Homework part C.

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

Deadline: week 1-6 June 2024 (the precise date will be fixed by your lab teacher).

C1. (**Las Vegas**, 11 points = 1p + 5p + 4p + 1p) Let $n \in \mathbb{N}^* \setminus \{1\}$. This problem is about generating at random a permutation.

(a) Implement (in R) the following randomized algorithm for generating a permutation (at random): generate independently and uniformly n standard uniform variates (from the interval $[0,1]$): $U_1, U_2, \ldots, U_n$. Then sort them in their non-decreasing order:

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

and return $(\pi(1), \pi(2), \ldots, \pi(n))$.

Let $k \in \mathbb{N}^*$ be a small non-negative integer number (e. g. $k = 4$ or 5). Suppose now that we have a source of random bits and we want to sort a size n list of words (sequences of bits) of length at least k. The process of sorting is described below.

Initially you have to generate at random n sequences of bits of length k (each bit is independently generated): $W_1, W_2, \ldots, W_n$; these words can be in R vectors.

(b) Implement the following function for strictly comparing two such words lexicographically: let $L_{ij} = \min\{length(W_i), length(W_j)\}$, then

- if there exists a $l \leqslant L_{ij}$ such that

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

then W_i is lexicographically strictly smaller than W_j, or

- if $W_i[h] = W_j[h], \forall 1 \leqslant h \leqslant L_{ij}$, then randomly generate bits one by one for the shorter word until one of the words becomes lexicographically strictly smaller than the other or,
- if their lengths became equal and the words are still equal, then generate pair of bits one by one for both words until one of the words becomes lexicographically strictly smaller than the other.

(*Note that during this process the words could be altered (bits are added to their end) and once a word is altered it remains so.*)

(c) Implement the randomized QuickSort algorithm from the 10th lecture for strictly ordering a list of words (over 0/1) - initially having the same size k - using the above function for comparing two words.

(*Pay attention to the fact that the words could be altered during the execution but this doesn't change their final order).*

(d) Implement (in R) the following randomized algorithm for generating a permutation (at random): generate independently and uniformly n words of length k: $W_1, W_2, \ldots, W_n$, sort them using the above randomized QuickSort:

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

then return $(\pi(1), \pi(2), \ldots, \pi(n))$.

Probabilities & Statistics

C2. (**Monte Carlo**, 4 points = 3p + 1p) Let $G = (V, E)$ be a graph with $2n$ or $(2n + 1)$ vertices and m edges. For a bipartition (A, B) of V (i. e., $A, B \subseteq V$, $A \cap B = \varnothing$, and $A, B \neq \varnothing$) the corresponding *cut* $E(A, B)$ is

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

(a) Implement the following randomized algorithm for finding a cut $E(A, B)$ of maximum cardinality: build A by choosing independently and uniformly at random n vertices from V, then $B = V \setminus A$. After that compute the cardinality of the corresponding cut.

(b) How do you increase the chances to get a cut of cardinality as large as possible?

Solutions to these exercises (the corresponding R functions and their calls) will be written in an single R script.

Probabilities&Statistics