Symulacje komputerowe w fizyce

22-23.11.2022

Ćwiczenia 7: Perkolacja

Zadanie

1. Zaimplementuj algorytm Leatha na sieci kwadratowej w przypadku filtrowania cieczy przez materiał porowaty (mamy warstwę cieczy na górnej krawędzi siatki i próbujemy przebić się do dolnej krawędzi). Zastosuj periodyczne warunki brzegowe na bocznych krawędziach.

2. Zbadaj prawdopodobieństwo utworzenia perkolującego klastra P(p) oraz średni rozmiar klastra nieperkolującego S(p) w obszarze przejścia fazowego dla różnych wielkości układu (L = 20, 50, 100).

3.* Oblicz długość korelacji ξ dla różnych wielkości układu LUB stwórz algorytm do renormalizacji układu dla b = 2.

Zadanie 1 - szczegóły

1. Tworzymy kwadratową sieć LxL (niech na początek L = 20). Pierwotne węzły oznaczamy -1, puste 0, a zajęte 1.

```
lattice = np.ones((L, L)) * (-1)
```

2. Stosujemy zamknięty brzeg układu na górze i na dole (dodatkowa flaga -2) oraz periodyczne warunki po bokach (np. sprytne wykorzystanie dzielenia z resztą %).

3. Implementujemy kolejkę. Początkowo do kolejki dodajemy cały jeden rząd na górze układu (pierwszy pod zamkniętym warunkiem brzegowym.

4. Wykonujemy serię obrazków rosnącego klastra.

```
plt.imshow(lattice, interpolation=`nearest`, cmap = `magma`)
```

Kolejka czy stos?

Kolejka

- First In, First Out (FIFO)
- dzięki deque, nie tracimy na szybkości (w zwykłej liście usuwanie pierwszego elementu jest czasochłonne)

Stos

- Last In, First out (LIFO)
- można zaimplementować jako zwykłą listę

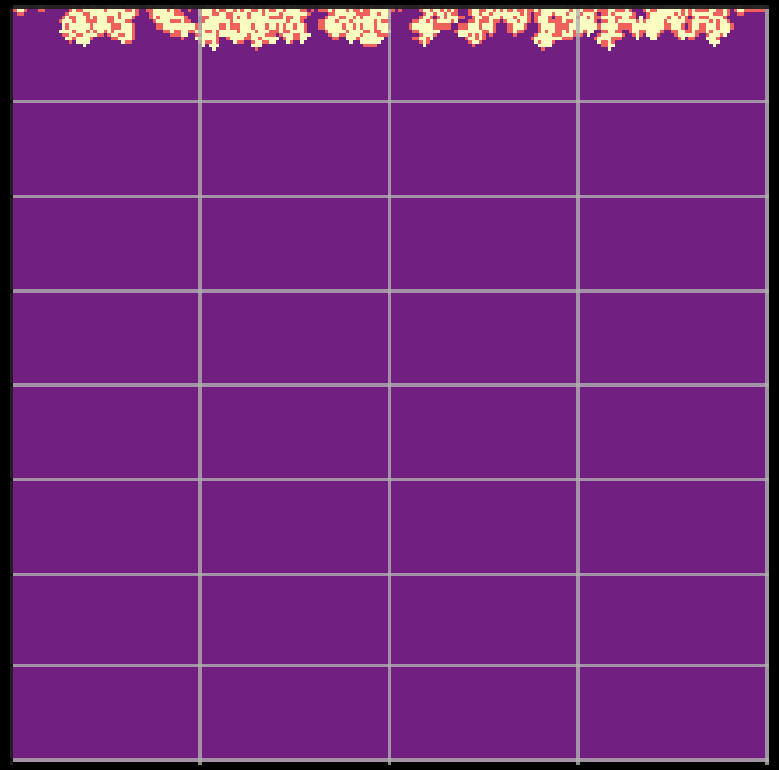
```
from collections import deque

cluster = deque() # tworzymy kolejkę
cluster.append(i) # dodajemy do kolejki
i = cluster.popleft() # zdejmujemy
# pierwszy element
```

```
cluster = [] # lub
# cluster = deque() # tworzymy stos
cluster.append(i) # dodajemy do stosu
i = cluster.pop() # zdejmujemy
# ostatni element
```

Końcowy rezultat ten sam, ale różna ewolucja!

Zadanie 1 - ewolucja



Zadanie 2 - szczegóły

1. Dla przyspieszenia, możemy od razu generować całą tablicę decyzji.

   ```
   tossing = np.random.random((L, L)) < p
   # p – zadane prawdopodobieństwo
   ```

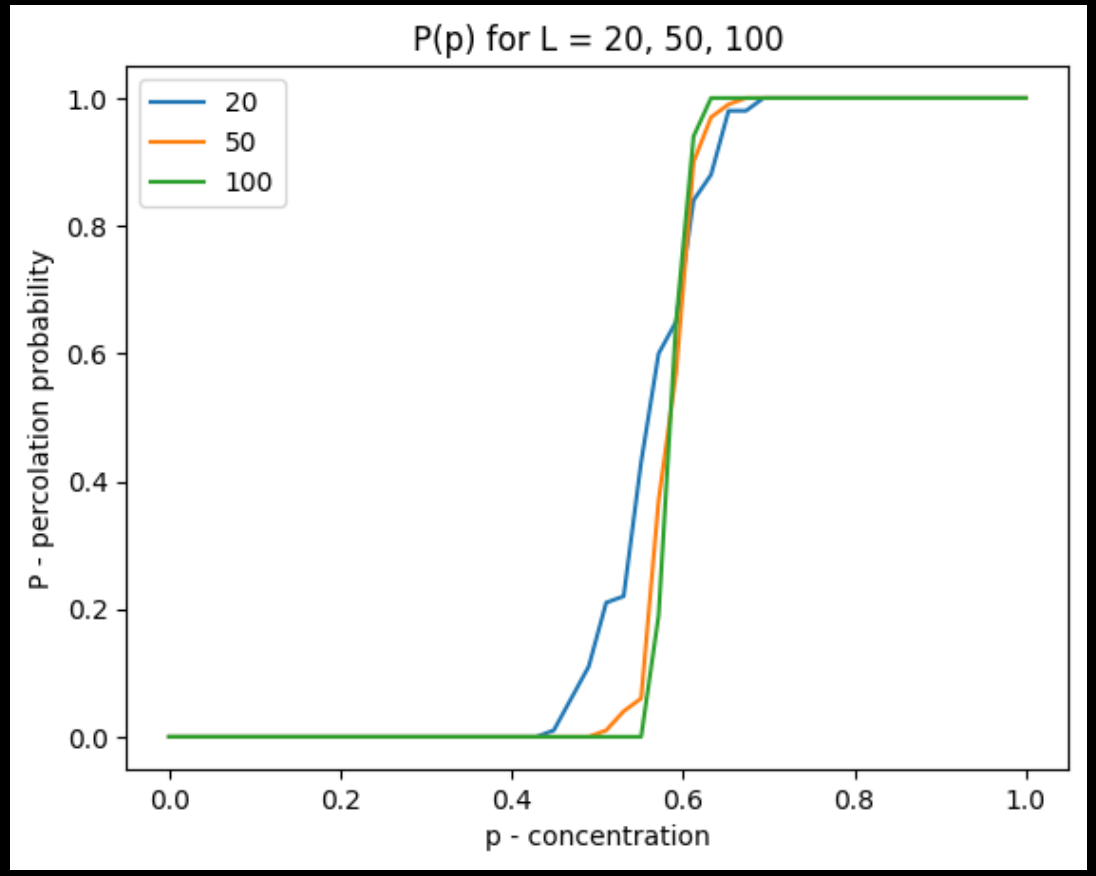
2. Wielkość wyhodowanego klastra obliczamy jedną komendą!

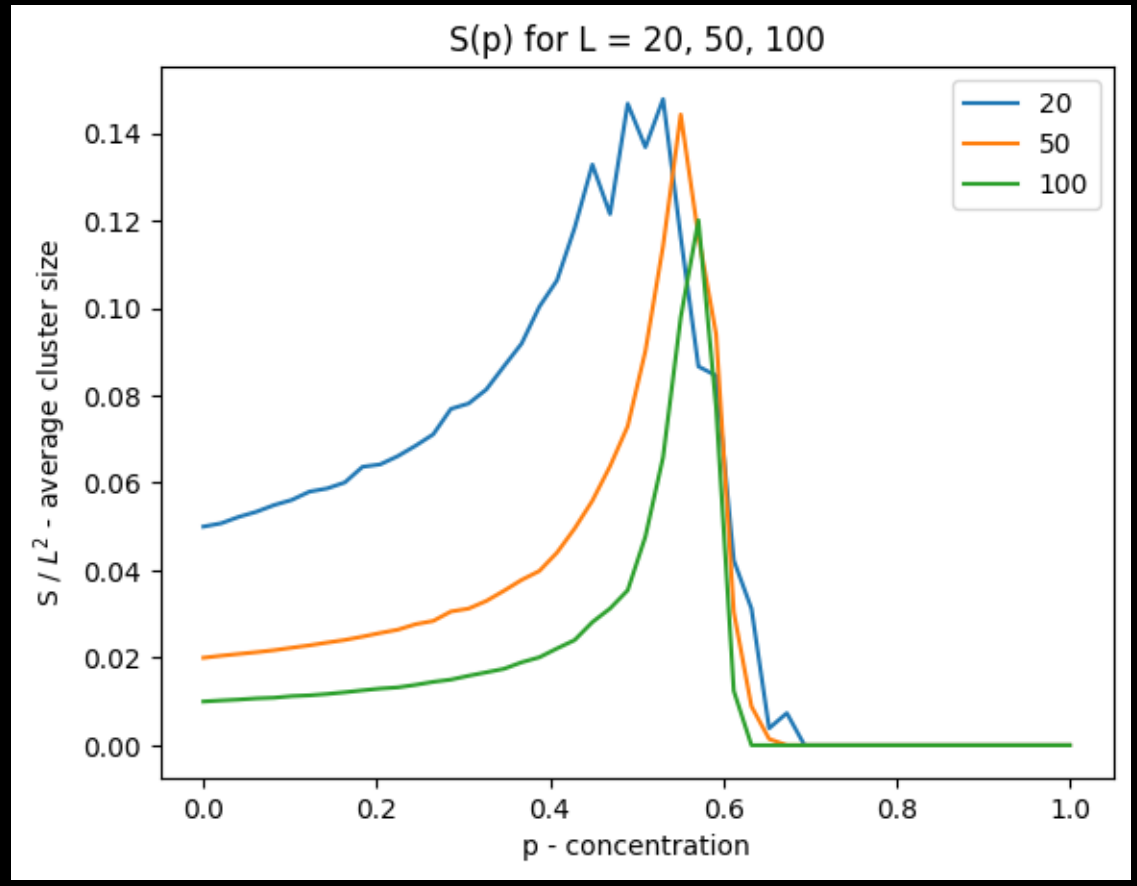
   ```
   np.sum(lattice == 1)
   ```

3. W mądry sposób sprawdzamy, czy klaster perkoluje (np. możemy policzyć sumę wartości w ostatnim rzędzie tablicy (z wyjątkiem zamkniętego brzegu)).

4. Sprawdzamy dla przedziału $p \in (0.54, 0.64)$, np. co 0.01. Dla każdej wartości zbieramy co najmniej $n = 100$ próbek.

Zadanie 2 - wykresy





Zadanie dodatkowe - długość korelacji

$$\xi = R\sqrt{2}$$

$$R^2 = \frac{1}{N}\sum_{i=1}^{N}(\vec{r}_i - \vec{r}_0)^2$$

$$\vec{r}_0 = \frac{1}{N}\sum_{i=1}^{N}\vec{r}_i$$

N – rozmiar klastra

$\vec{r}_i$ – położenia zajętych węzłów

Punktacja zadania

1. Algorytm Leatha, obrazki z ewolucji klastra na progu perkolacji ($p = 0.59$) dla dużej sieci ($L = 100$) – 0.5 pkt.

2. Wykresy prawdopodobieństwa utworzenia „nieskończonego” klastra $P(p)$ i średniego rozmiaru nieperkolacyjnego klastra $S(p)$ w przedziale $p \in (0.54, 0.64)$ dla L = 20, 50, 100 – 0.5 pkt.

3.* Wartości długości korelacji ξ dla L = 20, 50, 100 LUB obrazki układu przed i po renormalizacji – 0.2 pkt.