Secventa Kolakoski

Mario-Catalin Pislariu

Martie 6, 2025 Universitatea Bucuresti

1.Introducere

Secventa Kolakoski **1221121221221121122...** este o secventa *distincta si numarabila* alcatuita din $\{1, 2\}$ astfel incat fiecare termen, incepand de la 122, **descrie lungimea blocurilor consecutive** *de cifre din secventa*, fiecare bloc avand o valoare $\neq$ de predecesorul si/sau succesorul sau.

Pentru a fi studiata aceasta a fost descrisa matematic $k_1k_2k_3...$ unde k_i reprezinta lungimea celui de-al i-lea bloc, iar k_1=1.

Ne propunem sa analizam evolutia lungimilor k_j,$\forall k_j \in k_1k_2k_3...k_n$ unde k_n este lungimea blocului n $\geq$2.

2.Breviar istoric

Secventa a aparut pentru prima data intr-o publicatie scrisa de Rufus Oldenburger(1939).Ea a fost descrisa prin intermediul "dinamicii simbolice" folosindu-se referinte la lucrarile anterioare ale lui Morse si Hedlund.

Dinamica simbolica este un concept din matematica reprezentand studiul sistemelor dinamice definte pe un spatiu discret format din secvente infinite de simboluri abstracte. Evolutia sistemului dinamic este definita ca o simpla schimbare a secventei.

Dupa 26 de ani,William Kolakoski(1965) a publicat in *American Mathematical Monthly*(Gazeta Matematica Americandrim) primii 41 de termeni ai secventei impreuna cu 3 probleme:

1.*Care este o regula simpla de descriere a secventei?*
2.*Care este formula celui de-al n-lea termen?*
3.*Este secventa aceasta periodica?*

Un an mai tarziu, Necdet Ucoluk(1966) a raspuns la intrebarile 1 si 3, clasificand secventa ca fiind neperiodica.

Atat Kolakoski, cat si Ucoluk nu stiau aparent de publicatia lui Oldenburger.

Dupa alti 26 de ani, Michael Keane(1991) impreuna cu altii au concluzionat ca densitatea simbolurilor 1 in secventa este de $\frac{1}{2}$ (i.e numarul de apartii al celor doua simboluri este egal). Aceasta problema ramanand inca nedemonstrata pana azi, impreuna cu existenta propriu-zisa a unei densitati in secventa.

Vasek Chvatal(1993) a demonstrat o densitate superioara de 0.501 si o denistate inferioara de 0.499.

Michael Rao(2012) a imbunatatit aceste limite la 0.5001 si 0.4999.

Arturo Carpi(1994) s-a folosit de rezultatele anterioare si a demonstrat ca secventa Kolakoski nu contine cuburi,ci doar patrate de lungimea 2,4,6,18 si 54.

Consideram ca un patrat este un subsir nevid de forma xx(cu lungimea definita ca $\lambda(xx) = 2\lambda(x)$, iar un cub este un subsir nevid de forma xxx,analog avand lungimea $\lambda(xxx) = 3\lambda(x)$.

Tot in 2012 John Nilsson a oferit un algoritm de calcul al secventei $k_1k_2k_3...k_n$ cu complexitatea in timp T=O(n) si spatiu S=O(logn).

Complexitatea in timp a unui algoritm este un cuantificator al timpului necesar rularii algoritmului in cazul cel mai defavorabil.(notatia Big O)

Complexitatea in spatiu se refera la memoria necesara alocarii algoritmului pentru a rezolva o anumita problema.

Aceasta ia in considerare atat partile fixe (independente de input) precum: memoria pentru instructiuni, constante, variabile etc. , cat si partile variabile (care depind de dimensiunea input-ului) precum: memoria de recursie din stiva, variabilele dereferentiate etc..

3.Abstractizare

Notatii

$\mathbb{N}$ este multimea numerelor naturale

ε este sirul nul(de lungime zero)

Σ este multimea alfabet(un cuvant este o secventa finita de simboluri din Σ, un subcuvant este obtinut din eliminarea unor simboluri dintr-un cuvant fara a schimba ordinea celor ramase, aici $\Sigma = \{1, 2\}$)

Σ^*=$\cup_{n\geq 0}\Sigma^n$ este un set finit de siruri peste Σ

Σ^n este un set de 2^n siruri $\sigma_1 \dots \sigma_n$ de lungime exacta n$\geq$0

$\Sigma^{\mathbb{N}}$ este un set numarabil infinit de siruri $\sigma_1\sigma_2\sigma_3\dots$ peste Σ

$\bigcup$=$\Sigma^*, \Sigma^{\mathbb{N}}$(i.e $\Sigma^* \cup \Sigma^{\mathbb{N}}$) este universul sirurilor considerate

$\sigma = \sigma_1 \dots \sigma_n$ este o secventa finita a carei lungimi este $\lambda(\sigma) := n$

σ^j este un sir de j siruri consecutive σ (unde $\sigma^0 = \varepsilon, \sigma^{j+1} = \sigma^j\sigma = \sigma\sigma^j$)

Fie $\sigma = \sigma_1^{j_1}\sigma_2^{j_2}\sigma_3^{j_3}\dots$, unde $\sigma_i \neq \sigma_{i+1}$ si $j_i > 0$, spunem ca $\sigma_n^{j_n}$ este cea de-a n-a runda in σ , iar lungimea ei este j_n.

Putem defini astfel o functie de "aflare a lungimii rundelor" T:U$\rightarrow$U,

$$\mathrm{T}(\sigma_1\sigma_2\sigma_3\dots)=1^{\sigma_1}2^{\sigma_2}1^{\sigma_3}2^{\sigma_4}\dots$$

unde σ=$\sigma_1\sigma_2\sigma_3\dots$ este numita "mama" sirului T(σ) (unele siruri pot sa nu aiba "mama").
Exemplu:
daca σ=12221121("mama")$\rightarrow$T(σ)=122112212112 (la randul ei σ contine o runda de lungime 3 (222), deci nu are "mama" in $\{1, 2\}^*$ putem spune c-a murit "bunica").

Definitie

Secventa Kolakoski k este unicul punct fix al aplicatiei $\sigma \in \Sigma^{\mathbb{N}} \rightarrow$T($\sigma$). (k=T(k) defineste in mod unic k$\in \Sigma^{\mathbb{N}}$)

Demonstratia unicitatii

Fie k=$k_1k_2k_3\ldots$ astfel incat k=T(n). Atunci $k_1k_2k_3\ldots = 1^{k_1}2^{k_2}1^{k_3}\ldots$,unde $1 \leq k_j \leq 2$. Comparand k_1 cu 1^{k_1} observam ca $k_1 = 1$, deci $1k_2k_3\ldots = 12^{k_2}1^{k_3}\ldots$, comparand k_2 cu 2^{k_2} observam ca $k_2 = 2$, deci $12k_3\ldots = 12^21^{k_3}\ldots$(i.e $12k_3\ldots = 1221^{k_3}\ldots$) , analog $k_3 = 2$. Pentru $j \geq 3$, k_j din stanga ecuatiei este definit de $k_1, k_2, \ldots, k_m$ din dreapta ecuatiei, $\forall$m<j ,$m \in \mathbb{N}^*$ acest lucru demonstreaza faptul ca solutia este unica (si ofera totodata un algoritm de computatie al acesteia).

Generalizare

Putem defini atat

$$T_1(\sigma_1\sigma_2\sigma_3\sigma_4\ldots) = 1^{\sigma_1}2^{\sigma_2}1^{\sigma_3}2^{\sigma_4}\ldots$$

cat si

$$T_2(\sigma_1\sigma_2\sigma_3\sigma_4\ldots) = 2^{\sigma_1}1^{\sigma_2}2^{\sigma_3}1^{\sigma_4}\ldots$$

T_1 coincide cu T, iar T_2 difera doar prin inceputul secventei cu 2, in loc de 1.

Spunem ca $\sigma = \sigma_1\sigma_2\sigma_3\sigma_4\ldots$ este atat "mama" lui $T_1(\sigma)$, cat si "mama" lui $T_2(\sigma)$.
Incepand din orice u∈U=$\Sigma^{\mathbb{N}} \cup \Sigma^*\backslash\{\varepsilon\}$ ca si radacina obtinem un arbore binar infinit unde subarborele stang al lui σ este $T_1(\sigma)$, iar cel drept este $T_2(\sigma)$.
In mod analog lui T(σ) se poate demonstra unicitatea solutiei nevide a lui $T_2(\sigma)$, pe care o vom nota $T_2(\hat{k}) = \hat{k}$, unde $\hat{k} \in \Sigma^{\mathbb{N}}$, $\hat{k} = 22112122122\ldots$(corespunzatoare Secventei Kolakoski k cu primul simbol sters , deci k=1$\hat{k}$).

Observatie:
Uneori Secventa Kolakoski este definita ca si $\hat{k}$ in loc de k.

Daca consideram secvente finite, i.e $\sigma \in \Sigma^*$, atunci exista anumite puncte fixe triviale (valori din domeniul de definitie in care valoarea functiei este egala cu argumentul ei)

$$T_1(\varepsilon) = \varepsilon \text{ , } T_2(\varepsilon) = \varepsilon \text{ si } T_1(1) = 1$$

Reprezentarea arborelui

Incepand din $\sigma = 2 \in \Sigma^*$, obtinem urmatorul arbore binar infinit , ale carui noduri sunt secvente finite cu elemente din multime $\{1, 2\}$. In cadrul acestuia apar secvente finite din Secventa Kolakoski k=122112... si $\hat{k} = 221121\ldots$.

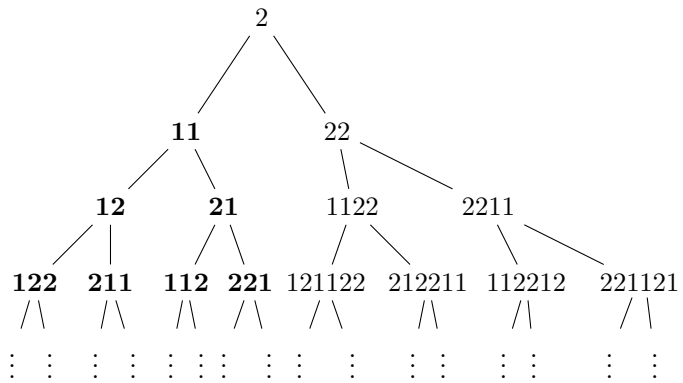


Figura 1. Arborele binar generat de secventele **k** si $\hat{k}$ (noduri secvente finite)

Daca alegem un drum aleator(sau determinist) din radacina arborelui, obtinem o secventa finita care apare(sau poate aparea, fiindca nu este demonstrat) in Secventa Kolakoski.

Cele mai multe descoperiri si presupuneri cu privire la aceasta secventa au implementari ale unor asemenea set-uri.
De exemplu , putem prespune ca densitatea simbolurilor 1 intr-un asemenea set este de $\frac{1}{2}$.

Probleme nedemonstrate

Exista nenumarate probleme nedemonstrate si prespuneri cu privire la Secventa Kolakoski k .
Acestea sunt cateva dintre ele:

Consideram P(A) multimea partilor unei multimi A. ($|P(A)| = 2^n$, unde $n = |A|$).

Pentru un set de cuvinte de lungime n ce apar in k, care este cardinaliatatea P(n) (i.e complexitatea subcuvintelor lui k) ?
∗Dekking a demonstrat in 1981 ca P(n) creste polinomial.

Daca un cuvant $x = x_1x_2...x_n$ apare in k, apare la infinit la un moment anume(infinit de des)? Apare cu spatii marginite ? Apare inversul sau $x^{-1} = x_n...x_2x_1$? Apare complementul sau $x' = x'_1x'_2...x'_n$? (unde 1'=2, 2'=1)

Echidistributie:
Exista o anumita frecventa a simbolurilor 1 in k ? Este aceasta frecventa egala cu $\frac{1}{2}$?

Echidistributia subcuvintelor:
Pentru fiecare cuvant w din k, exista o frecventa a lui w in k?

Tranzitivitate stricta:
Pentru fiecare cuvant w din k, exista frecventa lui w in $k_nk_{n+1}\dots$ uniform in n?

Functia de discrepanta a unei multimi masoara deviatia acesteia de la distributia uniforma.
Reamintim ca Secventa Kolakoski $k = k_1k_2k_3\dots$ unde fiecare bloc k_j are lungimea j, $\forall j \in [1, \infty)$.
Consideram in continuare Functia de Discrepanta Kolakoski $\delta(n) := \sum_{1\leq j\leq n}(-1)^{k_j}$.

Fie doua functii f(x) si g(x), f(x)=o(g(x)) <=> pentru x $\to \infty$ $lim_{x\to\infty}\frac{f(x)}{g(x)} = 0$ (i.e f(x) creste strict mai incet decat g(x)).
Este Functia de Discrepanta Kolakoski $\delta(n) = o(n)$? (creste $\delta(n)$ strict mai incet decat n? Echivalent cu denistatea simbolurilor de 1 in k este de $\frac{1}{2}$)

Echistributia:
Fie un sir $\sigma = \sigma_1\dots\sigma_n \in \Sigma^*, \lambda(\sigma) = n$ (i.e lungimea lui σ).

Fie $x \in \Sigma$, definim $\lambda_x(\sigma)$ ca numarul de aparitii ale simbolului x in σ. Exemplu: $\lambda_1(1121) = 3, \lambda_2(1121) = 1$.
In continuare abreviem $\lambda_x(k_1 \dots k_n)$ cu $\lambda_x(n)$.
Secventa k este echidistribuita <=>

$$lim_{n\to\infty} \frac{\lambda_1(n)}{n} = lim_{n\to\infty} \frac{\lambda_2(n)}{n} = \frac{1}{2}$$

Dar cum $\lambda_1(n) + \lambda_2(n) = n$ => echivalent cu:

$$lim_{n\to\infty} \frac{\lambda_1(n)+\lambda_2(n)}{n} = 1$$

Cum:

$$\sum_{j=1}^{n} k_j = \lambda_1(n) + 2\lambda_1(n) => lim_{n\to\infty} \frac{1}{n} \sum_{j=1}^{n} k_j = \frac{3}{2}$$

In alte cuvinte "factorul de expansiune"(indica cat de mult se mareste secventa) care merge de la sirul "mama" $k_1 \dots k_n$ la sirul $1^{k_1}2^{k_2}1^{k_3} \dots (?)^{k_n}$ este asimptotic egal cu $\frac{3}{2}$.

O alta conditie echivalenta este $\delta(n) = o(n)$ <=>

$$\delta(n) := \sum_{1\le j\le n}(-1)^{k_j} = \lambda_2(n) - \lambda_1(n) = n - \lambda_1(n) - \lambda_1(n) = n - 2\lambda_1(n)$$

Observatie:
Daca k_j s-ar comporta ca niste variabile aleatoare independenta (ceea ce nu fac), ne-am astepta ca $\delta(n) = O(n^{\frac{1}{2}}\text{loglog n})$. (i.e $\delta(n)$ este marginita superior de $n^{\frac{1}{2}}$loglog n)

Algoritmi

Deoarece demonstrarea echidistributiei lui k nu este realizabila prin intermediul metodelor teoretice, s-a incercat computarea lui $\delta(n)$ pentru diferite valori mari ale lui n si analizarea acestori computari pentru a observa daca se obtine o evidenta numerica care sa afirme\sau sa nege echidistributia.
Din ceea ce cunoastem computatile din intervalul $1 \le n \le 5 * 10^{17}$ afirma existenta acesteia.

Determinarea secventei

Vom descrie si analiza in cele ce urmeaza cativa algoritmi de generare a Secventei Kolakoski k pentru un numar dat n.
Algoritmul intuitiv de generarea a secventei $k = k_1k_2 \dots k_n$ intr-un mod liniar foloseste faptul ca secventa k este un punct fix al functiei de "aflare a lungimii rundelor" $T_1(\sigma_1\sigma_2\sigma_3 \dots) = 1^{\sigma_1}2^{\sigma_2}1^{\sigma_3}2^{\sigma_4} \dots$.

Reamintim ca x' este complementul lui $x \in \Sigma$, unde $\Sigma = \{1, 2\}$ (pentru eficienta se poate lua $\Sigma\{0, 1\}$), iar 1'=2,2'=1.

Complexitate spatiu/timp O(n)

Folosim un vector A cu lungimea de n+1, $A = A_1 \dots A_{n+1}$ si indicii i,j ,unde A_i este elementul de pe pozitia i din A.

Pseudocod:

$(A_1, A_2, i, j) \leftarrow (1, 1', 2, 2)$;
while $i \leq n$ do
  if $A_j = 1$ then $(A_i, i, j) \leftarrow (A'_{i-1}, i + 1, j + 1)$
  else $(A_i, A_{i+1}, i, j) \leftarrow (A'_{i-1}, A'_{i-1}, i + 2, j + 1)$

Rezultat:
  $A_1 \dots A_n = k_1 \dots k_n$

Acest algoritm are complexitatea in timp O(n) si in spatiu O(n), iar i creste cu 2 cand $A_j = 2$, dupa cum este ilustrat in urmatorul tabel.(valorile sarite in "( )")

i	j	A_i	A_j
1	1	1	1
2	2	2	2
(3)		2	
4	3	2	1
(5)		1	
6	4	2	1
7	5	1	1
8	6	2	2
(9)		2	
10	7	1	1

Tabel 1: Descriere cod n=10

Complexitate spatiu O(n) , timp O(log n)

In anul 2012 Nilsson a reusit sa imbunatateasca acest algoritm reducand spatiul necesar generarii $k = k_1 \dots k_n$ de la O(n) la O(log n).(acelasi si pentru computarea $\delta(n)$).

El s-a folosit de generarea Secventei print-o procedura recursiva, unde referinta la A_j a fost inlocuita cu un apel recursiv la aceeasi procedura, daca nu are loc $j \leq 2$.

Pseudocod:

//functia genereaza secventa de la a 3-a pozitie
//se apeleaza de n-2 ori increment(0)

int increment(int k)
if $P_k = 0$ then $P_k \leftarrow 22$
if $P_k = 11$ then $P_k \leftarrow 1$, return 1
else if $P_k = 22$ then $P_k \leftarrow 2$, return 2
else if $P_k = 1$ then $P_k \leftarrow increment(k+1)$ ==1 ? 2:22,return 2
else if $P_k = 2$ then $P_k \leftarrow increment(k+1)$ ==1 ? 1:11,return 1

//afisarea valorilor se face prin afisarea lui increment(0) de n-2 ori

Observatie:
a==b ? x:y echivalent cu if a=b then a=x else a=y

Procedura recursiva de mai sus poate fi explicata prin intermediul unei structuri arborescente(totusi, ea nu construieste un arbore in timpul executiei).
Secventa Kolakoski K descrie un arbore binar, ale carui noduri sunt simbolurile $\{1, 2\}$ si poate fi generat intuitiv dupa urmatoarele reguli:

1.Pornim din radacina ($Nod_{curent} = 2$)
2.Parcurgem fiecare Nivel de la stanga la dreapta
3.Nodul curent da numarul de aparitii ale nodului de sub el "copil"
4.$Nod_{curent} \neq Nod_{anterior}$

5.Daca Nod_{curent}=primul din $Nivel_{nou}$, $Nod_{anterior}$=ultimul din $Nivel_{anterior}$
6.Altfel $Nod_{anterior} = Nod_{curent-1}$

Observatie:
Dupa parcurgerea unui nivel,cunoastem nr. de aparitii ale tuturor nodurilor de pe nivelul urmator, dar nu valoarea lor.

Ultimul nod de pe $Nivel_{anterior}$ este cel mai din **dreapta**.
$Nod_{curent-1}$ este nodul cel mai apropiat din **stanga**.

Ambii algoritmi genereaza valorile de la pozitia a 3-a a secventei, vom alege radacina ca fiind simbolul 2 de pe aceasta pozitie.

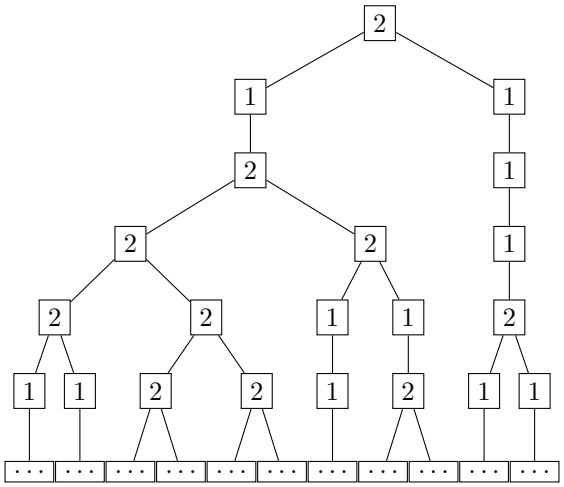


Figura 2. Structura arborescenta a Secventei k

Primul algoritm prezentat "parcurge" arborele descris mai sus , generandu-l de fiecare data.
Practic i "tine minte" nr. de noduri parcurse si folosim A_{i-1} pentru a afla nodul anterior, iar j "tine minte" index-ul lui A_j care dicteaza nr. de aparitii ale nodului curent.
Complexitatea sa se datoreaza faptului ca instantierea nodului curent A_i necesita accesarea elementului anterior de fiecare data, si echivalent parcurgerea arborelui de la index-ul j.

Observam in Tabelul 1. cum diferenta dintre i si j se mareste cu fiecare valoare de 2 a lui j, adica timpul necesar "sa ne uitam" in spate devine din ce in ce mai mare.

In al doilea algoritm, este rezolvata aceasta problema prin gener-

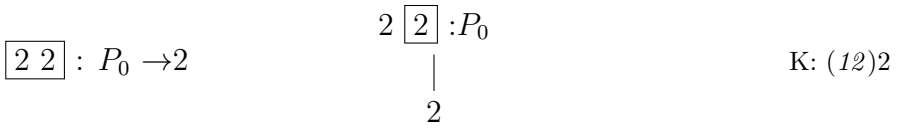
area arborelui de jos in sus, recursiv.In parcurgerea arborelui contorizam simbolurile pe care le vedem in nodurile acestuia, adaugand de fiecare data nodurile "parinte".

Generam dinamic doar partea arborelui in care ne aflam in momentul respectiv,stocand nodurile prin intermediul vectorului P, iar dupa ce obtinem valoarea urmatoare actualizam elementul P_k pentru a "schimba partea".

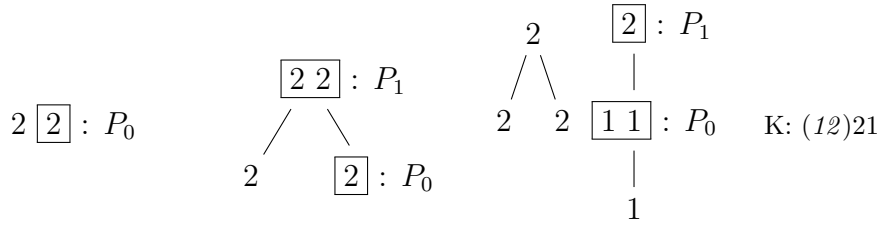
De precizat ca P nu memoreaza intreg arborele, ci doar nodurile copil ale nodulului ce urmeaza generat, in momentul in care valoarea nodului curent este 2 sau 1 apeland recursiv functia pentru urmatorul nod k_{+1}, aflandu-i "parintii".

In cele ce urmeaza vom arata cum genereaza concret algoritmul primele 6 (8 cu (12)) valori ale Secventei K:

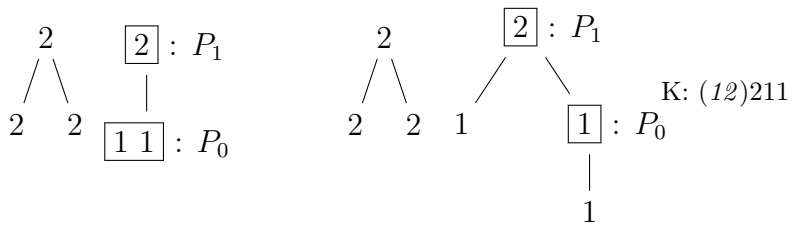
Observatie:
Valorile returnate de P_0 reprezinta mereu urmatorul termen al secventei.



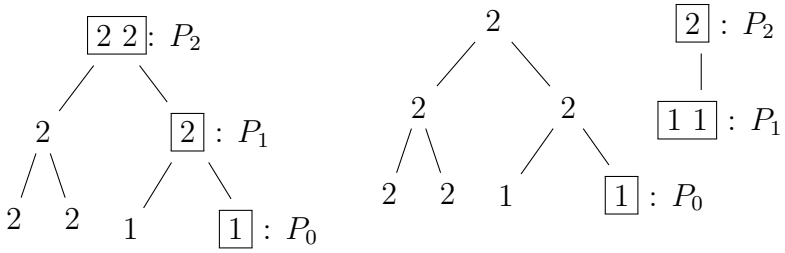
P_0 este neinitializat , deci ia valoarea 22, dupa care ia valoarea 2, returnand 2.

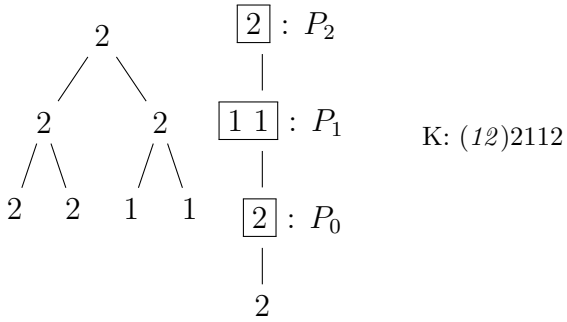


In cea de-a doua incrementare a lui P_0 generam urmatorul nod P_1, fiind neinitializat ia valoarea 22, returnand 2 prin recursie catre P_0 si schimbandu-si valoarea la 2. Cum $2 \neq 1$, P_0 ia valoarea 11 , returnand 1.

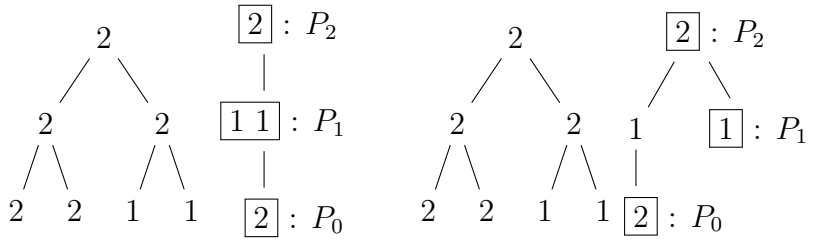


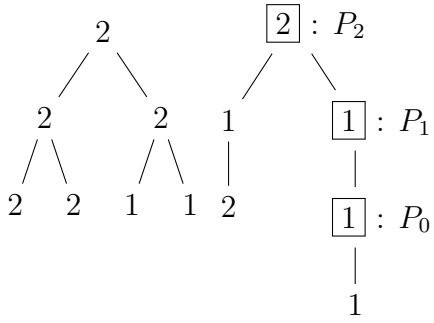
Conform graficului din Figura 2. P_1(nivelul 1) contine 2 noduri, algoritmul "schimba partea", P_0 ia valoarea 1 prin recursie si returneaza 1.





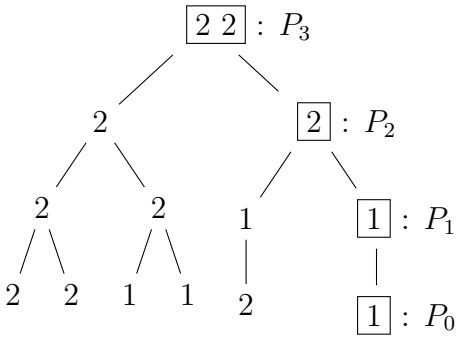
A patra incrementare a lui P_0, generam urmatorul nod P_2, ia valoarea 22, dupa care analog 2. Practic de fiecare data cand P_k contine 22 neglijam primul simbol 2 si construim un nou arbore in nodurile copil ale acestuia.Returneaza 2. De precizat ca algoritmul face acest lucru de jos in sus, practic "vede" mereu ce este in P_0 si construieste de jos in sus "parintii". De exemplu aici, P_0 implica intai recursia P_1, care si ea apeleaza recursia P_2(ia val. 22 si returneaza 2). In momentul in care ultimul nivel actual (aici P_2) termina recursia se actualizeaza si "copii" sai.

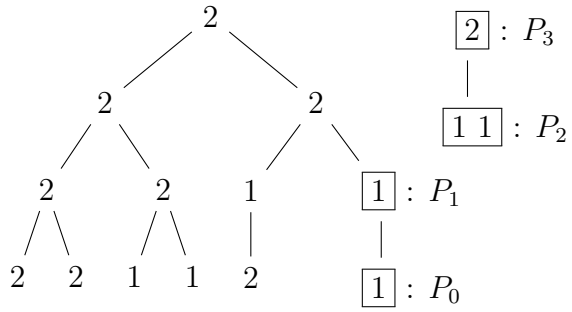




K: (*12*)21121

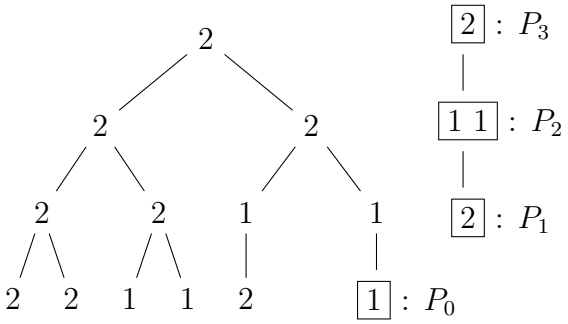
Observam iarasi din graficul din Figura 2. ca P_2(nivelul 2) contine 2 noduri, deci "schimbam partea",primul simbol din P_1 a fost deja folosit la incrementarea anterioara (i.e val. sa este 11), analog P_1 ia valoarea 1 si returneaza 1,iar P_0 se actualizeaza prin recursie la 1 (increment(1)$\rightarrow 1 = 1$, deci $P_0 = 1$ return 1).

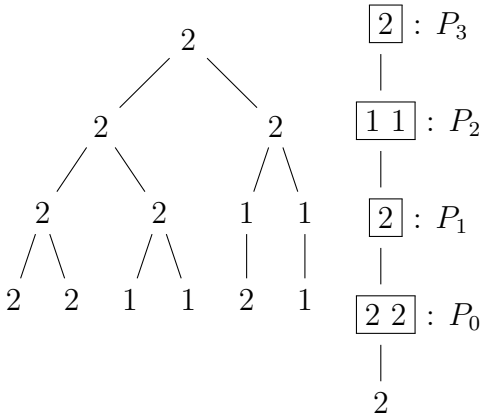




K: (*12*)21121

A sasea incrementare a lui P_0, generam urmatorul nod P_3, deoarece recursia lui P_2 implica P_3 ($P_2 = 2$), acesta ia valoarea 22 si "negiljam" primul simbol.In momentul intoarcerii din recursie P_2 ia valoarea 11 ($1 \neq 2$).





K:(*12*)211212

Cum recursia P_2 returneaza 1, P_1 ia valoarea 2 (valoarea initiala a lui $P_1 = 1$, implica recursia P_2, cum $1 = 1$ ia prima valoare dintre 2:22) si returneaza 1, deci $P_0 = 22$, care returneaza 2. (analog $P_0 = 1$ implica recursia P_1 , cum $1 \neq 2$ ia a doua valoarea dintre 2:22)

Pentru n termeni, functia se autoapeleaza recursiv de O(log n) ori , cu fiecare nivel de mai sus avand aproximativ $\frac{2}{3}$ din valorile nivelului anterior .Fiecare nivel din recursie are un spatiu constant de memorie pentru variabilele locale, deci complexitatea in timp este de O(n), iar cea in spatiu este O(log n). Acesta este cel mai eficient algoritm gasit pana acum pentru a generea intreaga Secventa K.

Este posibila insa generarea unei singure valori din secventa (sau a unei valori $\delta(n)$) cu o complexitate in timp $<< O(n)$.
Primul pas este convertirea algoritmului recursiv,intr-unul iterativ care sa genereze secventa. Apoi, fiindca fiecare nod determina un anumit numar de copii, vrem sa stocam informatia necesara pentru a ajunge la nodul dorit fara sa generam intreaga secventa. Acest lucru se realizeaza prin intermediul unui look-up table, in care memoram doar nodurile $P_k = 2$ si nodurile lor copil, sarind peste celelalte. In cazul in care ni se cere o valoare a unui nod peste care am fi sarit, apelam la algoritmul iterativ.

Automate

Un FST(Finite State Transducer) este un model computational folosit pentru a manipula masini cu stari finite, care mapeaza secventele de input in secvente de output. Acestea sunt adesea folosite in procesarea naturala a limbajelor(NLP natural language processing) pentru task-uri precum: analiza morfologica,corectarea greselilor de ortografie,recunoasterea vocala si traducere.

Secventa Kolakoski K poate fi reprezentata printr-o aplicatie unica a unui astfel de FST, reprezentata prin urmatorul tabel.

stare initiala	input	output	stare finala
1	1	1	2
1	2	11	2
2	1	2	1
2	2	22	1

Tabel 2: FST K

Analog putem desena tabelul pentru FST $K^2 = K \circ K$:

stare initiala	input	output	stare finala
11	1	1	22
11	2	12	21
12	1	2	21
12	2	21	22
21	1	11	12
21	2	1122	11
22	1	22	11
22	2	2211	12

Tabel 3: FST K

In continuare $K^3 = K \circ K \circ K$, iar inductiv, $K^n = K \circ \ldots \circ K$ (de n ori), determinand un tabel cu 2^{n+1} coloane.

In anul 2012 Michael Rao s-a folosit de aceste compuneri ale FST-urilor K pentru a determina Secventa Kolakoski K si a imbunatati limitele densitatii simbolurilor de 1 si 2 la o densitate superioara de 0.501 si una inferioara de 0.499. Rao nu si-a descris metoda in detaliu, insa este foarte plauzibil ca acesta a ales un anumit M determinat de resursele computationale disponibile, calculand K,K^2,...,K^M(sau anumite subsiruri), apoi s-a folosit de aceste FST-uri pentru a calcula $k_1, \ldots, k_n$, deci $\delta(1), \ldots, \delta(n)$(sau anumite subsiruri ale lor).

In incheiere, Secventa Kolakoski a reusit sa atraga atentia cercetatorilor din domeniul matematicii si informaticii de mai bine de 8 decenii, aceasta continua sa fie o "engima" pentru cercetatori si pasionati, fiind surprinzatoare prin simplitatea enuntului si dificultatea demonstratiilor presupunerilor cu privire la ea.

Bibliografie:

(Australian National University and University of Newcastle) Fast algorithms for the Kolakoski sequence:
https://maths-people.anu.edu.au/~brent/pd/Kolakoski-UNSW.pdf

Wikipedia Kolakoski Sequence
https://en.wikipedia.org/wiki/Kolakoski_sequence

(Universitatea din Craivova)Probabilitatii:
https://www.ucv.ro/pdf/departamente_academice/dma/suporturi_curs/Matematica-curs-STERBETI_CATALIN.pdf

FST:https://en.wikipedia.org/wiki/Finite-state_transducer

(Universitatea din Biefeld) A Space Efficent Algorithm for the Calculation of the Digit Distribution in the Kolakoski Sequence https://arxiv.org/pdf/1110.4228

The Kolakoski tree(python):
https://11011110.github.io/blog/2016/10/13/kolakoski-tree.html

ChatGPT: chatgpt.com