Nagyméretű adathalmazok kezelése

Adatfolyamok (1)

Ureczky Bálint (ABDA7Z)

Forrás:
Data Streams: Algorithms and Applications
/S. Muthukristan/

Kivonat

Input: gyorsan érkezik

Tárhely: limitált, < O(n)

Végigolvasás: egyszer, párszor

Alkalmazások:

- IP hálózat forgalom analízis
- Szöveg folyamokban bányászat
- Masszív adathalmazok

1. Rejtvény

Hiányzó szám megtalálása

$$\Pi = \{1, 2, \ldots, n\} \textit{ permutációja}$$

$$\Pi_{-1} = \Pi \setminus \{\Pi[n]\} = \{\Pi[1], \Pi[2], \ldots, \Pi[n-1]\}$$

<u>Feladat:</u> Adott Π_{-1} esetén: $\Pi[n] = ?$

1. Rejtvény

Általánosítás:

$$\Pi = \{1, 2, \ldots, n\} \textit{ permutációja}$$

$$\Pi_{-k} = \{\Pi[1], \Pi[2], \ldots, \Pi[n-k]\}$$

<u>Feladat:</u> Adott Π_{-k} esetén: $\{\Pi[n-k+1], \ldots, \Pi[n]\} = ?$

1.Rejtvény / 1.megoldás

Triviális megoldás:

A számok nyilvántartása egy n-dimenziós tömbben.

Algoritmus:

$$B \leftarrow [000...0]$$

$$for\ i = 1..n$$

$$B[\Pi[i]] \leftarrow 1$$

$$\boxed{\Pi[i] \in \{e \mid B[e] = 0\}}$$

Memóriaigény:
O(n) (sok)

1.Rejtvény / 2.megoldás

Elegendő a számok összegét figyelni:

$$\left.\begin{aligned} S_{\Pi} &= 1+2+\ldots+n \\ S_{\Pi-1} &= S_{\Pi} - \Pi[n] \end{aligned}\right\} \Pi[n] = S_{\Pi} - S_{\Pi-1}$$

Algoritmus:

$$S \leftarrow S_{\Pi} = 1+2+\ldots+n = \frac{n(n+1)}{2}$$

$$for \quad i = 0..n-1$$

$$S \leftarrow S - \Pi[i]$$

$$\boxed{\Pi[n] = S}$$

Memóriaigény:
log(n*(n+1)/2)~
2log(n) = O(logn)
Min log(n) szükséges (eredmény eltárolása)
Lehet-e a 2logn-nél kevesebb?

1.Rejtvény / 3.megoldás

Összeg helyett vegyük a bitenkénti XOR-zást(⊗):

$$\left.\begin{aligned} S_{\Pi} &= 1 \otimes 2 \otimes \ldots \otimes n \\ S_{\Pi-1} &= S_{\Pi} \otimes \Pi[n] \end{aligned}\right\} \Pi[n] = S_{\Pi} \otimes S_{\Pi-1}$$

Algoritmus:

$$S \leftarrow S_{\Pi} = 1 \otimes 2 \otimes \ldots \otimes n$$

$$for \quad i = 0..n-1$$

$$S \leftarrow S \otimes \Pi[i]$$

$$\boxed{\Pi[n] = S}$$

Memóriaigény:
log(n)
OPTIMÁLIS !
(csak k=1-re)

1.Rejtvény / 2.megoldás, k=2-re

K=2 esetén a számok összegén kívül a számok négyzetösszegét is figyeljük!

$$\left.\begin{aligned} S_{\Pi} &= \sum_{i=1}^{n} \Pi[i] \\ S_{\Pi}^2 &= \sum_{i=1}^{n} \left(\Pi[i]\right)^2 \\ S_{\Pi-2} &= \sum_{i=1}^{n-2} \Pi[i] \\ S_{\Pi-2}^2 &= \sum_{i=1}^{n-2} \left(\Pi[i]\right)^2 \end{aligned}\right\} \begin{aligned} s_1 &= S_{\Pi} - S_{\Pi-2} \\ s_2 &= S_{\Pi}^2 - S_{\Pi-2}^2 \end{aligned}$$

Amiből megoldható az egyenletrendszer

$$\left.\begin{aligned} \Pi[n] + \Pi[n-1] &= s_1 \\ \left(\Pi[n]\right)^2 + \left(\Pi[n-1]\right)^2 &= s_2 \end{aligned}\right\} \left\{\Pi[n], \Pi[n-1]\right\} = \frac{s_1 \pm \sqrt{2s_2 - s_1^2}}{2}$$

1.Rejtvény / 2.megoldás, k-ra

Hatványösszegek:

$$s_P\left(x_1, x_2, \ldots, x_k\right) = \sum_{i=1}^{k} \left(x_i\right)^P$$

Memóriaigény:

$$\log\left(n\right) + \log\left(n^2\right) + \ldots + \log\left(n^k\right) =$$

$$= \left(1 + 2 + \ldots + k\right)\log\left(n\right) =$$

$$\frac{k\left(k+1\right)}{2}\log\left(n\right)$$

Egyenlet megoldása: k-adfokú egyenletrendszer..

1.Rejtvény / 4.megoldás, k-ra

Szimmetrikus polinomok:

$$\sigma_P\left(x_1, x_2, \ldots, x_k\right) = \sum_{j_1 < \ldots < j_P}^{k} \left(x_{j_1} \cdot x_{j_2} \cdot \ldots \cdot x_{j_P}\right)$$

Pl:

$$\sigma_1\left(x_1, x_2, \ldots, x_k\right) = x_1 + x_2 + \ldots + x_k$$

$$\begin{aligned}\sigma_2\left(x_1, x_2, \ldots, x_k\right) = {} & x_1x_2 + x_1x_3 + \ldots + x_1x_k + \\ & + x_2x_3 + x_2x_4 + \ldots + x_2x_k + \\ & + \ldots + \quad x_{k-1}x_k\end{aligned}$$

Előállítás:

$$\sigma_P\left(x_1, x_2, \ldots, x_{k+1}\right) = \sigma_P\left(x_1, x_2, \ldots, x_k\right) + x_{k+1} \cdot \sigma_P\left(x_1, x_2, \ldots, x_k\right)$$

2.Rejtvény: Mutató és üldöző

Adott n+1 mutató, amik egymásra mutatnak, de nem az utolsóra:

A skatulya-elv értelmében lesz olyan, amire több is mutat.

Feladat: keressünk egy ilyet.

2.Rejtvény/1.megoldás

Járjuk végig az elemeket és egy n-dimenziós tömbbe jegyezzük fel, hogy egy adott elemre mutattak-e már, vagy mutattak-e többen is.

Memóriaigény: O(n) bit (nagyon pazarló!)

n lekérdezés

2.Rejtvény/2.megoldás

Vegyük sorra az elemeket (1..n), melyikre mutathat több elem.
Ehhez járjuk végig az összes elemet és számoljuk meg hány mutat rá.

A memóriaigény csökken, de a lekérdezések számának rovására:

Memóriaigény: O(logn) bit
n^2 lekérdezés

2.Rejtvény/3.megoldás

Alkalmazzuk az „oszd meg és uralkodj” elvet és először csak az egyik felére koncentráljunk, hogy azon elemekre hányan mutatnak összesen. Az n+1 mutatóból a többi a másik felére fog mutatni.

A skatulya-elv értelmében ki tudjuk választani azt a félt, amelyikre továbbra is fennáll, hogy többen mutatnak rájuk, mint az elemszámuk és így egészen addig folytathatjuk a felezést, amíg meg nem találunk egy olyan elemet, amire legalább 2-en mutatnak.

Memóriaigény: O(logn) bit

nlogn lekérdezés

2.Rejtvény

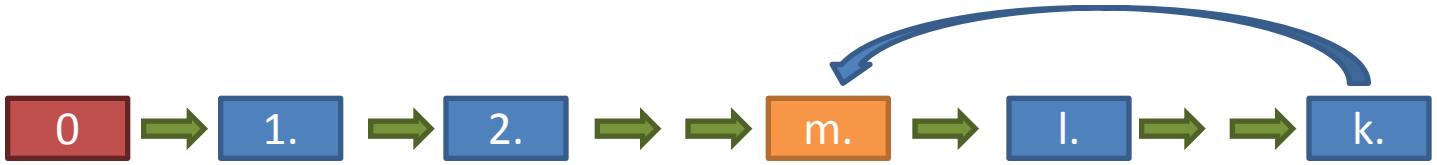
Az nyilvánvaló, hogy szintén legalább O(logn) bit szükséges

Ekkor viszont O(logn/loglogn) végigolvasás kell.

2.Rejtvény/4.megoldás

Tegyük fel, hogy véletlen hozzáférésünk van az elemekhez.

Az utolsó elemtől haladjunk visszafele, kövessük (üldözzük) a mutatókat és előbb-utóbb egy olyan elemhez kell jutnunk, amire már mutattak. Csak ezekkel az elemekkel foglalkozzunk:



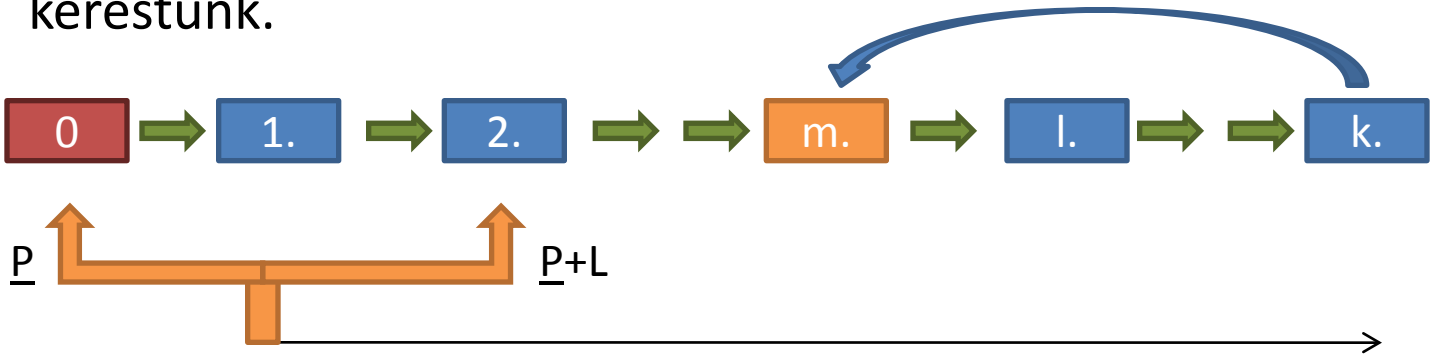
2.Rejtvény/4.megoldás

A dolog érdekessége, hogy ennek detektálásához elegendő log n bit.

1.lépés: Kövessük a mutatót n lépésen keresztül, így biztosan bekerülünk a körbe.

2. lépés: Mérjük le a kör hosszát, azaz kövessük a mutatókat addig, amíg vissza nem érünk ugyanehhez az elemhez.

3. lépés: a kör hosszának (L) ismeretében induljunk ki a 0. és L. elemből (a lista szerinti számozásban) és lépegessünk 1-esével, amíg a két mutatónk nem mutat ugyanoda. Ez egy olyan elem lesz, amit kerestünk.



Memóriaigény: O(logn) bit

O(n) lekérdezés