A-20D014 | B-19D016 |
|      3      | A-20D002 | B-19D004 | A-20D007 | B-19D009 | A-20D012 | B-19D014 | A-20D017 |
|      4      | B-19D002 | A-20D005 | B-19D007 | A-20D010 | B-19D012 | A-20D015 | B-19D017 |
|      5      | A-20D003 | B-19D005 | A-20D008 | B-19D010 | A-20D013 | B-19D015 | A-20D018 |
+-------------+----------+----------+----------+----------+----------+----------+----------+

Process finished with exit code 0
```

TIME COMPLEXITY

Time Complexity of an algorithm is the representation of the amount oftime required by the algorithm to execute to completion.
The time complexity of this program is

Zero appending in matrix : $O(n^2)$
Courses assignend in zig zag manner :$O(2n^2)$
Register numbers appended : $O(n^2)$
Print the table :$O(n)$

So,

$O(n^2 +2n^2 +n^2 +n)$

$O(4n^2 +n)$

INFERENCE:

Thus, the zig-zag seating arrangement of II year and Ill year students in Hall No. A10 using suitable data structures and algorithms.