PPPD - Lab. 09

Zadanie punktowane, lab 09, grupa A, 2021/2022, autor: Łukasz Brzozowski

Temat: Uproszczony model Barabásiego–Alberta

Część I

Wstęp

Rozważmy dowolną grupę n osób, które dla uproszczenia będziemy oznaczać liczbami ze zbioru $S := \{0, 1, \ldots, n-1\}$. Celem zadania jest zamodelowanie relacji znajomości pomiędzy tymi osobami oraz zaobserwowanie własności tego modelowania. Relację znajomości oznaczmy $\rho \subseteq S \times S$ – jest ona symetryczna i przeciwzwrotna, czyli

$$\forall_{x,y \in S}\ \rho(x, y) \implies \rho(y, x),$$

$$\forall_{x \in S}\ \sim \rho(x, x).$$

(Oczywiście $\rho(x, y) \iff (x, y) \in \rho$)
Intuicyjnie, jeśli osoba x zna osobę y, to osoba y zna osobę x. Wymagamy też, żeby nikt nie był swoim własnym znajomym (dobrym przykładem jest tutaj relacja bycia znajomymi na Facebooku).

W celu zareprezentowania relacji pomiędzy naszymi osobami w pamięci komputera, możemy użyć macierzy A o rozmiarze $n \times n$, którą będziemy nazywać macierzą znajomości. Pamiętając, że nasze osoby identyfikujemy z kolejnymi liczbami naturalnymi, definiujemy A jako macierz elementów $a_{i,j}$ (pierwszy indeks to numer wiersza, drugi indeks to numer kolumny) takich, że:

$$a_{i,j} = \begin{cases} 1, & \text{jeśli } \rho(i, j) \\ 0, & \text{wpp.,} \end{cases}$$

przy czym wiersze i kolumny wszystkich macierzy w tym zadaniu indeksujemy od zera. Dodatkowo, będzie interesowała nas popularność poszczególnych osób, czyli liczba ich znajomych. Oznaczmy tę liczbę przez deg (stopień popularności), czyli

$$\deg(i) = |\{j \in S : \rho(i, j)\}|,$$

oraz przez $k(i)$ liczbę osób, które mają dokładnie i znajomych:

$$k(i) = |\{x \in S : \deg(x) = i\}|$$

Zadanie

W pierwszej części zadania chcemy wygenerować macierz znajomości dla grupy n osób, przy czym każdy zna każdego (poza samym sobą). Ta macierz posłuży nam za punkt wyjściowy w dalszej części zadania. Napisz funkcje:

- `macierz_startowa(n)` – która przyjmuje liczność grupy osób n i zwraca macierz znajomości grupy, w której każdy zna każdego (poza samym sobą).

- `licznosc_stopni(A)` – która przyjmuje macierz znajomości A odpowiadająca pewnej grupie osób S z relacją ρ i zwraca listę ℓ, która na i-tej pozycji ma wartość $k(i)$. Długość listy ℓ wynosi

  $$\max(\deg(i))_{i \in S} + 1,$$

  czyli zawiera wszystkie niezerowe zaobserwowane wartości, ale nie ma zer na końcu. Możesz użyć wbudowanej funkcji `max`.

Przykłady wywołań

```
A = macierz_startowa(4)
l = licznosc_stopni(A)
for row in A:
  print(row)
```

```
## [0, 1, 1, 1]
## [1, 0, 1, 1]
## [1, 1, 0, 1]
## [1, 1, 1, 0]
```

```
print(l)
```

```
## [0, 0, 0, 4]
```

Część II

Wstęp

Spróbujemy teraz zamodelować, jak może rosnąć grupa znajomych. Zakładamy następujący scenariusz:

0. Zaczynamy z pewnym gronem osób S, w którym każdy zna każdego (poza samym sobą).
1. Do grona S dołącza jedna nowa osoba.
2. Ta nowa osoba zapoznaje się z dokładnie m losowo wybranymi osobami z S (relacja jest symetryczna, więc jeśli x poznaje y, to y poznaje x).

Kroki 1-2 powtarzamy, dopóki nasze grono nie osiągnie liczności N osób.

Zadanie

Zaimplementuj funkcję `nowe_grono(A, N, m)`, która przyjmuje macierz startową A z poprzedniego punktu oraz całkowitoliczbowe parametry N i m. Funkcja ta zwraca macierz znajomości B nowego grona osób, które jest wynikiem wzrostu grona reprezentowanego przez A – zgodnie z podanym wyżej algorytmem. Końcowe grono osób ma mieć liczność N, a w każdym kroku nowa osoba poznaje dokładnie m losowych osób. Funkcja ta musi sprawdzać, że $0 < m \leq |S_0| < N$, gdzie $|S_0|$ to początkowa liczność grona reprezentowanego przez A.

Wskazówka: Do wybrania losowo k elementów z listy `lista` bez zwracania możesz użyć funkcji `sample` z pakietu `random`:

```
losowo_wybrani = random.sample(lista, k)
```

Przykłady wywołań

```
random.seed(126)
B = nowe_grono(A, 6, 2)
for row in B:
  print(row)
```

```
## [0, 1, 1, 1, 1, 0]
## [1, 0, 1, 1, 0, 0]
## [1, 1, 0, 1, 1, 0]
## [1, 1, 1, 0, 0, 1]
## [1, 0, 1, 0, 0, 1]
## [0, 0, 0, 1, 1, 0]
```

Część III

Wstęp

Oznaczając dalej przez m liczbę osób, które poznaje nowa osoba, i przez N końcową liczność naszego grona, teoria przewiduje, że w naszym nowym gronie z części II – dla każdego k – około $p(k, m) \cdot N$ osób będzie miało dokładnie k znajomych, przy czym

$$p(k, m) = \begin{cases} 0, & \text{jeśli } k < m, \\ \frac{e^{1-\frac{k}{m}}}{m}, & \text{wpp.,} \end{cases}$$

gdzie e to liczba Eulera (w Pythonie `math.e` z pakietu `math`). Przykładowo spodziewamy się, że w nowym gronie, które ma 100 osób i które powstało z parametrem $m = 3$, będzie około $100 \cdot p(7, 3)$ osób mających dokładnie 7 znajomych.

Zadanie

W poniższych funkcjach ustalamy parametry dla wszystkich symulacji: $N = 100$, $m = 3$ i początkowa liczność grona osób $n = 5$.

Napisz funkcje:

- `p(k, m)` – która implementuje powyższy wzór na funkcję p.
- `srednie_bledy(T)` – która generuje nowe grono T razy (zgodnie z podanymi wyżej parametrami) i sprawdza, jak wartości $k(i)$ różnią się od przewidywanych wartości teoretycznych. Dokładniej, funkcja ta zwraca listę średnich różnic zaobserwowanych wartości $k(i)$ od wartości teoretycznych, czyli zwraca listę v taką, że:

  $$v_i = \frac{1}{T} \sum_{t \in \{1,\dots,T\}} \big(k_t(i) - N \cdot p(i, m)\big),$$

  gdzie $k_t(i)$ to wartość $k(i)$ zaobserwowana w t-tej symulacji. Zakładamy, że długość listy v wynosi N, ponieważ jedna osoba może mieć maksymalnie $N - 1$ znajomych.

Narysuj wykres policzonych średnich błędów z tysiąca symulacji. Dla danych dwóch list x, y zawierających odpowiednio współrzędne punktów na osiach X i Y, wykres punktów możemy narysować, korzystając z:

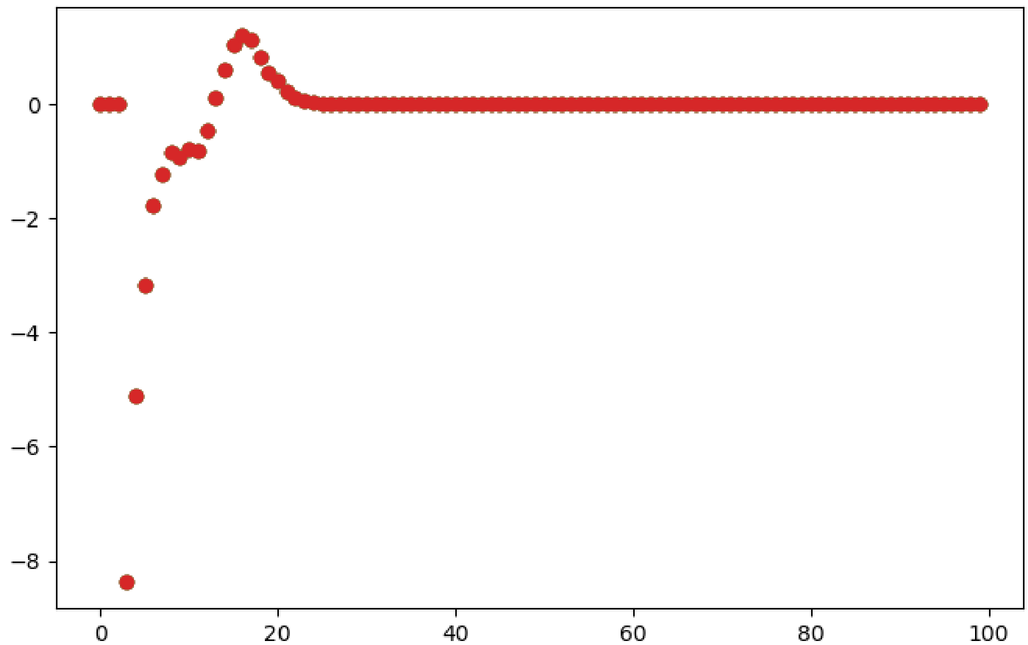
```
import matplotlib.pyplot as plt

plt.scatter(x, y)
plt.show()
```

Przykłady wywołań:

```
sb = srednie_bledy(1000)
# 5 pierwszych elementów sb
print(sb[:5])
```

```
## [0.0, 0.0, 0.0, -8.351333333333226, -5.11837701912635]
```



Rysunek 1: Otrzymany wykres błędów

Punktacja

Za poszczególne elementy można uzyskać następującą liczbę punktów:

- funkcja `macierz_startowa` – 1 pkt.
- funkcja `licznosc_stopni` – 2 pkt.
- funkcja `nowe_grono` – 3 pkt.
- funkcje `p` i `srednie_bledy` – 3 pkt.
- wykres – 1 pkt.

Uwaga

- Jeśli program się nie kompiluje (interpretuje), ocena jest zmniejszana o połowę.
- Jeśli kod programu jest niskiej jakości (nieestetycznie formatowanie, mylące nazwy zmiennych itp.), ocena jest zmniejszana o `2p`.
- W zadaniu nie można korzystać z append (w tym z operatora "+=" na listach), delete (ani operatora "-="), slice i indeksowania ujemnego.