Temă pentru acasă - partea C.

15 puncte [11p: C1] + [4p: C2]

Termen limită: 1-6 iunie 2024 (data precisă este stabilită de instructorul de laborator).

C1. (**Las Vegas**, 11 puncte = 1p + 5p + 4p + 1p) Let $n \in \mathbb{N}^* \setminus \{1\}$. Această problemă este despre generarea de permutări aleatoare.

(a) Implementaţi (în R) următorul algoritm aleator generarea unei permutări aleatoare: generaţi independent şi uniform n valuri aleatoare uniforme standard (i. e., din intervalul $[0, 1]$): $U_1, U_2, \ldots, U_n$. Apoi sortaţi-le în ordine crescătoare:

$$U_{\pi(1)} \leqslant U_{\pi(2)} \leqslant \ldots \leqslant U_{\pi(n)}$$

şi returnaţi $(\pi(1), \pi(2), \ldots, \pi(n))$.

Fie $k \in \mathbb{N}^*$ un număr natural nu prea mare (e. g. $k = 4$ sau 5). Se presupune că avem o sursă aleatoare de biţi şi dorim să sortăm o listă de n şiruri de biţi (cuvinte) de lungime cel puţin k.

Iniţial generăm aleator n şiruri de biţi de lungime k (fiecare bit fiind independent generat): $W_1, W_2, \ldots, W_n$; aceste cuvinte pot fi vectori în R.

(b) Implementaţi următoarea funcţie pentru compararea lexicografică strictă a două astfel de cuvinte: fie $L_{ij} = \min\{length(W_i), length(W_j)\}$, atunci

- dacă există $l \leqslant L_{ij}$ astfel încât

$$W_i[l] < W_j[l], \forall 1 \leqslant h < l \text{ şi } W_i[h] = W_j[h],$$

  atunci W_i este lexicografic strict mai mic decât W_j, sau
- dacă $W_i[h] = W_j[h], \forall 1 \leqslant h \leqslant L_{ij}$, atunci generăm aleator biţi unul câte unul şi-i adăugăm cuvântului mai scurt până unul dintre cele două cuvinte devine lexicografic strict mai mic decât celălalt sau
- dacă lungimile lor devin egale iar cuvintele sunt încă egale, atunci se generează perechi de biţi pentru amândouă cuvintele până când unul devine lexicografic strict mai mic decât celălalt.

(*Se observă că în timpul acestui proces cuvintele se pot schimba (se pot adăuga biţi la sfârşitul lor) şi dacă un cuvânt se schimbă el rămâne schimbat.*)

(c) Implementaţi algoritmul QuickSort randomizat din cursul 10 pentru ordonarea strictă a unei liste de cuvinte (peste 0/1) - iniţial având aceeaşi lungime k - folosind funcţia de mai sus pentru compararea a două cuvinte.

(*Luaţi în considerare faptul că unele cuvintele se pot modifica în cursul execuţiei, dar asta nu va schimba ordinea finală a cuvintelor).*

(d) Implementaţi (in R) următorul algoritm aleator pentru generarea unei permutări aleatoare: generaţi independent şi uniform n cuvinte de lungime k: $W_1, W_2, \ldots, W_n$, sortaţi-le folosind algoritmul QuickSort aleator de mai sus:

$$W_{\pi(1)} < W_{\pi(2)} < \ldots < W_{\pi(n)},$$

apoi returnaţi $(\pi(1), \pi(2), \ldots, \pi(n))$.

C2. (**Monte Carlo**, 4 puncte = 3p + 1p) Fie $G = (V, E)$ un graf cu $2n$ sau $(2n+1)$ noduri şi m muchii. Pentru o bipartiţie (A, B) a lui V (i. e., $A, B \subseteq V$, $A \cap B = \varnothing$ şi $A, B \neq \varnothing$) *tăietura* corespunzătoare $E(A, B)$ este

$$E(A, B) = \{uv \in E \, : \, u \in A, v \in B\}.$$

(a) Implementaţi următorul algoritm aleator pentru determinarea unei tăieturi $E(A, B)$ de cardinal maxim în G: construiţi A alegând independent şi uniform n noduri din V, apoi $B = V \setminus A$. Determinaţi apoi cardinalul tăieturii corespunzătoare.

(b) Cum creşteţi şansele de a găsi o tăietură de cardinal cât mai mare?

Rezolvările acestor exerciţii (funcţiile R şi apelurile lor) vor fi redactate într-un singur script R.