Ștan Andrei
Grupa E13

Examen PA

Subsemnatul Ștan Andrei-Vlăduț, student al Facultății de Informatică, Universitatea Alexandru Ioan Cuza, declar că rezolvarea care urmează îmi aparține în integralitate și că cunosc regulamentul universității referitor la sancțiunile posibile (inclusiv nepromovarea disciplinei, exmatriculare) aplicate în cazz de fraudă sau tentativă de fraudă.

[redacted].

2.) b.) Probabilitatea ca algoritmul să ruleze până infinit este 0, deoarece există măcar un cazz în care algoritmul s-ar opri (care 0, 1, 2, 3, 4).

a.) Probabilitatea este: $\left(\frac{1}{6}\right)^{i-1} \cdot \frac{5}{6}$

Probabilitate fiind dedusă din: (probabilitatea de a reintra în $f())^{i-1}$. (probabilitatea de returna o valoare)

b.) (Continuare) Probabilitatea de a tinde la infinit lui $\left(\frac{1}{6}\right)^i$ (probabilitatea de a rula la infinit)

[redacted] $\lim_{i \to \infty} \left(\frac{1}{6}\right)^i = \lim_{i \to \infty} \frac{1^i}{6^i} = 0$

c.) $\frac{2}{6} + \frac{1}{6} \cdot \frac{2}{6} + \frac{1}{6} \cdot \frac{2}{6} \cdot \frac{2}{6} + \dots$

$\lim_{n \to \infty} \left(\left(\frac{1}{6}\right)^n \cdot \frac{2}{6} \right) = 0$ (relație de dependență)

$\searrow 0$

$P = \frac{2}{6} + 0 = \frac{2}{6}$

d.) $\frac{2}{6} + \frac{1}{6}\cdot\frac{2}{6} + \frac{1}{6}\cdot\frac{2}{6}\cdot\frac{2}{6} + \dots$

$\lim_{n\to\infty}\left(\left(\frac{1}{6}\right)^n \cdot \frac{2}{6}\right) = 0$ (limita $\to 0$) $\quad P = \frac{2}{6} + 0 = \frac{2}{6}$

e.) $\frac{1}{6} + \frac{1}{6}\cdot\frac{1}{6} + \frac{1}{6}\cdot\frac{1}{6}\cdot\frac{1}{6} + \dots$

$\lim_{n\to\infty}\left(\left(\frac{1}{6}\right)^n \cdot \frac{1}{6}\right) = 0$ (limita $\to 0$) $\quad P = \frac{1}{6} + 0 = \frac{1}{6}$

c, d și e sunt bazate pe: (probabilitatea de a returna x) + (probabilitatea de a returna f()) · (prob x) ... pentru x=0 sau x=1 sau x=2 ....

$P = (\text{prob } x) + \lim$

3.) a.) Cazul cel mai nefavorabil este în funcție de k (k fiind un număr foarte mare)

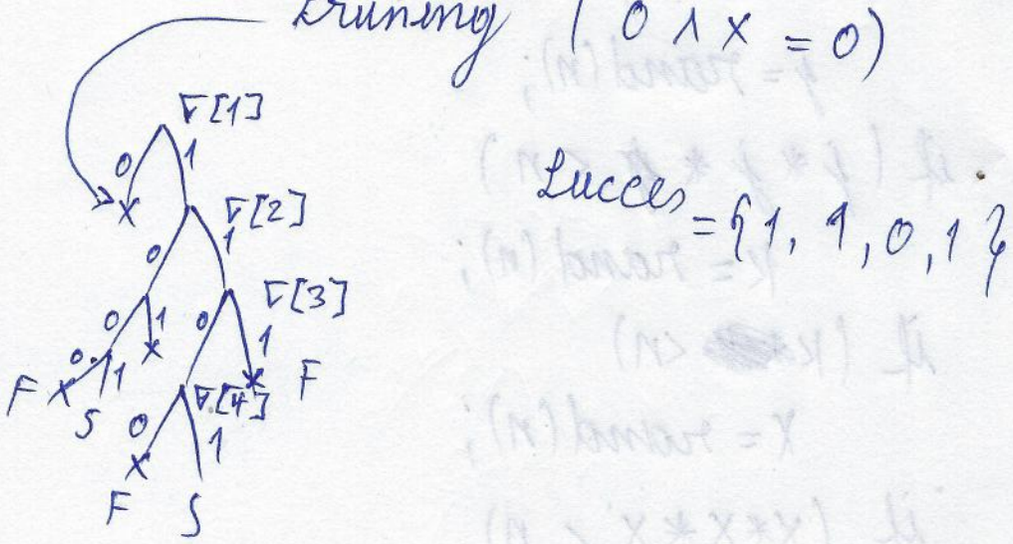
~~$O(n^2)$~~ din cele 2 for-uri - $O(\log^2 n)$

~~$[O(n^2)]^{\frac{k}{2}}$~~ din faptul că se reintră în funcția d() în care se reintră în for, ~~așa că complexitatea~~ rezultă faptul că, complexitatea depinde de k. Cum se face k-2 pentru fiecare reintrare în d() $\Rightarrow \frac{k}{2}$

b.)
```
d(n, m, k)
{ if (n <= 0 || m <= 0 || k <= 0) { return 0; }
  best = 0;
  for (i = 0; i * i < n; i = i + 2)
    {for (j = 0; j * j < m; j = j + 2)
      { if (temp + a[i][j] * b[j][k] > best)
          best = best + a[i][j] * b[j][k];
        if (k <= 1) k = 0; else k = k - 2;}
  return best; }
```

c.) Din faptul că evit reintrarea în cele 2 for-uri de multiple dăți (prin faptul că reintrii în funcție), acest lucru reduce complexitatea față de programul inițial.

4.) a.)



1.) a.) Este o problemă de decizie, având și valoare de returnare fie 0, fie 1. (liniar)

c.) Complexitatea este $O(n^5)$, dacă nu este luat în calcul că intră într-o buclă infinită (din primul, al treilea și/sau al cincilea for, în care valorile rămân 0), altfel este complexitate infinită ($n >= 1$)

d.) Cazul cel mai favorabil este când $n = 0$, evitând astfel toate for-urile, returnând direct 0.

b.)
```
n=0;
i=rand(n);
if (i*i<n)
    j=rand(n);
if (j*j*j<n)
    k=rand(n);
if (k<n)
    x=rand(n);
if (x*x*x<n)
    y=rand(n);
if (y<n)
    if (M[i][j][k][x][y]>10)
        f1=1; break;
return 1;
```