KORSZERŰ SZÁMÍTÁSTECHNIKAI MÓDSZEREK A FIZIKÁBAN

jegyzőkönyv

Simonová Alexandra

Perkoláció számolása NxN-es rácson

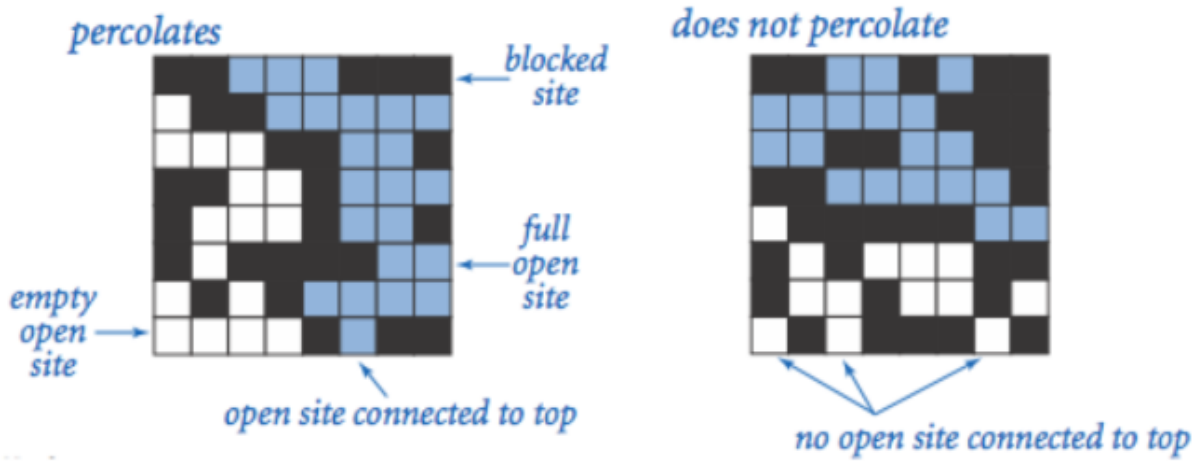
Beadás időpontja: 2021. május 7.

Bevezetés

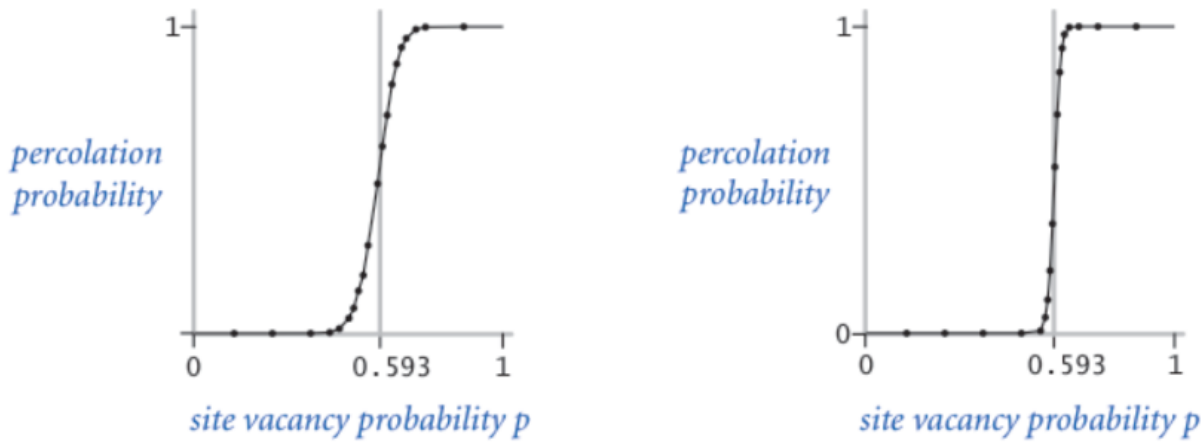
A jegyzőkönyvben a Korszerű számítástechnikai módszerek a fizikában c. tárgyhoz készült beadandó feladat elméleti leírását és működési elvét ismertetem.

Elméleti leírás

Egy rendszerben, adott esetben $N \times N$-es rácson, milyen feltételek mellett alakul ki óriás összefüggő komponens. A rács elemei vagy nyitottak, vagy zártak, és teljesen nyitott, ha minden szomszédos kapcsolódó elem nyitva van a környezetében. A rendszer *perkolál*, ha egy ilyen teljesen nyitott elem található a legelső és legutolsó sorban (vagy a legelső,legutolsó sorban és az oldalakon is). Ilyenkor egy teljesen összefüggő részhálózat alakul, melyben a kapcsolatokon (elemeken) keresztül eljuthatunk akármelyik elemhez a részhálózaton belül.



Ha az elemek, illetve a köztük lévő kapcsolatok, vagy élek függetlenül egymástól p valószínűséggel nyíltak($1-p$ valószínűséggel zártak), mi a valószínűsége hogy a rendszer perkolál? $p = 0$ esetben a rendszer nem perkolál, $p = 1$ esetén igen. Az alábbi ábrán a perkoláció valószínűsége látható az élek nyíltságának valószínűégének függvényében.



Algoritmus és működési elv

Először többféle módszerrel próbáltam végigmenni a rácson, de nem vezetett eredményre az, hogy "tényleges rácsként" kezeltem a rácsot. Majd megtaláltam Quick Union és Quick Find algoritmusokat, amivel már sokkal egyszerűbb volt a rács elemek kezelése és összekötése.

Ezeknek a működése azon alapul, hogy kialakítunk egy *array*-t, aminek az indexei reprezentálják az $N \times N$ elemet. Az *array* elemei pedig az adott elem `origin`-je, ami megváltozik, ha összekapcsolódik két elem. Az `origin` értéke a kezdeti állapotban maga az elem. Az összekapcsolásokat a `join` függvény végzi. Két elem akkor van összekötve, ha egyenlő az `origin`-jük. A `join` függvényb súlyozást is végez, annak érdekében, hogy ne alakuljanak hosszú fák az egymás alá kapcsoláskor, hogy ezzel javítsa az algoritmus hatékonyságát. A `WQUni` nevű osztály tartalmazza az ezekhez kapcsolódó tagokat és függvényeket.

A rács reprezentációja és a nyitott és egymás melletti elemek összekötése `Percolation` nevű osztály által valósul meg, ezen végzi aztán a `WQUni` nevű osztály a *Quick Union*- és *Quick Find*-al kapcsolatos műveleteket. A `PerkStats` nevű osztály `computePcrit` függvényében történik az elemek randomizált kinyitása, random szám generátorral, egyenletes eloszlásból, addig amíg a rendszer nem perkolál, az alsó és felső virtuális elem(sor) nem ér össze a nyílt elemeken keresztül. A random sor és oszlop értékek átadódnak az `open` függvénynek, ami elvégzi a kapcsolatok kinyitását és megnöveli a nyitott elemek számát 1-el. A *main* függvényben ebből az értékből számolódik a kritikus valószínűség értéke.

A rács nagyon egyszerű megjelenítését a nyitott elemek pozíciójával készítettem.

Felhasznált irodalom

- Union-Find data type[1]
- S-ayanide - Percolation[2]
- Algorithms - Robert Sedgewick and Kevin Wayne[3]
- egyéb posztok és youtube videók