6.（発展）課題4, 5

（発展）課題4：混雑回避アルゴリズム

- 先ほどのテーマパークシミュレーションでは、あらかじめプログラムで与えた訪問順序に基づいてシミュレーションを実行しました。
- 実際には、たとえば案内板を見て、お客さん自身が「予想待ち時間の短いアトラクション」を探して移動するのが一般的です。

- そこで、次のアトラクションを選択する際に、訪問予定のアトラクションの中で最も待ち行列長が短いアトラクションを選択するように、プログラムを変更してみてください。（lecture5-e-4.ipynb）
  - その際には、Visitorクラスに対して若干の修正をした上で、next_action関数を変更してみてください。

```
from collections import deque

# 各アトラクションの待ち行列
attractions = [deque(), deque(), deque()]
# 次のアトラクションに移動するお客さんの待機場所
next_pool = deque()
# アトラクションを回り終わったお客さんの待機場所
goal = deque()


# お客さんの数
num_of_visitors = 3

# 客を表現するクラス
class Visitor:
    def __init__(self, name, lst):
        self.name = name # 客の名前
        self.attr = lst  # 訪問したいアトラクションのリスト

# 客のデータ
a = Visitor("A", [0, 1, 2])  # a さんは 0,1,2 に行きたい
b = Visitor("B", [0, 1, 2])  # b さんは 0,1,2 に行きたい
c = Visitor("C", [0, 1, 2])  # c さんは 0,1,2 に行きたい


# next_poolに登録
next_pool.append(a)
next_pool.append(b)
next_pool.append(c)
```

dequeではなく、リストに変更する

（訪問順序は決まっていないため）

```
# 客が次の行動を選択する関数
def next_action(visitor):
    if len(visitor.attr) > 0:
        # お客さん visitor の残りのアトラクションの中で、
        # 待ち行列長が最小のアトラクションに並ぶ
```

ここを考えてください。

```
    else:
        # 回り終えたのでゴールへ
        print(visitor.name, "さんがアトラクションを回り終えました。")
        goal.append(visitor)

# メインのプログラム
for t in range(100):
    （以下同じ）
```

（発展）課題5：テーマパークシミュレーションの拡張

- 実際のテーマパークを考えるにあたっては、これまでのテーマパークシミュレーションのままでは十分ではありません。以下のような拡張が必要です。
  1. お客さんの数を N 人に
  2. アトラクションの数を M 個に
  3. 各アトラクションごとにサービス時間（service time）が異なる
  4. 各アトラクションごとに単位時間あたりに案内できる人数（capacity）が異なる
  5. お客さんの好みの反映（必ずしも、存在するすべてのアトラクションに訪問するわけではない）
  6. アトラクションの地理的配置を考えて、次のアトラクションへの移動時間（moving time）を考慮

- 以上の中からいくつかの要素を取り入れたテーマパークシミュレーションのプログラムを作って、それを提出してください。
  - どのような拡張を行ったかわかるように説明も書いてください。
  - 上記以外のオリジナルの要素を考えてもかまいません。

参考：テーマパーク問題

- 前スライド 1. ～ 6. の拡張を適用したうえで、次のような問題を考えます：
  - i番目のお客さんがアトラクションを回り終えるまでの時間（滞在時間：visiting time）を vt_i とし、vt_i の平均を vt とします。
  - vt自体は、テーマパークの基本設定とお客さんのプラン（訪問順序の組み合わせ）によって決まります。
  - テーマパークの基本設定を1つ決めたときに、vt を最小化するようなお客さんのプランはどのようなプランでしょうか？
- 上記の問題は、テーマパーク問題（theme park problem）と呼ばれており、現在も研究されています。

- 来場者の人数をN、アトラクション数をMとしたときに、テーマパーク問題の解（最小化するプラン）を、プランの全探索によって見つけるための時間のオーダーはいくつになるでしょう？