Seminar 3
Algoritmi nedetrminiști. Algoritmi probabiliști.

Ștefan Ciobâcă, Dorel Lucanu
Universitatea "Alexandru Ioan Cuza", Iași

March 8, 2022

Hint: multe formule utile (e.g. pentru serii, integrale, combinări, aranjamente, limite, probabilități) în informatica teoretică sunt disponibile aici: `https://www.tug.org/texshowcase/cheat.pdf`.

1. Scrieți un algoritm probabilist ce modelează aruncarea unui ban măsluit: se poate obține "cap" cu probabilitatea $\frac{5}{9}$ și "pajură" cu probabilitatea $\frac{4}{9}$.

2. Fie următorul algoritm probabilist (care nu primește nimic la intrare):

```
ok = true;
i = 0;
while (ok) {
    uniform x from {0, 1};
    if (x == 0) {
        ok = false;
    }
    ++i;
}
print(i);
```

   (a) Executați de câteva ori algoritmul.

   (b) Care este probabilitatea ca algoritmul să se oprească? Hint: calculați probabilitatea ca algoritmul să se oprească după exact i iterații ale buclei `while`, pentru fiecare $i \in \{1, 2, 3, \ldots\}$.

3. Fie următorul algoritm probabilist, care primește la intrare un număr natural n și întoarce un număr natural:

```
sum = 0;
for (i = 0; i < n; ++i) {
    uniform x from {0,1};
    sum = sum + x;
}
print(sum);
```

   (a) Executați de câteva ori algoritmul.

   (b) Care este media valorilor întoarse de algoritm pentru un input $n \in \mathbb{N}$ arbitrar?

4. Fie o funcție probabilistă `rand2` care întoarce cu probabilitate 0.5 valoarea 0 și cu probabilitate 0.5 valoarea 1.

   Scrieți o funcție probabilistă `zar` care nu primește niciun argument și întoarce un număr natural între 0 și 5, fiecare cu aceeași probabilitate. Puteți folosi în funcția `zar` funcția `rand2` dar nu aveți voie să folosiți altă funcție/instrucțiune probabilistă (cum ar fi `uniform`).

5. Fie o funcție probabilistă `rand2p` care nu primește niciun argument și întoarce 0 cu probabilitate p și 1 cu probabilitate $1-p$. Numărul p este un număr real $p \in (0,1)$, dar valoarea lui nu este cunoscută.

   Scrieți o funcție `rand2corect` care nu primește niciun argument și întoarce 0 cu probabilitate 0.5 și 1 cu probabilitate 0.5. În implementarea funcției `rand2corect` puteți folosi funcția `rand2p`, dar nicio altă funcție/instrucțiune probabilistă.

6. Folosindu-vă în continuare (doar) de funcția probabilistă `rand2`, scrieți o funcție `random`, care primește ca argument un număr natural n și întoarce un număr natural din mulțimea $\{0, 1, \ldots, n-1\}$, fiecare cu probabilitatea $1/n$.

7. Fie problemele `SSD1`, `SSD2`, `SSD3` din cursul 2. Scrieți câte un algoritm nedeterminist care rezolvă fiecare din aceste probleme.

8. O formulă `3-CNF` din logica propozițională (e.g. $(x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_5 \vee \neg x_2) \wedge (x_2 \vee x_2 \vee \neg x_2) \wedge (x_3 \vee x_1 \vee \neg x_2)$) poate fi reprezentată printr-o matrice v (pentru formula, precedentă, $v[n][3] = \{\{1, -2, 3\}, \{-1, 5, -2, \}, \{2, 2, -2\}, \{3, 1, -2\}\}$).

   (a) Problema 3-satisfiabilității este următoarea:

   `Input`: O matrice $v[n][3]$ care reprezintă o formulă `3-CNF` ca mai sus și n – numărul de clauze.

   `Output`: *da*, dacă formula este satisfiabilă; *nu*, altfel.

   Scrieți în Alk un algoritm nedeterminist care rezolvă problema 3-satisfiabilității.

   (b) Descrieți ca pereche input-output problema validității unei formule `3-DNF`. Puteți scrie un algoritm nedeterminist pentru problema validității?

   (c) Problema 3-non-echivalenței este următoarea:

   `Input`: Două matrici $v[n][3]$ și $w[m][3]$ care reprezintă două formule `3-CNF` ca mai sus (n este numărul de clauze din prima formulă și m numărul de clauze din cea de-a doua formulă).

   `Output`: *nu*, dacă cele două formule sunt echivalente (au aceeași valoare de adevăr pentru orice asignare); *da*, altfel.

   Scrieți în Alk un algoritm nedeterminist care rezolvă problema 3-non-echivalenței. Puteți scrie un algoritm nedeterminist pentru problema 3-echivalenței?

   (d) Scrieți un algoritm *determinist* pentru problema 3-satisfiabilității. Ce complexitate-timp are algoritmul în cazul cel mai nefavorabil?