Assignment 5

Wedding Shuffle

EC602 Design by Software

Contents

% Fall 2021

1 Introduction

1.1 Assignment Goals

The assignment goals are to help you learn about

- recursion
- avoiding $O(2^N)$ complexity for Fibonnaci-like calculations

1.2 Group Size

The maximum group size is 3 students.

1.3 Due Date

The python assignment is due 2020-11-5 at 23:59:59

1.4 Assignment Value

This assignment is worth 10 points.

1.5 Late policy

1.6 Submission Link

You can submit here: wedding hw5 submit link

2 Problem Definition

2.1 Wedding Shuffle

A number of wedding guests are seated at a circular table. The master of ceremonies asks all the guests to stand, and sit back down either one seat to the left, one seat to the right, or in the same chair.

All guests must have a chair once everyone is seated.

Your task is to determine all possible ways that the guests can do this "wedding shuffle."

2.2 Wedding Shuffle with Barriers

This second problem is similar, but now the table has a set of plexiglass barriers in between some of the chairs.

3 The assignment: `Wedding` class

Write a program `wedding.py` with a class `Wedding` which has two main methods as shown below:

```python
class Wedding:
    def __init__(self):
    def shuffle(self, guests):
    def barriers(self, guests, bars):
```

You may add other methods and data structures to the class as necessary to solve the problem.

3.1 The Wedding Class

Each `Wedding` object uses a set of names for the guests. The guests are labelled with individual characters in a string. So, for example, the string

```
abc12
```

represents five guests whose names are `a`, `b`,`c`, `1`, and `2`.

3.2 The `shuffle` method

The `shuffle` method should return all possible arrangements of the guests specified as a list of strings. The initial seating arrangement is as specified in the parameter `guests`. For visualization, suppose that `guests[0]` is seated at the one-o'clock position of a clock face, and `guests[1]` is to their left (at 2 oclock). `guests[n]` is to the left of `guests[n-1]` until we arrive back to `guests[0]` The last guest is to the immediate right of `guests[0]`, they are seated at "11 o'clock."

Here is a simple example. If the guests are `abc`, then the result of `shuffle("abc")` will be

```
abc
acb
bac
bca
cab
cba
```

3.3 The `barriers` method

The `barriers` method should return all possible arrangements of guests as a list of strings.

The barriers are specified as a list of integers. A barrier always comes before the position it specifies. So a barrier at 0 means that the person seated at position 0 cannot move to their right (to position n-1). Using the clock face analogy, a barrier at 0 is like a barrier at 12-o'clock (just to the right of `guests[0]`)

The barriers should be included in the output as a visual aid, using the `|` character.

Here is a simple example: the result of `barriers("abcd",{2})` will be

```
ab|cd
ab|dc
ba|cd
ba|dc
```

3.4 Program name

The template program and the `main` to be used for testing is provided in wedding_original.py

3.5 Libraries allowed

No libraries are allowed for the moment.

3.6 Downloads

- wedding_answers.txt this is the output of the command “tests”
- wedding_original.py this is the template file

3.7 Grading Scheme

4 Grading Scheme

Out of 100 total points, the grade is determined as follows:

- 30 points for passing the specifications of shuffles
- 30 points for passing the specifications of barriers
- 30 points for program speed (your program must pass the specifications to get credit for this)
- 10 points for pylint / pycodestyle.